# Computational Models of Relations in Text and Knowledge Graphs for Logical Reasoning and Graph-Text Conversion

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München


eingereicht von
Martin Schmitt

München, den 24. August 2021

Erstgutachter: *Prof. Dr. Hinrich Schütze*
Zweitgutachter: *Prof. Dr. Axel-Cyrille Ngonga Ngomo*
Drittgutachter: *Prof. Dr. Benjamin Roth*

Tag der Einreichung: 24. August 2021
Tag der mündlichen Prüfung: 02. März 2022

**Eidesstattliche Versicherung**
(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5.)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig ohne unerlaubte Beihilfe angefertigt ist.

München, den 24. August 2021

Martin Schmitt

# Abstract

Knowledge graphs (KGs) store facts in the form of triples that contain two entities and the relation between them. This relation is usually a standardized part of the KG schema and could thus be denoted by any arbitrary unique identifier, e.g., an integer. Typically, however, each relation is assigned a short phrase, such as *born in*, to describe its meaning intuitively. In this way, the triple resembles a natural language statement although it often lacks the fluency of a real sentence.

It is common to reason about facts from KGs and texts to infer new knowledge. An important yet underexplored setting for this is relation inference in context (RIC). Here, the validity of a triple $(e_1, r, e_2)$ has to be determined based only on a single other relation $r'$ that is known to hold for $e_1, e_2$. After disambiguating the relations $r, r'$ with the context, this boils down to deciding relation inclusion, i.e., logical implication. If the relation arguments $e_1, e_2$ are not restricted to entities, i.e., we allow for arbitrary noun phrases, and if the relations are only sequences that lead to wellformed sentences, then RIC is also a fundamental task for many use cases in natural language processing, such as question answering, event coreference, and recognizing textual entailment.

This thesis examines relations in texts and KGs in two settings: Relation inference and text-graph conversion. The latter builds the bridge between KGs and texts by improving KG interpretability (graph-to-text) and making textual facts accessible to machines (text-to-graph). In the first publication, we extract a large-scale KG from text with lemmatized dependency paths as relations and create a human-annotated RIC benchmark from it. We evaluate and analyze existing methods and propose several new methods based on argument-relation co-occurrence information. The second paper explores the use of pretrained language models (PLMs) for RIC and investigates the influence of combining them with textual patterns expressing entailment or non-entailment. In the third paper, we replace the textual patterns with strings of artificial tokens, i.e., tokens that do not exist in the language model's vocabulary and whose continuous representations can therefore be learned freely. We find that artificial tokens improve performance even more than natural text.

The fourth publication describes an unsupervised system for both graph-to-text and text-to-graph conversion. In the fifth paper, we propose a new way of injecting graph structure into the text-to-text Transformer architecture and evaluate it on graph-to-text generation. The sixth paper investigates the use of PLMs for graph-to-text generation and analyzes possible reasons for their success in this task. In the seventh paper, insights from the previous chapters are used to implement a practical text-to-graph system. Trained on image captions and scene graphs, the automatically generated graphs from this system are used to improve few-shot

learning of scene graph extraction from images.

# Zusammenfassung

Wissensgraphen (engl. *knowledge graphs*, KGs) speichern Fakten in Form von Tripeln, die zwei Entitäten und die Relation zwischen ihnen enthalten. Diese Relation ist normalerweise ein standardisierter Teil des KG-Schemas und könnte daher durch einen beliebigen eindeutigen Bezeichner, z. B. eine Zahl, ausgedrückt werden. In der Regel wird jedoch jeder Relation eine kurze Phrase zugeordnet, wie z. B. *born in*, um ihre Bedeutung intuitiv zu beschreiben. Auf diese Weise ähneln Tripel Aussagen in natürlicher Sprache, wenn sie auch meist nicht so flüssig sind wie ein echter Satz.

Es ist üblich, durch logisches Schließen aus den Fakten in KGs und Texten neues Wissen abzuleiten. Ein wichtiger, aber noch nicht ausreichend erforschter, Bereich hierfür ist Relationsinferenz im Kontext (engl. *relation inference in context*, RIC). Hier soll der Wahrheitswert eines gegebenen Tripels $(e_1, r, e_2)$ bestimmt werden, und zwar nur anhand einer einzigen anderen Relation $r'$, von der bekannt ist, dass sie für $e_1, e_2$ gilt. Nach der Disambiguierung der Relationen $r, r'$ durch den Kontext läuft dies darauf hinaus zu entscheiden, ob eine Relationsinklusion, d. h. logische Implikation, vorliegt. Wenn die Argumente $e_1, e_2$ der Relation nicht auf Entitäten beschränkt sind, d. h. wenn sie beliebige Substantivphrasen sein können, und wenn für die Relationen nur Sequenzen erlaubt sind, die zu wohlgeformten Sätzen führen, dann ist RIC auch eine grundlegende Aufgabe für viele Anwendungsfälle in der natürlichen Sprachverarbeitung, wie z. B. die Beantwortung von Fragen (engl. *question answering*), Koreferenz von Ereignissen und das Erkennen logischer Konsequenz zwischen Texten.

In dieser Arbeit werden Relationen in Texten und KGs in zwei Bereichen untersucht: Relationsinferenz und Text-Graph-Konvertierung. Letztere schlägt die Brücke zwischen KGs und Texten, indem sie die Interpretierbarkeit von KGs verbessert (Graph-zu-Text) und textuelle Fakten für Maschinen zugänglich macht (Text-zu-Graph). In der ersten Veröffentlichung extrahieren wir einen großen KG aus Text mit lemmatisierten Dependenzpfaden als Relationen und erstellen daraus eine manuell annotierte RIC-Benchmark. Wir evaluieren und analysieren bestehende Methoden und schlagen mehrere neue, auf Argument-Relation-Kookkurrenz basierende Methoden vor. Der zweite Beitrag befasst sich mit der Verwendung von vortrainierten Sprachmodellen (engl. *pretrained language models*, PLMs) für RIC und untersucht den Einfluss einer Kombination von PLMs mit textuellen Mustern, die jeweils Präsenz oder Absenz von logischer Folge ausdrücken. In der dritten Arbeit ersetzen wir die textuellen Muster durch Zeichenketten aus künstlichen Token, d. h. Token, die nicht im Vokabular des Sprachmodells vorhanden sind und deren kontinuierliche Repräsentationen daher frei gelernt werden können. Wir stellen fest, dass künstliche Token zu sogar noch besseren Ergebnissen führen als

natürlicher Text.

Die vierte Veröffentlichung beschreibt ein unüberwachtes System für die Umwandlung von Graphen in Text und von Text in Graphen. In der fünften Veröffentlichung wird ein neuer Mechanismus zur Einbindung von Graphstruktur in die Text-zu-Text-Transformer-Architektur vorgeschlagen und für seine Tauglichkeit zur Graph-zu-Text-Generierung evaluiert. Der sechste Beitrag untersucht die Verwendung von PLMs für die Graph-zu-Text-Generierung und analysiert mögliche Gründe für ihren Erfolg bei dieser Aufgabe. Im siebten Beitrag werden die Erkenntnisse aus den vorherigen Kapiteln genutzt, um ein praktisches Text-zu-Graph-System zu implementieren. Nachdem es auf Bildunterschriften und Szenegraphen trainiert wurde, werden die automatisch generierten Graphen dieses Systems verwendet, um die Extraktion von Szenegraphen aus Bildern mit wenig annotierten Bilddaten (engl. *few-shot scene graph extraction*) zu verbessern.

# Publications and Declaration of Co-Authorship

**Chapter 2** corresponds to the following publication:

> Martin Schmitt and Hinrich Schütze (2019). "SherLIiC: A Typed Event-Focused Lexical Inference Benchmark for Evaluating Natural Language Inference". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy, pp. 902–914

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my advisor who assisted me in improving the draft.

**Chapter 3** corresponds to the following publication:

> **Martin Schmitt** and Hinrich Schütze (2021). "Language Models for Lexical Inference in Context". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online, pp. 1267–1280

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my advisor who assisted me in improving the draft.

**Chapter 4** corresponds to the following publication:

> **Martin Schmitt** and Hinrich Schütze (2021). "Continuous Entailment Patterns for Lexical Inference in Context". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic, pp. 6952–6959

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

**Chapter 5** corresponds to the following publication:

> **Martin Schmitt**, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze (2020). "An Unsupervised Joint System for Text Generation from Knowledge Graphs and Semantic Parsing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, pp. 7117–7130

After an initial brainstorming with Sahand Sharifzadeh, I conceived of the original research contributions. I performed all implementations and evaluations. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my coauthors who assisted me in improving the draft.

**Chapter 6** corresponds to the following publication:

> **Martin Schmitt**, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze (2021). "Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs". In: *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*. Mexico City, Mexico, pp. 10–21

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the article and did most of the subsequent corrections. I regularly discussed this work with my coauthors who assisted me in improving the draft.

**Chapter 7** corresponds to the following publication:

> Leonardo F. R. Ribeiro, **Martin Schmitt**, Hinrich Schütze, and Iryna Gurevych (2021). "Investigating Pretrained Language Models for Graph-to-Text Generation". In: *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*. Online, pp. 211–227

Leonardo F. R. Ribeiro and I conceived of the original research contributions. Leonardo F. R. Ribeiro performed all implementations and evaluations. He wrote the initial draft of the article and did most of the subsequent corrections. I assisted him in designing the experiments for assessing the influence of graph structure

and contributed significantly to their description and discussion in section 6 of the article. We regularly discussed this work with our coauthors and improved the draft together.

**Chapter 8** corresponds to the following publication:

> Sahand Sharifzadeh, Sina Moayed Baharlou, **Martin Schmitt**, Hinrich Schütze, and Volker Tresp (2022). "Improving Scene Graph Classification by Exploiting Knowledge from Texts". In: *Proceedings of the AAAI Conference on Artificial Intelligence, 36(2)*. Online, pp. 2189–2197

The high-level idea of the paper (training an image-based classifier with a combination of texts and images with very few labels) was conceived of by Sahand Sharifzadeh and further developed by Sahand Sharifzadeh and Sina Moayed Baharlou. Sahand Sharifzadeh and Sina Moayed Baharlou were responsible for and carried out all work for the computer vision part of the paper, including the integration of the knowledge extracted from texts. I was responsible for and carried out all work for the NLP part of the paper. I designed and implemented all text-to-graph models and generated the graphs from text that the proposed training method relies on. I designed and conducted the experiments to evaluate the text-to-graph component of the proposed pipeline and all text-to-graph baselines quantitatively (table 2) and qualitatively (table 1). Sahand Sharifzadeh and Sina Moayed Baharlou wrote most of the initial draft of the article and did most of the subsequent corrections. I identified and described relevant related work in the field of knowledge extraction from text and described the model architectures of the text-to-graph models I designed and implemented. I regularly discussed this work with the lead author and also assisted in improving the draft as a whole.

München, den 24. August 2021

Martin Schmitt

# Contents

# Chapter 1

# Introduction

## 1.1 Problem Formulation

### 1.1.1 Motivation

Relational knowledge is considered foundational for higher cognitive processes (Halford et al., 2010). It is therefore not surprising that relations can be found at the core of many prominent concepts in computer science, be it organizing database systems (Codd, 1970), designing computer programs (Booch et al., 2005; Lewis and Loftus, 2009, section 1.6 "Object-oriented programming"), or formalizing knowledge for artificial intelligence (Baader et al., 2007). While, of course, there can be the need to express relationships between any number of entities or objects, the binary relation, i.e., the relation between two entities, is enough for most applications because any $n$-ary relation can be modeled by reifying relation instances and introducing $n$ new binary relations, i.e., one per original argument slot (Hayes and Welty, 2006).

Besides being an important building block of description logics (Borgida, 1996; Horrocks et al., 2006) and ontology languages (Antoniou and van Harmelen, 2004) and the resulting widespread use for creating knowledge bases (KBs) or knowledge graphs (KGs), binary relations are also ubiquitous in linguistics, the formal study of language. Syntactic dependency, for example, is defined as an asymmetric relation between *two* words in a sentence (Tesnière, 1959; Hudson, 1990) and lexicographers make use of many binary relations, such as hypernymy or antonymy, to structure lexical knowledge (Miller, 1992, inter alia). Even the definition of basic word order in linguistic typology hinges on transitive declarative main clauses (Comrie, 1989), such as *Alice likes Bob*, suggesting that they are considered prototypical for a natural language utterance. Arguably, such a sentence expresses a binary relation, expressed by the verb (predicate), between the verb's arguments (subject and object). Given its adequateness for expressing knowledge

both formally (in a KG) and informally (through natural language), the focus of this thesis is thus the computational modeling of binary relations in natural language texts and KGs. As a general term for both a KG fact and a natural language sentence with subject, predicate, and object, we will use *SPO statement*.

Despite their similar (SPO) structure, it is nontrivial to translate a KG fact to a corresponding natural language sentence and back, let alone a collection of facts to a coherent text or vice versa. So an obvious endeavor in the study of formally and informally expressed knowledge is to look for means to convert one to the other. Graph-to-text generation makes formal knowledge bases accessible to non-experts and enhances their interpretability. The opposite direction supports machines in understanding natural language texts, enabling them to assist humans in making sense of a larger number of texts in shorter time.

All knowledge collections are inevitably incomplete, either because things are left implicit in a text, because the manual curation of KGs is very time-intensive, or simply because humanity's knowledge as a whole is still limited in a certain domain. One of the most useful supporting tasks in this context is therefore the discovery of new links, the inference of new knowledge beyond what is expressed literally in a text or KG. This link prediction or knowledge base completion (KBC) task traditionally aims at modeling the probability of arbitrary individual KG facts given all KG facts seen during training (Nickel et al., 2016). Focusing on the semantics of relations, this thesis approaches the knowledge inference task from a different angle. We are interested in all the implications a given SPO statement entails when we abstract away from the concrete subject and object entities. Besides providing a better idea of what kind of semantics is captured by a computational model of relations, such a task also has the advantage of producing human-interpretable entailment rules (see Chapter 2) and of evaluating the influence of lexical semantics in natural language inference (NLI) applications. Only recently, lexical semantics was detected as a blind spot of many statistical NLI models (Glockner et al., 2018). Thus it is worthwhile to study the role of SPO statements in general NLI. Besides its importance for analyzing NLI phenomena and KBC (e.g., Hosseini et al., 2019), entailment detection between SPO statements has also been found useful for automatic question answering (Schoenmackers et al., 2010) and modeling event coreference (Shwartz et al., 2017; Meged et al., 2020).

### 1.1.2 Approach

We choose a data-driven approach to the computational modeling of binary relations. Alleviating a lack of relevant resources, we collect and annotate data to improve the fundamental understanding of the phenomena involved with relational data, both in KGs and text documents. Besides delivering new insights by analyzing the data directly, a data-driven approach also allows for the application of

machine learning methods and thus the creation of automatic software solutions to the tasks of relational inference and KG-text conversion.

Regarding machine learning models, we focus on deep learning (DL) techniques and artificial neural networks (NNs) (Goodfellow et al., 2016). We make this choice because of their recent success in all parts of natural language processing (NLP) and artificial intelligence in general, but also for their flexibility as universal function approximators (Lu et al., 2017) and their suitability for both supervised and unsupervised learning (Goodfellow et al., 2016).

### 1.1.3 Research Questions

Concretely, we aim to answer the following research questions:

(i) *Data:* What kind of data is adequate to train and evaluate a statistical model of relational semantics? How can we collect such data?

(ii) *Models:* How can relational semantics be modeled effectively in a computer system? How can such a model be used to detect entailment or to translate between knowledge graphs and unstructured text?

(iii) *Analysis:* What modeling choices work better than others? What elements of our models and algorithms are decisive for their performance?

(iv) *Improvements:* How can we improve model performance or efficiency? Can we train models with less annotated data or no annotated data at all? Can our insights be used to improve downstream applications?

### 1.1.4 Outline

In this chapter, we define the basic notions used throughout this thesis. We clarify terminology, introduce fundamental work used in subsequent chapters, and show commonly used evaluation methods. In Chapter 2, we describe the collection of a new dataset and benchmark for textual relation inference in context (RIC). Using these data, we evaluate a variety of baseline models and discover key challenges of the task. Chapter 3 investigates the usage of contextualized embeddings provided by a pretrained language model (PLM) for the same task. We examine the influence of different textual patterns surrounding the actual input on the contextualized representations and find a useful inductive bias in handcrafted entailment templates. Revisiting the notion of such patterns, Chapter 4 defines a framework for adding arbitrary vectors to the input, i.e., vectors that do not necessarily correspond to any vocabulary element. We analyze when and how these continuous patterns lead to better task performance.

Beginning the part on KG-text conversion, Chapter 5 discusses the problem of data scarcity as a result of the large variety of KGs and language. We propose a solution based on unsupervised learning and exemplify its worth by creating a new graph-text benchmark for the domain of scene graphs, i.e., KGs describing images, and image captions, i.e., texts describing images. In Chapter 6, we introduce a new way of injecting graph structure in the text generation architecture Transformer (Vaswani et al., 2017) for parameter-efficient, high-performance graph-to-text generation. Chapter 7 investigates the use of pretrained Transformer language models on this task and analyzes how and if they make use of the input graph structure at all. Finally in Chapter 8, we showcase an application for KG extraction from text. We describe a practical text-to-graph model based on our insights from the previous chapters and show how an image analysis system can benefit from structured knowledge that was automatically generated from texts.

# 1.2 Knowledge Representation

## 1.2.1 Knowledge Bases

The term knowledge base (KB) most commonly denotes a description logic ontology, which consists of the following pairwise disjoint sets: $\mathcal{O} = (N_I, N_C, N_R, \mathcal{A})$ where $N_I$ is a set of individual names, $N_C$ a set of concept names, $N_R$ a set of role names, and $\mathcal{A}$ a set of axioms, i.e., statements that must be true in the situation described by the ontology (Baader et al., 2007). Individual names refer to specific entities in the ontology, e.g., $\texttt{Amazon}, \texttt{Bob} \in N_I$. Concepts are unary predicates, also called types or classes, e.g., $\text{COMPANY}(\cdot) \in N_C$, i.e., they refer to (sub-)sets of entities. Roles are binary predicates (i.e., binary relations), e.g., $works\_for(\cdot, \cdot) \in N_R$, i.e., they refer to sets of entity pairs. Finally, the axioms $\mathcal{A}$ express the actual knowledge.

There is no logical difference between different types of axioms but traditionally we distinguish *assertional*, *terminological*, and *relational* axioms. Assertional axioms state facts about individual entities, referred to by their individual names from $N_I$. Besides asserting (in-)equality of two entities, these axioms mainly express facts using the available concept and role names, e.g., $\text{COMPANY}(\texttt{Amazon})$, $works\_for(\texttt{Bob}, \texttt{Amazon})$ etc. Terminological axioms define a taxonomy on concept expressions where such an expression can consist of the basic concepts from $N_C$, the logical connectors *or*, *and*, and *negation*, as well as description logic-specific operators that define concepts w.r.t. the number and optionally the name of relations to other entities and, recursively, the logical properties of these "neighboring" entities. For instance, the description logic formula $\text{UNEMPLOYED} \equiv \forall works\_for.\, \bot$ defines the concept UNEMPLOYED as all entities without any $works\_for$ connections.[1] Besides equivalence ($\equiv$), the most important building block of these taxonomies is concept inclusion, i.e., entailment ($\sqsubseteq$). Finally, relational axioms state knowledge about roles. Besides asserting various properties of relations, such as transitivity, symmetry, or disjointness from other relations, they can be used to define a taxonomy on roles like terminological axioms do for concepts. For example, $works\_for \sqsubseteq pays^-$ encodes the fact that working for an organization entails being paid by it.[2]

## 1.2.2 Knowledge Graphs

A knowledge graph (KG) is commonly defined as a multi-relational graph, in other words a directed edge-labeled graph (Hogan et al., 2021): $G := (V, E, L)$, where

[1] $\bot$ stands for the empty concept.

[2] The superscript $^-$ denotes the inverse role.

$V$ is a set of nodes, $L$ a set of edge labels, and $E \subseteq V \times L \times V$ is a set of edges. This definition is very flexible as it allows for several special cases, e.g., nodes that are also edge labels or isolated nodes without any connections. While the latter can make sense for certain applications (see Koncel-Kedziorski et al., 2019), usually the actual knowledge is represented by the edges and thus nodes only make sense in the presence of connections (relations) to other nodes. So often the set of edges $E$ is considered representative for the whole KG because we can infer $V$ and $L$ under the assumption that they contain all and only those nodes and edge labels that are used in $E$. As such a *triple* consisting of two nodes and an edge label is often called a *fact*, we can also say that a KG is adequately represented by a set of facts.

The terms knowledge base and knowledge graph are commonly used interchangeably and indeed a description logic ontology can be expressed as a graph, i.e., we can express every axiom $a \in \mathcal{A}$ as a KG fact $f \in E$. Although it is possible to encode the whole flexibility of description logic formulae in such a graph (Hogan et al., 2021; Baader et al., 2005), common knowledge graphs like Freebase (Bollacker et al., 2008) or Wikidata (Vrandečić and Krötzsch, 2014) rarely make use of the full spectrum. We will therefore only cover what is necessary for the most common use cases. Given an ontology $\mathcal{O} = (N_I, N_C, N_R, \mathcal{A})$, we define the corresponding KG as follows:

$$G := \left(N_I \cup N_C \cup N_R \cup N_R^- \cup \mathcal{B}_C, \mathcal{F}_\mathcal{A}, N_R \cup N_R^- \cup \mathcal{B}_R\right)$$

where $\mathcal{F}_\mathcal{A}$ is the set of axioms translated to facts and $\mathcal{B}_C, \mathcal{B}_R$ contain a number of built-in concepts and edge labels necessary for complex concept expressions and stating certain logical relationships, such as entailment and equivalence.[3]

As every element of the ontology, i.e., individuals, concepts, and roles, become nodes in the KG, we can make statements about any of them. Most of the logical statements make use of the built-in components but notably the ontology roles also serve as edge labels, which lets us define relations between entities in terms of these roles. So we see at once that assertional axioms can be encoded in a straightforward way (see Fig. 1.1 for examples).

Looking at terminological axioms, we can simply model them as edges between concept nodes representing concept inclusion ($subclass\_of \in \mathcal{B}_R$) where concept equivalence can be represented by two edges. For details on constructing complex concepts in a graph using $\mathcal{B}_C$ and $\mathcal{B}_R$, we refer to the literature (Hogan et al., 2021; Baader et al., 2005; de Bruijn and Heymans, 2010).

Finally, a role taxonomy can be modeled in the same way ($subproperty\_of \in \mathcal{B}_R$), using roles as nodes. Other constraints like being a transitive relation or being disjoint from certain other relations similarly make use of elements from $\mathcal{B}_C, \mathcal{B}_R$.

[3]Cf. the OWL schema vocabulary: `https://www.w3.org/2002/07/owl`, accessed on

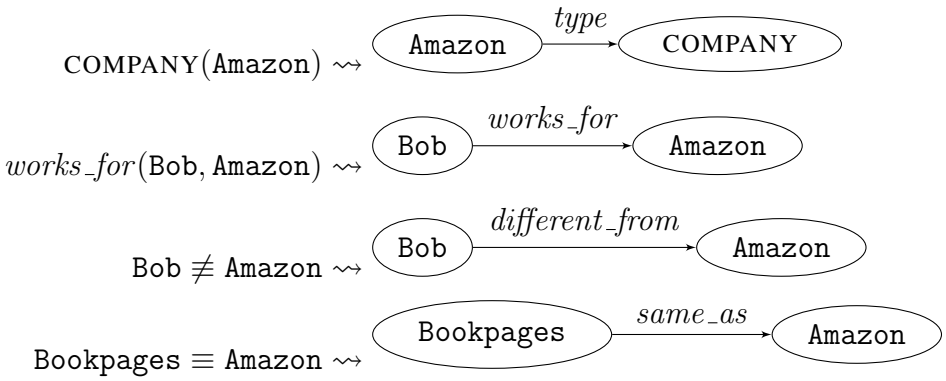



**Figure 1.1** – *Example translations of assertional axioms $a \in \mathcal{A}$ to knowledge graph facts $f \in \mathcal{F}_\mathcal{A}$ with the concept* COMPANY $\in N_C$*, the role* $works\_for \in N_R$*, and the individuals* $\texttt{Amazon}, \texttt{Bob}, \texttt{Bookpages} \in N_I$*. The edge labels* $type$, $different\_from$, $same\_as$ *come from* $\mathcal{B}_R$*.*

Like a description logic ontology, a KG does not distinguish between different kinds of knowledge it encodes. Nevertheless, it is convenient to keep the traditional categories of axioms introduced above to refer to different types of KG facts as well. Inspired by terminology in the description logic literature and for brevity, we will call KG facts coming from assertional axioms *ABox facts*, those coming from terminological axioms *TBox facts*, and those encoding relational axioms *RBox facts*.

Given that our main focus lies on relations, we need a few last definitions for them. As stated above, the interpretation of a relation $R \in N_R$ is the set of individual entity pairs, for which the relation holds. This is called the extension $\mathcal{E}(R)$ of the relation $R$, the set of its arguments (subject and object). In KG terms, we define the extension of $R \in N_R$ as the set of node pairs it links, i.e., $\mathcal{E}(R) = \{ (e_1, e_2) \mid e_1, e_2 \in N_I \wedge (e_1, R, e_2) \in E \}$. So instead of considering semantic links to real-world entities referenced by their names in $N_I$, we keep our definition of relation extension $\mathcal{E}(R) \subseteq N_I \times N_I$ purely on the symbolic level of KG nodes, which is enough for our reasoning purposes.

Often a relation only makes sense if it relates certain types of entities. For example, the semantics of the $capital\_of$ relation dictate that its subject be a city and its object a place that can have a capital, such as a country. The acceptable types of entities for a certain relation are called its *domain* (subject slot) and *range* (object slot). They can be defined in a KG using built-in relations, e.g.,

16th August 2021.

$(capital\_of, domain, \text{CITY}) \in \mathcal{F}$, $domain \in \mathcal{B}_R$. We call a relation with such guarantees w.r.t. its arguments a *typed relation*. Given that the top[4] concept $\top \in \mathcal{B}_C$, which by definition includes all entities in the KG, is always part of a KG, one can argue that every relation is a typed relation because a relation that is sensibly defined for any kind of entity pair would simply have $\top$ as its domain and range.

Finally, while it is common to avoid whitespace in the definition of KB role names and KG edge labels, typical strategies being camel case ($worksFor$) or snake case ($works\_for$), there is no theoretical reason for this restriction. So everything we develop for KG facts can equally be applied to SPO statements in general. So the only condition for any natural language statement to be treated as a KG fact is that the three parts of a triple have to be distinguishable unambiguously (see also *open information extraction*, described in Section 1.5.2).

## 1.3 Methods

### 1.3.1 Deep Learning

Deep learning (DL) is a subfield of machine learning. DL models are commonly called artificial neural networks (NNs). An NN models its input by introducing representations that are expressed in terms of other, simpler representations (Goodfellow et al., 2016). It learns these representations end-to-end without the need for manual feature engineering.

Whenever we say that the parameters of a DL model are trained or learned, we refer to some sort of gradient descent optimization (Cauchy, 1847) of an objective function that represents the performance on the task to be learned. Nearly all of DL is powered by some variant of *stochastic gradient descent* (Goodfellow et al., 2016, Section 5.9). The particular type of optimization algorithm and the objective function are stated for each experiment in each chapter individually. For the purpose of this introduction, we can consider the actual learning a black box and only focus on the modeling part.

#### Feedforward Networks

Fig. 1.2 shows a simple feedforward NN, also called a multilayer perceptron (Rosenblatt, 1962). With $\boldsymbol{x} \in \mathbb{R}^4$, 4 input features are put into the neural model. This could, for example, be a word embedding of dimension 4. A word embedding is a continuous representation of a word in a vector space assigned by a simple

[4] Cf. `owl:Thing` in OWL or `rdfs:Resource` in RDF Schema (`https://www.w3.org/TR/rdf-schema/`, accessed on 16th August 2021).

$$\sigma_0(\sigma_1(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_0)$$

$$\boldsymbol{x} \in \mathbb{R}^4$$
$$\boldsymbol{W}_0 \in \mathbb{R}^{5\times 2}$$
$$\boldsymbol{W}_1 \in \mathbb{R}^{4\times 5}$$

$\sigma_0, \sigma_1$ are non-linear activation functions.

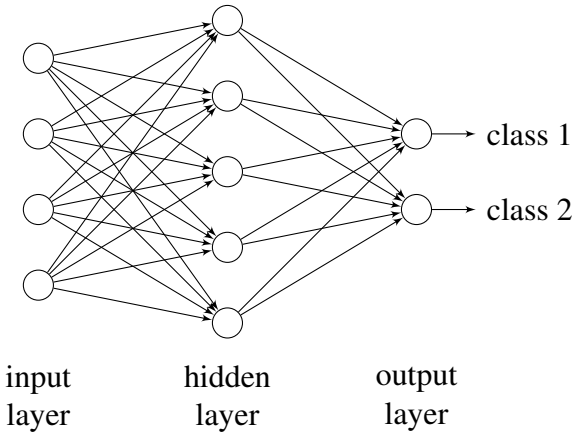



**Figure 1.2** – *Example of a multilayer perceptron for binary classification. The equation on the left describes the same model architecture as the diagram on the right. A popular choice for $\sigma_0$ is the softmax function, which normalizes the output weights such that they sum to 1. In this way, the output can be interpreted as a probability distribution over all (in this case 2) classes.*

lookup table. See Section 1.3.2 for a discussion of pretraining approaches for word embeddings.

The learned matrix $\boldsymbol{W}_1 \in \mathbb{R}^{4\times 5}$ converts this input representation to 5 hidden features, which, in turn, are transformed to 2 output features that represent the 2 classes of some binary classification task. Each matrix multiplication is followed by a non-linear activation function. For $\sigma_1$ this could be, for instance, $ReLU(x) = max(x, 0)$ (with position-wise application of $max$). For the output layer, it makes sense to normalize the output features so that they sum to 1. A popular way of achieving this is by setting $\sigma_0(\boldsymbol{x})_i = \frac{\mathrm{e}^{x_i}}{\sum_{j=1}^{d} \mathrm{e}^{x_j}}$, which is known as softmax function (Goodfellow et al., 2016).

Having shown the most basic version of a feedforward NN, it is straightforward to extend it by adding more layers, i.e., more learned weight matrices. This very simple DL model is rarely used on its own and over the years more complex models have become the standard in NLP. One of these models is of particular interest to us: the Transformer.

### The Transformer Model

The Transformer is a sequence-to-sequence model that was introduced by Vaswani et al. (2017) for machine translation. Fig. 1.3 illustrates the complete architecture.

The Transformer consists of an encoder and a decoder, each of which consists of two types of subnetworks, multi-head attention and feedforward. An encoder

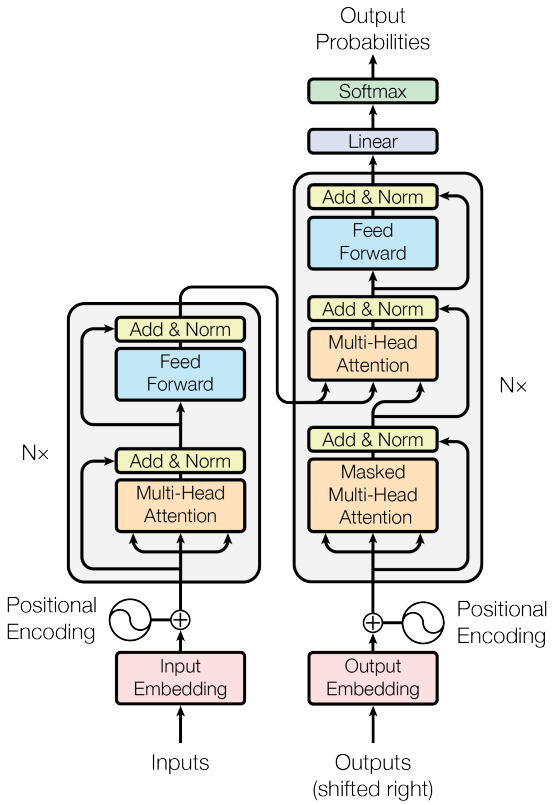



**Figure 1.3** – *The complete Transformer encoder-decoder architecture. Figure by Vaswani et al. (2017).*

block features one of each whereas the decoder block has an additional multi-head attention component that links it to the encoder. Both encoder and decoder blocks are usually repeated and several instances are stacked one on top of the other. In this context, one encoder (resp. decoder) block is called an encoder (resp. decoder) layer. As we saw the feedforward NNs already in the last section, we will now primarily focus on multi-head attention.

Having been introduced as a way of aligning an input with an output sequence in machine translation (Bahdanau et al., 2015), attention is used mainly for contextualization, i.e., sequence encoding, in Transformer. It is used to encode the input sequence (self-attention), the partially generated output sequence (masked self-attention), and to condition the generation on the encoder output (encoder-decoder attention). Each of these use cases is implemented by the same basic mechanism, called scaled dot-product attention (Fig. 1.4, left). It accepts three inputs, the query, key, and value, and outputs a new representation of the same dimensionality as the value. The only difference is that for self-attention all three inputs are the same whereas encoder-decoder attention takes only the contextualized partial generation

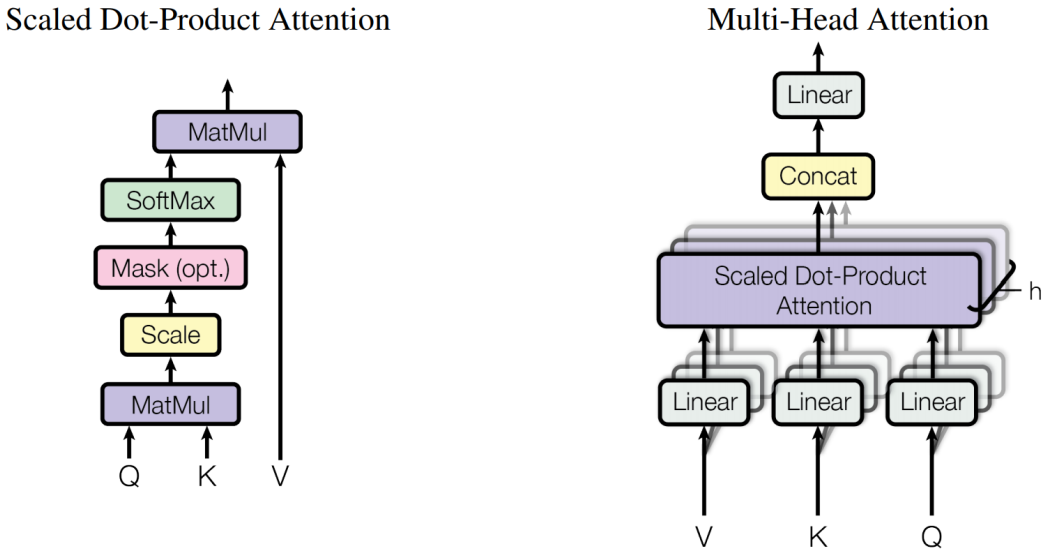



**Figure 1.4** – *Scaled dot-product attention (left) and multi-head attention with several parallel attention layers (right). Figure by Vaswani et al. (2017).*

as query while value and key come from the encoder. One instance of scaled dot-product attention is defined as follows:

$$attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{QK^\intercal}{\sqrt{d_k}})\boldsymbol{V}$$

where $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$, $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$, $\boldsymbol{Q} \in \mathbb{R}^{m \times d_k}$ are matrices built by packing together $n$ (resp. $m$) values, keys, and queries.

Multi-head attention then runs several of these attention layers in parallel and concatenates[5] their results:

$$MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (\Big\|_{i=1}^{h} attention(\boldsymbol{Q}\mathbf{W}_i^{query}, \boldsymbol{K}\mathbf{W}_i^{key}, \boldsymbol{V}\mathbf{W}_i^{value}))\boldsymbol{W}^{out}$$

where each of the $h$ heads learns its own projection matrices for keys, values, and queries, stored in $\mathbf{W}^{key} \in \mathbb{R}^{h \times d \times d_k}$, $\mathbf{W}^{value} \in \mathbb{R}^{h \times d \times d_v}$, $\mathbf{W}^{query} \in \mathbb{R}^{h \times d \times d_k}$ respectively, and $\boldsymbol{W}^{out} \in \mathbb{R}^{hd_v \times d}$ projects the concatenation of all heads' results back to the overall model dimension $d$. Setting $d_k = d_v = d/h$ is a common choice to keep the computational cost similar to that of a single-head attention with full dimensionality. Vaswani et al. (2017) found it beneficial to jointly attend to information from different representation subspaces, which the multi-head approach allows.

[5] We write $\Big\|$ to denote matrix concatenation.

The last important component in Fig. 1.3 that we have not covered yet is the concept of positional encodings. Multihead-attention is agnostic to position information in a sequence (Dufter et al., 2020). Indeed, attention is more suited to operate on a set (Lee et al., 2019). Therefore, the sequential information in a natural language sentence has to be injected in a different manner. There are a lot of different variants on how to do this (see the recent survey by Dufter et al., 2021), the simplest of which is to add an embedding of the absolute token position $p \in \{1, 2, \ldots, n\}$ to the input embeddings. Fig. 1.3 shows this approach.

The work presented in Chapter 6 introduces a new position model to incorporate the graph structure of an input KG into Transformer. In combination with multiple heads, multiple views of the graph can be learned. The Transformer architecture is also important for other parts of this thesis because it is the basis of modern approaches to contextualized word embeddings and transfer learning (see Sections 1.3.2 and 1.3.3).

### 1.3.2 Word Representations

A word in the context of this thesis denotes the smallest unit in the input to an NN. In practice, these units are obtained by some *tokenization* algorithm and might also be smaller than words in the linguistic sense, i.e., so-called subwords such as provided by wordpiece (Schuster and Nakajima, 2012) or BPE (Sennrich et al., 2016). A *word embedding* is a continuous vector representation of such an input unit. As other model parameters, it can simply be learned from the task objective after random initialization. It has been shown, however, that word vectors estimated from co-occurrence information are beneficial for generalization in a number of downstream tasks (Socher et al., 2011; Collobert et al., 2011; Kim, 2014). A word unseen during training will not be represented as a random vector anymore but its vector will be similar to the representation of other words that occur in similar contexts. The *distributional hypothesis* states that these words tend to have similar meanings (Harris, 1954; Firth, 1957).

An example how the co-occurrence of words can yield meaningful vector representations is shown in Fig. 1.5. The figure visualizes two dimensions of a $|\Sigma|$-dimensional space where $\Sigma$ is the whole vocabulary. The matrix $\boldsymbol{C} \in \mathbb{N}^{|\Sigma| \times |\Sigma|}$ stores the number of times any pair of words from $\Sigma$ occur in the same context window.[6] The $i$th row in $\boldsymbol{C}$ corresponds to the vector representation of the $i$th word in $\Sigma$. To simplify notation, $\boldsymbol{C}$ is directly indexed with the respective words in Fig. 1.5.

[6]Using a context window essentially means that two words are considered co-occurring if and only if their distance in a text is lower than a given threshold. Sometimes sentence boundaries are also taken into account (Mikolov et al., 2013). The specific definition of a *context* is generally treated as a hyperparameter of a particular embedding model.

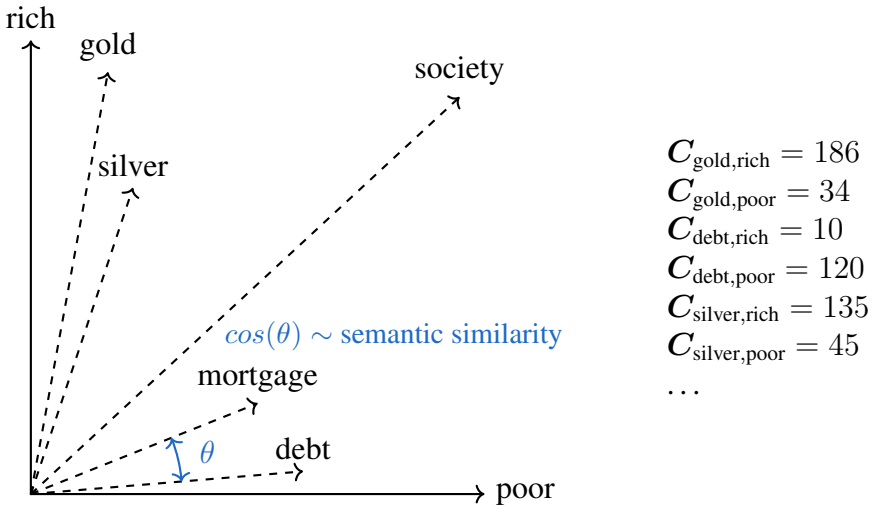



**Figure 1.5** – *Example how co-occurrence information can yield meaningful vector representations. Only the two dimensions corresponding to the words "rich" and "poor" are shown. Figure taken from (Dufter, 2021).*

### Language Modeling

Collecting co-occurrence information or learning from it is deeply connected to a task called *language modeling*. Traditionally, a language model assigns a probability to a sequence of words (Jurafsky and Martin, 2020, chap. 3), i.e., it judges how likely it is for the word sequence to occur in a language. It usually does so by estimating the probability $P(w|h)$ of a word $w$ given some history $h$. Such a model, of course, benefits greatly from co-occurrence information like we saw before to distinguish high-probability words from low-probability words given some context.

We distinguish two types of language models: *Masked* language models (MLMs) and *autoregressive* language models (ALM). An ALM follows the traditional approach. It has access to all previous tokens and predicts the most probable next one, i.e., it estimates $P(w_t|w_1, w_2, \dots w_{t-1})$. In a Transformer decoder (Section 1.3.1), this is achieved by masking the self-attention such that the hidden token representations only depend on previous tokens (Vaswani et al., 2017; Dai et al., 2019). An MLM is sometimes also called an *autoencoding language model*. It sees a corrupted version of the input tokens and tries to reconstruct the original string, i.e., it estimates $P(w_t|\tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_t, \dots, \tilde{w}_n)$ where $\tilde{w}_i$ is most often the original $w_i$ but – with some probability – can also be something else. For instance, the BERT MLM (Devlin et al., 2019) is trained on text with 12% of the tokens replaced by special `[MASK]` symbols and 1.5% replaced by randomly chosen tokens that

are different from the original ones.

### Pretrained Word Type Embeddings

Typical vocabularies can have thousands, if not tens of thousands, of entries, which makes it impractical to use vocabulary-sized word vectors. For they involve a high computational cost and their sparsity can also harm performance. Therefore several techniques have been proposed for computing dense word embeddings of lower dimension. Mikolov et al. (2013)'s word2vec trains a very simple NN to predict all words in an unlabeled text given their respective neighbors in a certain context window.[7] Following the terminology in Fig. 1.2, the resulting word vectors are taken from the input layer of this NN. The word2vec approach is therefore akin to randomly initializing word representations and learning them from the (downstream) task objective. The difference is that the word2vec task does not need labeled data but operates on raw text. It is similar to language modeling in that it estimates the probability of words given context but an important difference is that sequence information is ignored. Levy and Goldberg (2014a) embraced this fact by generalizing word2vec's skip-gram model to processing arbitrary pairs of words instead of a text corpus. In contrast, Ling et al. (2015) tried to improve word2vec by adding support for relative position information between words. This extension called *wang2vec* made the word embeddings more suitable for syntactic tasks. While wang2vec does model sequential information to some extent, it still boils down to modeling pairs of words (with a specific distance), i.e., word-word co-occurrence statistics, instead of modeling a complete sentence as a language model would.

GloVe (Pennington et al., 2014) works similarly in that it learns word vectors from a co-occurrence signal. The difference is that GloVe first aggregates global co-occurrence counts and then distills the embeddings from them. It has been argued that models like word2vec or GloVe are intimately connected to low-rank matrix approximation techniques, such as singular value decomposition (Trefethen and Bau, 1997) because they aim to find a matrix of lower dimensionality that expresses the same co-occurrence information as the raw counts in $\boldsymbol{C}$ (see Levy and Goldberg, 2014b; Levy et al., 2015).

### Pretrained Word Token Embeddings

The aforementioned methods all learn only one vector per word type and this vector remains static after pretraining, i.e., it does not adapt to individual contexts in the downstream task. An alternative to extracting the first layer, the embedding

[7] The difference between word2vec's CBOW and skip-gram model is of limited importance for this high-level characterization.

layer, after pretraining – as word2vec does – is keeping the whole pretrained NN and use some hidden layer representation as input to the downstream model. As higher-level representations can depend on more than a single input word at a time, this has the advantage that each token obtains an embedding based on its individual context. So a word, such as *bank*, can have different embeddings, depending on the appropriate sense in the sentence context (cf. *river bank* vs. *bank* as financial institute). And even unambiguous words can benefit from different embeddings in different contexts for purposes such as coreference resolution (Peters et al., 2018). To illustrate, consider a model trying to distinguish the two animals in the sentence *A small dog is chased by a larger dog.* This type of embedding is called *contextualized* (vs. static) or *word token* (vs. word type) embedding because there is a different embedding for each token based on its context instead of only one static embedding per word type.

Similar to the word type embeddings we have seen before, co-occurrence information is also a useful signal for training models of word token embeddings and one that does not require annotated data. Contextualized embedding models are therefore pretrained on language modeling (see above). The hidden states of ALMs – like ELMo (Peters et al., 2018) or GPT (Radford et al., 2018) – or MLMs – like BERT (Devlin et al., 2019) or BART (Lewis et al., 2020) – have been reported to improve downstream task performance, when used as input representations.

### 1.3.3 Transfer Learning

In general, transfer learning studies how machine learning models can be transferred to data outside of their training distribution (Ruder, 2019). It has become ubiquitous in NLP in the last years due to the success of contextualized word embeddings (see Section 1.3.2).

#### Classification

For classification tasks, we distinguish two scenarios for the use of these embeddings that are computed on the fly by a PLM: (1) The PLM's weights are frozen during training on the downstream task. Here, the PLM only delivers the contextualized embeddings as input representations for a new randomly initialized classification model with arbitrary architecture. (2) The gradient is propagated to the PLM itself and its weights are updated during training. Here, you could say that the PLM is more intimately part of the task-specific model and the new randomly initialized part is only an extension. In any case, the use of a PLM for downstream task training is notably different from the use of static word vectors such as created by word2vec. While word2vec embeddings are only ever used to initialize a specific layer of a new model architecture without considering the

(arguably small) rest of the word2vec NN, the (large) PLM's architecture is kept as is and thus defines a substantial part of the computations of the new model. So it makes sense to say that the use of contextualized word embeddings is a form of transfer learning. The knowledge acquired by the PLM during training on the language modeling task is to be transferred to the task we are actually interested in. Ruder (2019) calls this kind of transfer learning, where two different tasks are learned one after the other, *sequential transfer learning*, a type of *inductive transfer learning*. Empirical results suggest that language modeling works particularly well for boosting performance on various NLP tasks compared to other pretraining tasks (Wang et al., 2019).

Besides the approach of putting a task-specific model extension on top of a PLM and simply fine-tuning this extended model on target task training data, there has also been a recent trend to associate specific words with certain classes and thus transferring the knowledge from the language modeling task more directly to classification by taking the most probable word in context as classification decision. In contrast to what we have discussed so far, this approach does not need a randomly initialized extension to the PLM. For instance, Schick and Schütze (2021a) define a binary sentiment classification task (positive/negative) as prompting a PLM with a template like REVIEW *It was* `[MASK]`. and comparing the probabilities for *great* and *bad* as candidates for the masked word. The instantiated template *Best pizza ever! It was* `[MASK]`. would then result in a higher probability for *great* than for *bad*. If the textual template and the so-called verbalizer, i.e., the association between classes and words, are carefully chosen, zero-shot performance can already be quite high. Further fine-tuning increases performance further.

Although we do not use verbalizers, we show in Chapter 3 that combining textual patterns with the standard fine-tuning approach is already beneficial on its own for RIC. In Chapter 4, we take a look at what happens if the patterns are not bound to elements of the natural language vocabulary, i.e., if their embeddings contain arbitrary vectors, and analyze how to further boost the performance.

### Generation

Although it is highly effective to transfer knowledge from language modeling to classification tasks, such as RIC, the transfer to generative tasks is even more natural – at least for language modeling objectives that include the generation of fluent text. For ALMs, this is always the case. In contrast, MLMs often only predict single words (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020) or small groups of words (Raffel et al., 2019). But they can also be trained to generate the whole corrupted passage (Lewis et al., 2020), which means their objective does include generating fluent text in this case.

For generation, decoder-only autoregressive language models, such as Transformer-

XL (Dai et al., 2019), or encoder-decoder models, such as T5 (Raffel et al., 2019) or BART (Lewis et al., 2020), are simply fine-tuned on the target task without modifying or extending the architecture at all. Their usefulness for graph-to-text generation is investigated in Chapter 7.

## 1.4 Relation Inference in Context

### 1.4.1 Task Definition

Relation inference in context (RIC) is the task of detecting entailment between two relations in a context defined by their arguments. These arguments can be concrete entities (Berant, 2012), a mixture of concepts and individuals (Levy and Dagan, 2016), or only abstract entity types (Nakashole et al., 2012). Following our terminology from Section 1.2.2, we simply define the task over typed relations, i.e., the domain and range of the two relations form the context. This allows for all variants of the task because types can contain as many or few entities as necessary. Consider the following examples with the natural language relation *runs*:

(1) a. `Bezos` runs `Amazon`

b. `iPhone` runs `iOS`

(2) a. PERSON runs COMPANY

b. COMPUTER runs SOFTWARE

(1) shows sentences that might be found in a corpus whereas (2) has corresponding abstractions. It makes sense to attempt reasoning on this more abstract level because the resulting relation taxonomy (RBox facts) will be more general and the focus is more on the semantics of the involved relations than on the truth value of any inferred concrete statement (ABox fact). Consider the following statements that can be inferred from the previous examples:

(3) a. `Bezos` leads `Amazon`

b. `iPhone` executes `iOS`

(4) a. PERSON leads COMPANY

b. COMPUTER executes SOFTWARE

A model deciding whether Example (1a) entails Example (3a) or not risks being distracted by determining the factuality of the ABox fact (3a) instead, which is the focus of the KBC task (Demir et al., 2021). If the model has to decide whether

Example (2a) entails Example (4a), however, it can only rely on the semantics of the two relations in the given context.

Omitting the context completely is often not an option because of its importance for disambiguation. The examples above show that the same relation *runs* can entail either *leads* ((2a)⇒(4a)) or *executes* ((2b)⇒(4b)) in different contexts. Furthermore, it is realistic to assume the availability of domain and range information for most relations in a knowledge graph. Chapter 2 also shows a method for inducing these type signatures if necessary.

RIC is also often called *entailment graph* induction (Berant et al., 2010, 2012; Hosseini et al., 2018, 2019). This name reflects the fact that the ultimate goal in this task is not only to detect entailment between a few pairs of relations but rather the induction of a whole collection of RBox facts, building a relation taxonomy. As we have seen before, such relational axioms can be expressed as a graph. An additional aspect in the induction of the whole entailment graph is the inclusion of global constraints that enforce known properties of entailment, e.g., transitivity. In this work, we only focus on the local phenomena of RIC.

When formulating this task on natural language data, RIC is an instance of the lexical inference task because the lexical semantics of the words expressing the two relations have to be modeled.

### 1.4.2 Modeling Approaches

#### Distributional Semantics

*Distributional semantics* is arguably the most prominent approach to lexical semantics nowadays. We have seen in Section 1.3.2 that co-occurrence information can yield a useful signal for determining word similarity (distributional hypothesis). As (symmetric) similarity is not expected to be enough for RIC, which, in nature, is an asymmetric task, the *distributional inclusion hypothesis* (DIH) goes one step further: If a word $w_1$ entails another word $w_2$, then $w_2$ is expected to occur in all typical contexts of $w_1$ (Weeds et al., 2004; Geffet and Dagan, 2005). Indeed, given an entailment like *dog* ⇒ *animal* and a sentence *A dog lies on the street.*, it makes perfect sense to replace *dog* with the more general term. It is, however, not guaranteed that any text corpus – even a large one – accurately reflects this. It is always possible that some words do not occur together although such a sentence would make sense in theory. It is therefore risky to base a computational model only on counting the overlap of sparse feature vectors.

While most of the DIH methods have been developed for nominal entailment (hypernymy), it is interesting to evaluate their potential for RIC. We present the results of such an evaluation and a comparison to word similarity methods based on dense word vectors in Chapter 2 where we take the entity pairs in a relation's

| Pattern | Example |
|---|---|
| such Ys as X* | such animals as dogs, cats, or hamsters |
| X*, or/and other Ys | dogs, cats, or other animals |
| Y, including/especially X* | all animals, including dogs or cats, ... |

**Table 1.1** – *Example Hearst patterns (Hearst, 1992) for the nominal entailment* $X \Rightarrow Y$ *and example sentences instantiating them.*

| Pattern | Example |
|---|---|
| Xed, i.e., Yed | ... transformed, i.e., integrated ... |
| Yed or at least *X*ed | ... killed or at least wounded ... |
| whether to X or Y | ... whether to open or close ... |

**Table 1.2** – *Example patterns from VerbOcean (Chklovski and Pantel, 2004) detecting three different verb-verb relations, i.e.,* narrow similarity*,* strength*, and* antonymy *(from top to bottom) and example sentences instantiating them.*

extension $\mathcal{E}$ as its distributional features.

### Text Patterns

While there is little research on RIC itself, we again find inspiration for a line of methods in the area of nominal entailment. Textual patterns, such as "*X, such as Y*", were one of the earliest methods to mine hypernym-hyponym pairs (Hearst, 1992). This method is based on the observation that certain phrases naturally tend to link a class to its members in a text (see Table 1.1 for examples).

Although adapting such patterns seems like a natural fit for RIC and there has been work on mining verb-verb relations leveraging text patterns (see Table 1.2 for examples), we have to keep in mind that we will miss a lot of valid examples if we consider all verb pairs invalid that happen to not co-occur in a corpus. This is similar to the problems of sparse vector methods based on the DIH, which we discussed before. In an attempt to overcome these problems, Roller et al. (2018) achieved large improvements for hypernymy detection, i.e., nominal entailment, by using low-rank embeddings of Hearst pattern information. In a similar spirit, we aim to combine patterns for verbal entailment with the generalizing abilities of PLMs in Chapter 3.

# 1.5 Graph-Text Conversion

## 1.5.1 Text Generation from Knowledge Graphs

### Modeling

Converting a KG to a text expressing the same information is akin to machine translation (MT) where a sentence in one language is to be converted to a sentence in another language with the same meaning. A KG could also be seen as an encrypted version of the corresponding text in the target language, a point of view also proposed for MT in the past (Weaver, 1955).

The most common approach in neural MT is to model the conditional probability $P(Y|X)$ of the target sequence $Y$ given the source sequence $X$ in an encoder-decoder architecture (Sutskever et al., 2014), such as the aforementioned Transformer (Section 1.3.1). Consequently, we adapt techniques originally developed for unsupervised MT (Lample et al., 2018) to facilitate unsupervised graph↔text conversion in Chapter 5 and describe a way of injecting graph structure in the Transformer model in Chapter 6.

In contrast to MT, texts and KGs have a large vocabulary overlap. So bilingual (Vulić and Moens, 2016; Artetxe et al., 2017) or multilingual word embeddings (Dufter et al., 2018) are much less crucial for graph→text than a copy mechanism (Gu et al., 2016) or modeling syntax on the sentence-level and content ordering on the document-level (Wiseman et al., 2017; Zhao et al., 2020). KGs – as a type of structured knowledge – tend to be less verbose than texts and therefore have a smaller vocabulary. The KG entities also occur – usually verbatim – in the corresponding texts but the relation names are generally much shorter than the spelled out version in a text. This shows the importance of relational semantics also for graph→text generation.

### Evaluation

Compared to classification, which RIC is an instance of, the evaluation of a language generation model is not as straightforward. While it is simple to compare a predicted class to an annotated ground truth class and count the number of errors, it is unclear how to rank two automatically generated translations w.r.t. their similarity to a reference translation. This is an open question to the extent that even regular shared tasks investigate it (Mathur et al., 2020b).

Although said shared task reports other metrics to correlate better with human judgments and various works (Callison-Burch et al., 2006; Smith et al., 2016) have shown its weaknesses, BLEU (Papineni et al., 2002) remains the de facto standard for evaluating generation models (Mathur et al., 2020a). BLEU is a modified

precision measure of the word n-grams in the generated output $H$ compared to the set $\{R_i\}$ of references, weighted by a brevity penalty BP that punishes overly short generations. The standard BLEU measure compares n-grams with $1 \leq n \leq 4$.

$$\text{BLEU}(H, \{R_i\}) = \text{BP}(H, \{R_i\}) \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4} \log p_n(H, \{R_i\})\right) \tag{1.1}$$

$$p_n(H, \{R_i\}) = \frac{\sum\limits_{g \in T} \min(r_g, \mathit{count}(g, H))}{\sum\limits_{g \in \mathit{ngrams}_n(H)} \mathit{count}(g, H)} \tag{1.2}$$

$$\text{where } T = \mathit{ngrams}_n(H) \cap \bigcup_i \mathit{ngrams}_n(R_i)$$

$$r_g = \max_i \mathit{count}(g, R_i)$$

$$\text{BP}(H, \{R_i\}) = \begin{cases} \exp\left(1 - \min\limits_i \frac{|R_i|}{|H|}\right) & \text{if } \forall i.\ |H| \leq |R_i| \\ 1 & \text{otherwise} \end{cases} \tag{1.3}$$

where $\mathit{ngrams}_n(\cdot)$ extracts all token n-grams of the given length $n$ and $\mathit{count}(g, X)$ counts how often a particular n-gram $g$ occurs in a sequence $X$.

Its complex computation makes BLEU fairly difficult to interpret intuitively – other than the guideline that a higher number means better correspondence to the reference texts. Two other problems also come to mind: (1) BLEU depends on the number of references in that a higher number of references leads to higher scores and (2) BLEU highly depends on the chosen text segmentation (tokenization). The first problem comes from the brevity penalty (Eq. (1.3)) only looking at the shortest reference and the overlap $T$ (Eq. (1.2)) counting an n-gram already correct if it occurs in *any* of the references. The second problem is an artifact of counting and comparing word n-grams up to a maximum length of 4. First, word overlap will not be correctly detected if the references and the generated output are not tokenized in the same deterministic way. Second, the use of shorter (subword) tokens distorts the picture in that $n$-grams vary in the number of "real" words (in the linguistic sense) they capture.

So, when interpreting BLEU scores from different datasets or models, we have to make sure that these conditions are the same. To ensure comparability of different models on the same dataset (i.e., without modifying the number of references between runs), generated outputs and references should be detokenized and fed as plain text to a widely recognized reference implementation, such as SacreBLEU (Post, 2018).

Other evaluation metrics for language generation, such as chrF++ (Popović, 2017) or character-level (Sacre)BLEU, aim to avoid this tokenization issue by comparing strings on the character level. Finally, other metrics aim to overcome the limitations of simple string matching by accounting for the fact that there is often more than one formulation of the same statement. For instance, METEOR (Banerjee and Lavie, 2005) does not require an exact match between words but relies on morphological stemming and a list of synonyms. Several metrics, such as BERTScore (Zhang et al., 2020), BLEURT (Sellam et al., 2020), and MoverScore (Zhao et al., 2019), also make use of pretrained Transformer language models to detect paraphrases. Given the different properties of all these automatic metrics, one should not rely on only one but rather compare several of them to judge language generation models.

The alternative to automatic metrics is to conduct a human evaluation where, typically, either domain experts or arbitrary crowd workers have to judge the output of several models and human references against each other according to different criteria, such as fluency, adequacy, etc. Although this is commonly perceived as the most reliable form of evaluation for language generation, there is also a number of disadvantages, such as high cost and a lack of reproducibility as well as comparability across such studies (Howcroft et al., 2020). The topic of examining and improving the reproducibility of human evaluation experiments has recently gained momentum in the first shared task on this subject to be held in 2021.[8]

### 1.5.2 Knowledge Graph Extraction from Text

Just as graph→text helps people understand the structured knowledge in KGs, text→graph makes information accessible to machines that would otherwise only be available as text, possibly because a non-KG-expert authored it. Given the abundance of information stored in natural language text, automatic text→graph conversion is a big opportunity for automatic information processing. The large variety of potentially relevant information, however, is also a challenge because not all textual relations can be expressed reasonably in any given KG schema. In analogy to the term *database schema* (Imielinski and Lipski, 1982), we use the term *KG schema* to denote the available node and edge labels in a KG and their intended semantics. We distinguish two main approaches to this challenge: (1) Discard information that does not fit into the schema as irrelevant or (2) dynamically extend the KG schema to put as much information from the text into the KG as possible.

The first is the traditional approach of completing an existing KB with information extracted from text. Here, we assume that the KG schema is rich enough to cover all possible knowledge of interest. The classic pipeline approach first

[8] `https://reprogen.github.io/`, accessed on 16th August 2021.

detects entity mentions in the text and links them to their corresponding entities in the KG (Kannan Ravi et al., 2021). Then, performing a task called *relation extraction*, the new KG facts are built by classifying each sentence that mentions at least two entities according to a given list of predefined relations (Nadgeri et al., 2021). There are also attempts at jointly solving these two tasks (Nayak and Ng, 2020; Yamada et al., 2020).

The second approach is called *open information extraction* (OIE) (Niklaus et al., 2018). Again, each sentence is processed separately and both entities and relations have to be detected to form KG facts. The difference is that these fact elements do not stem from the fixed set $V$ of entities and the fixed inventory $L$ of relations in a target KG, but they are identified freely, following syntactic and lexical heuristics (Banko et al., 2007; Fader et al., 2011). OIE can be used both to create a KG entirely from scratch (Mitchell et al., 2015) and to extend existing KGs (Riedel et al., 2013; Shi and Weninger, 2018). In the latter case, it can be a challenge to identify and unify semantically similar relations and entities in order to make full use of the additional knowledge. This research area is known under many names, such as ontology mapping (Choi et al., 2006), KG alignment (Wang et al., 2020), KB unification (Delli Bovi et al., 2015), or, specifically for the output of OIE, KB canonicalization (Vashishth et al., 2018; Wu et al., 2018).

In Chapter 2, we take a hybrid approach to constructing a textual event KG from scratch, the basis for our RIC benchmark named SherLIiC. We use entities linked to Freebase (Bollacker et al., 2008), thus grounding SherLIiC in Freebase, but define the relation between two entities as the lexicalized syntactic dependency path between them (for details see Chapter 2), thus flexibly allowing for an arbitrary number of possible relations. This flexibility is important to account for the large variety of natural language relations.

In Chapter 5, we present yet another approach. By means of unsupervised learning, we build a generative system that converts a text to a corresponding KG (and vice versa) in an end-to-end fashion, i.e., without the explicit steps of sentence boundary detection, entity detection, and relation extraction. Although the model does not choose from a given list of predefined relations and the generation is unconstrained, i.e., in principle, any string can be generated to represent a relation, we still encourage the model to adapt to a target KG schema by showing a collection of unlabeled KGs with the desired schema during unsupervised training. In summary, our model does not use parallel text-KG pairs and is not constrained on the entities or relations it may detect, but it still aims to mimic a given collection of example KGs.

In Chapter 8, the same approach is used with supervised training to provide a textual prior for scene graph classification.

## 1.6 Conclusion

In this introductory chapter, we motivated the research topic of computationally modeling binary relations and justified the two tasks, RIC and graph↔text conversion, as application domains. We gave an overview of important concepts that define the broader scope of this thesis and described relevant methods that are analyzed and used in the next chapters.

### 1.6.1 Contributions

Referring to the research questions outlined in Section 1.1.1, this thesis presents the following contributions:

(i) *Data:* We design and conduct the collection and annotation of new data displaying the phenomena of RIC and contrasting them with distributional similarity. In a large-scale evaluation, we demonstrate the shortcomings of previous data collections for modeling relational semantics (Chapter 2).

(ii) *Models:* We present several new models for RIC and graph↔text conversion and evaluate their effectiveness both quantitatively and qualitatively throughout the thesis.

(iii) *Analysis:* Extensive performance evaluations and ablation studies identify key factors of successful models. Chapter 7, in particular, gives insights into the inner workings of a PLM.

(iv) *Improvements:* We improve the state of the art for our two tasks along several axes: need for training data (Chapter 5), efficiency (Chapter 6), and task performance (Chapters 3 and 4).

### 1.6.2 Future Work

Our suggestions for future work are also structured around our four research questions:

(i) *Data:* The in-domain performance on existing benchmarks we report in Chapters 3 and 4 is already quite high while out-of-domain performance is not. So a challenge for future research will be the collection of more data to capture a larger variety of RIC phenomena during training and evaluation to (a) better assess the quality of RIC models and (b) improve generalization to existing datasets. Automatic annotation of the large collection of unlabeled inference candidates SherLIiC-InfCands described in Chapter 2 and methods

leveraging noisy annotations (e.g., Zhang and Sabuncu, 2018) could be a starting point for this endeavor.

(ii) *Models:* Despite their success in both supervised and unsupervised learning, sequence-to-sequence models do not capture the structure of a graph directly but only via the detour of a serialization (Zhao et al., 2020). This strongly suggests that there is a potential gain for graph↔text conversion if we can unify our work on PLMs (Chapter 7) or unsupervised training (Chapter 5) with a stronger graph-structural bias. There is a recent trend for both of these research directions (Ribeiro et al., 2021; Guo et al., 2020; Ke et al., 2021).

(iii) *Analysis:* Combining PLMs with text patterns or prompts has recently gained enormous popularity (Schick and Schütze, 2021a,b; Zhong et al., 2021; Gao et al., 2021, inter alia). Some studies (e.g., Liu et al., 2021) show that patterns based on artificial tokens instead of real words can lead to equal or even larger performance gains (see also Chapter 4), suggesting that there is still a lot about the interaction of PLMs and context templates that is not well understood. It will be a challenge for modeling relational inference in particular but also for machine learning in general to analyze these models and improve our understanding.

(iv) *Improvements:* Especially in the context of very low-resource graph↔text conversion, the question arises how a small number of labeled samples (e.g., 100) should ideally be used. In Chapter 5, we decide to use them for model selection, i.e., as a development set. In light of the recent success of PLMs in few-shot learning (Schick and Schütze, 2021b, inter alia), however, it might be a good idea to use part of the annotations for fine-tuning or employ a combination of unsupervised and few-shot training. It would be of practical value to investigate these possibilities and determine best practices.

# Chapter 2

# SherLIiC: A Typed Event-Focused Lexical Inference Benchmark for Evaluating Natural Language Inference

# SherLIiC: A Typed Event-Focused Lexical Inference Benchmark for Evaluating Natural Language Inference

**Martin Schmitt** and **Hinrich Schütze**
Center for Information and Language Processing (CIS)
LMU Munich, Germany
martin@cis.lmu.de

## Abstract

We present SherLIiC,[1] a testbed for lexical inference in context (LIiC), consisting of 3985 manually annotated *inference rule candidates* (InfCands), accompanied by (i) ~960k unlabeled InfCands, and (ii) ~190k *typed textual relations* between Freebase entities extracted from the large entity-linked corpus ClueWeb09. Each InfCand consists of one of these relations, expressed as a lemmatized dependency path, and two argument placeholders, each linked to one or more Freebase types. Due to our candidate selection process based on strong distributional evidence, SherLIiC is much harder than existing testbeds because distributional evidence is of little utility in the classification of InfCands. We also show that, due to its construction, many of SherLIiC's correct InfCands are novel and missing from existing rule bases. We evaluate a number of strong baselines on SherLIiC, ranging from semantic vector space models to state of the art neural models of natural language inference (NLI). We show that SherLIiC poses a tough challenge to existing NLI systems.

## 1 Introduction

Lexical inference (LI) can be seen as a focused variant of natural language inference (NLI), also called recognizing textual entailment (Dagan et al., 2013). Recently, Gururangan et al. (2018) showed that annotation artifacts in current NLI testbeds distort our impression of the performance of state of the art systems, giving rise to the need for new evaluation methods for NLI. Glockner et al. (2018) investigated LI as a way of evaluating NLI systems and found that even simple cases are challenging to current systems. In this paper, we release SherLIiC, a testbed specifically designed for evaluating a system's ability to solve the hard problem of modeling lexical entailment in context.

[1] https://github.com/mnschmit/SherLIiC

| | | |
|---|---|---|
| (1) | troponymy | ORGF[A] is granting to EMPL[B]<br>⇒ ORGF[A] is giving to EMPL[B] |
| (2) | synonymy + derivation | ORGF[A] is supporter of ORGF[B]<br>⇒ ORGF[A] is backing ORGF[B] |
| (3) | typical actions | AUTH[A] is president of LOC[B]<br>⇒ AUTH[A] is representing LOC[B] |
| (4) | script knowledge | PER[A] is interviewing AUTH[B]<br>⇒ PER[A] is asking AUTH[B] |
| (5) | common sense knowledge | ORGF[A] claims LOC[B]<br>⇒ ORGF[A] is wanting LOC[B] |

Table 1: Examples of SherLIiC InfCands and NLI challenges they cover. ORGF=organization founder, EMPL=employer, AUTH=book author, LOC=location, POL=politician, PER=person.

Levy and Dagan (2016) identified context-sensitive – as opposed to "context-free" – entailment as an important evaluation criterion and created a dataset for LI in context (LIiC). In their data, WordNet (Miller, 1995; Fellbaum, 2005) synsets serve as context for one side of a binary relation, but the other side is still instantiated with a single concrete expression. We aim to improve this setting in two ways.

First, we type our relations on *both sides*, thus making them more general. Types provide a context that can help in disambiguation and at the same time allow generalization over contexts because arguments of the same type are represented abstractly. An example for the need for disambiguation is the verb "run". "run" entails "lead" in the context of PERSON / COMPANY ("Bezos runs Amazon"). But in the context of COMPUTER / SOFTWARE, "run" entails "execute"/"use" ("my mac runs macOS"). Here, types help find the right interpretation.

Second, *we only consider relations between named entities* (NEs). Inference mining based on non-NE types such as WordNet synsets (e.g., ANIMAL, PLANT LIFE) primarily discovers *facts* like "parrotfish feed on algae". In contrast, the focus



*Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 902–914
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics

on NEs makes it more likely that we will capture *events* like "Walmart closes gap with Amazon" and thus knowledge about event entailment like ["$A$ is closing gap with $B$" $\Rightarrow$ "$B$ is having lead over $A$"] that is substantially different from knowledge about general facts.

In more detail, we create SherLIiC as follows. First, we extract verbal relations between Freebase (Bollacker et al., 2008) entities from the entity-linked web corpus ClueWeb09 (Gabrilovich et al., 2013).[2] We then divide these relations into typable subrelations based on the most frequent Freebase types found in their extensions. We then create a large set of inference rule candidates (InfCands), i.e., premise-hypothesis-pairs of verbally expressed relations. Finally, we use Amazon Mechanical Turk to classify each InfCand in a randomly sampled subset as *entailment* or *non-entailment*.

In summary, our contributions are the following: (1) We create SherLIiC, a new resource for LIiC, consisting of 3985 manually annotated InfCands. Additionally, we provide ~960k unlabeled InfCands (SherLIiC-InfCands), and the typed event graph SherLIiC-TEG, containing ~190k typed textual binary relations between Freebase entities. (2) SherLIiC is harder than existing testbeds because distributional evidence is of limited utility in the classification of InfCands. Thus, SherLIiC is a promising and challenging resource for developing NLI systems that go beyond shallow semantics. (3) Human-interpretable knowledge graph types serve as context for both sides of InfCands. This makes InfCands more general and boosts the number of event-like relations in SherLIiC. (4) SherLIiC is complementary to existing collections of inference rules as evidenced by the low recall these resources achieve (cf. Table 3). (5) We evaluate a large number of baselines on SherLIiC. The best-performing baseline makes use of typing. (6) We demonstrate that existing NLI systems do poorly on SherLIiC.

## 2 Generation of InfCands

This section describes creation (§ 2.1) and typing (§ 2.2) of the typed event graph SherLIiC-TEG and then the generation of SherLIiC-InfCands (§ 2.3).

### 2.1 Relation Extraction

For each sentence $s$ in ClueWeb09 that contains at least two entity mentions, we use MaltParser (Nivre et al., 2007) to generate a dependency graph, where nodes are labeled with their lemmas and edges with dependency types. We take all shortest paths between all combinations of two entities in $s$ and represent them by alternating edge and node labels. As we want to focus on relations that express events, we only keep paths with a nominal subject on one end. We also apply heuristics to filter out erroneous parses. See Appendix A for heuristics and Table 5 for examples of relations.

**Notation.** Let $\mathcal{R}$ denote the set of extracted relations. A relation $R \in \mathcal{R}$ is represented as a set of pairs of Freebase entities (its extension): $R \subseteq \mathcal{E} \times \mathcal{E}$, with $\mathcal{E}$ the set of Freebase entities. Let $\pi_1, \pi_2$ be functions that map a pair to its first or second entry, respectively. By abuse of notation, we also apply them to sets of pairs. Finally, let $\mathcal{T}$ be the set of Freebase types and $\tau\colon \mathcal{E} \to 2^{\mathcal{T}}$ the function that maps an entity to the set of its types.

### 2.2 Typing

We define a *typable subrelation* of $R \in \mathcal{R}$ as a subrelation whose entities in each argument slot share at least one type, i.e., an $S \subseteq R$ such that:

$$\forall i \in \{1,2\} : \exists t \in \mathcal{T} \colon t \in \bigcap_{e \in \pi_i(S)} \tau(e)$$

We compute the set $\mathrm{Type}_{k^2}(R)$ of the (up to) $k^2$ largest typable subrelations of $R$ and use them instead of $R$. First, for each argument slot $i$ of the binary relation $R$, the $k$ types $t^i_j$ (with $1 \le j \le k$) are computed that occur most often in this slot:

$$t^i_j := \arg\max_t \left| \{\, p \in R \mid t \in \tau^i_j(\pi_i(p)) \,\} \right|$$

with

$$\tau^i_1(e) = \tau(e)$$
$$\tau^i_{j+1}(e) = \tau^i_j(e) - \{t^i_j\}$$

Then, for each pair

$$(s,u) \in \{\, (t^1_j, t^2_l) \mid j,l \in \{1,\dots,k\} \,\}$$

of these types, we construct a subrelation

$$R_{s,u} := \{\, (e_1,e_2) \in R \mid s \in \tau(e_1), u \in \tau(e_2) \,\}$$

If $|R_{s,u}| \ge r_{\min}$, $R_{s,u}$ is included in $\mathrm{Type}_{k^2}(R)$. In our experiments, we set $k = r_{\min} = 5$.

[2] http://lemurproject.org/clueweb09

The type signature (tsg) of a typed relation $T$ is defined as the pair of sets of types that is common to first (resp. second) entities in the extension:

$$\mathrm{tsg}(T) = \left( \bigcap_{e \in \pi_1(T)} \tau(e), \bigcap_{e \in \pi_2(T)} \tau(e) \right)$$

**Incomplete type information.** Like all large knowledge bases, Freebase suffers from incompleteness: Many entities have no type. To avoid losing information about relations associated with such entities, we introduce a special type $\top$ and define $\arg\max_t |\emptyset| := \top$. We define the relations $R_{s,\top}$, $R_{\top,u}$ and $R_{\top,\top}$ to have no type restriction on entities in a $\top$ slot. This concerns approximately 17.6% of the relations in SherLIiC-TEG.

### 2.3 Entailment Discovery

Our discovery procedure is based on Sherlock (Schoenmackers et al., 2010). For the InfCand $A \Rightarrow B$ ($A, B \in \mathcal{R}$), we define the relevance score Relv, a metric expressing Sherlock's stat. relevance criterion $P(B \mid A) \gg P(B)$ (cf. Salmon, 1971).

$$\mathrm{Relv}(A, B) := \frac{P(B \mid A)}{P(B)} = \frac{|A \cap B|\,|\mathcal{E} \times \mathcal{E}|}{|A|\,|B|}$$

Our significance score $\sigma(A, B)$ is a scaled version of the significance test lrs used by Sherlock:

$$P(B \mid A)\,\mathrm{lrs}(A, B) = \frac{|A \cap B|\,\mathrm{lrs}(A, B)}{|A|}$$

with $\mathrm{lrs}(A, B)$ (likelihood ratio statistic) defined as

$$2\,|A| \sum_{H \in \{B, \neg B\}} P(H \mid A) \log(\mathrm{Relv}(A, H)).$$

Additionally, we introduce the *entity support ratio*:

$$\mathrm{esr}(A, B) := \frac{\left| \bigcup_{i \in \{1,2\}} \pi_i(A \cap B) \right|}{2\,|A \cap B|}$$

This score measures the diversity of entities in $A \cap B$. We found that many InfCands involve a few frequent entities and so obtain high Relv and $\sigma$ scores even though the relations of the rule are semantically unrelated. esr penalizes such InfCands.

We apply our three scores defined above to all possible pairs of relations $(A, B) \in \mathcal{R} \times \mathcal{R}$ and accept a rule iff all of the following criteria are met:

1. $\forall i \in \{1, 2\} : \pi_i(\mathrm{tsg}(A \Rightarrow B)) \neq \emptyset$
2. $|A \cap B| \geq r_{\min}$
3. $\forall i \in \{1, 2\} : |\pi_i(A \cap B)| \geq r_{\min}$
4. $\mathrm{Relv}(A, B) \geq \vartheta_{relv}$
5. $\sigma(A, B) \geq \vartheta_\sigma$
6. $\mathrm{esr}(A, B) \geq \vartheta_{\mathrm{esr}}$

where $\mathrm{tsg}(A \Rightarrow B)$ is component-wise intersection of $\mathrm{tsg}(A)$ and $\mathrm{tsg}(B)$ and $\vartheta_{relv} = 1000$, $\vartheta_\sigma = 15$, $\vartheta_{\mathrm{esr}} = 0.6$. We found these numbers by randomly sampling InfCands and inspecting their scores. Typing lets us set these thresholds higher, benefitting the quality of SherLIiC-InfCands.

Lastly, we apply Schoenmackers et al. (2010)'s heuristic to only accept the 100 best-scoring premises for each hypothesis. For each hypothesis $B$, we rank all possible premises $A$ by the product of the three scores and filter out cases where $A$ and $B$ only differ in their types.

**Fact**: location[B] is annexing location[A] .
Examples for location[B]: *Russia / USA / Indonesia*
Examples for location[A]: *Cuba / Algeria / Crimea*

☐ fact incomprehensible

**Please answer the following questions:**

| |
|---|
| Is it certain that location[B] is taking control of location[A]? ○ yes ○ no ○ incomprehensible |
| Is it certain that location[B] is taking location[A]? ○ yes ○ no ○ incomprehensible |
| Is it certain that location[B] is bordered by location[A]? ○ yes ○ no ○ incomprehensible |

Figure 1: Annotation Interface on Amazon MTurk

## 3 Crowdsourced Annotation

SherLIiC-InfCands contains ~960k InfCands. We take a random sample of size 5267 and annotate it using Amazon Mechanical Turk (MTurk).

### 3.1 Task Formulation

We are asking our workers the same kind of questions as Levy and Dagan (2016) did. We likewise form batches of sentence pairs to reduce annotation cost. Instead of framing the task as judging the appropriateness of answers, however, we state the premise as a fact and ask workers about its entailed consequences, i.e., we ask for each candidate hypothesis whether it is *certain* that it is also true. Fig. 1 shows the annotation interface schematically.

We use a morphological lexicon (XTAG Research Group, 2001) and form the present tense of a dependency path's predicate. If a sentence is flagged incomprehensible (e.g., due to a parse error), it is excluded from further evaluation.

| Annotated Subset of SherLIiC-InfCands | |
|---|---|
| Validated InfCands | 3985 |
| Balance yes/no | 33% / 67% |
| Pairs with unanimous gold label | 53.0% |
| Pairs with 1 disagreeing annotation | 27.4% |
| Pairs with 2 disagreeing annotations | 19.6% |
| Individual label = gold label | 86.7% |

Table 2: Statistics for crowd-annotated InfCands. The gold label is the majority label.

We put premise and hypothesis in the present (progressive if suitable) based on the intuition that a pair is only to be considered an entailment if the premise makes it necessary for the hypothesis to be true *at the same time*. This condition of simultaneity follows the tradition of datasets such as SNLI (Bowman et al., 2015) – in contrast to more lenient evaluation schemes that consider a rule to be correct if the hypothesis is true at any time before or after the reference time of the premise (cf. Lin and Pantel, 2001; Lewis and Steedman, 2013).

### 3.2 Annotation Quality

We imposed several qualification criteria on crowd workers: number of previous jobs on MTurk, overall acceptance rate and a test that each worker had to pass. Some workers still had frequent low agreement with the majority. However, in most cases we obtained a clear majority annotation. These annotations were then used to automatically detect workers with low *trust*, where *trust* is defined as the ratio of submissions agreeing with the majority answer and a worker's total number of submissions. We excluded workers with a *trust* of less than 0.8 and collected replacement annotations until we had at least five annotations per InfCand.

Table 2 shows that workers agreed unanimously for 53% and that the maximal number of two disagreements only occurs for 19.6%. The high number of times an individual agrees with the gold label suggests that humans can do the task reliably. Interestingly, the number of disagreements is not evenly distributed among the two classes *entailment*/*non-entailment*. If the majority agrees on *entailment*, it is comparatively much more likely that at least one of the workers disagrees (cf. Fig. 2). This suggests that our workers were strict and keen on achieving high precision in their annotations.

## 4 Baselines

We split our annotated data 25:75 into SherLIiC-dev and SherLIiC-test, stratifying on annotated label (*yes*/*no*) and number of disagreements with the majority (0/1/2). If unanimity of annotations marks particularly clear cases, we figure they should be evenly distributed on dev and test.

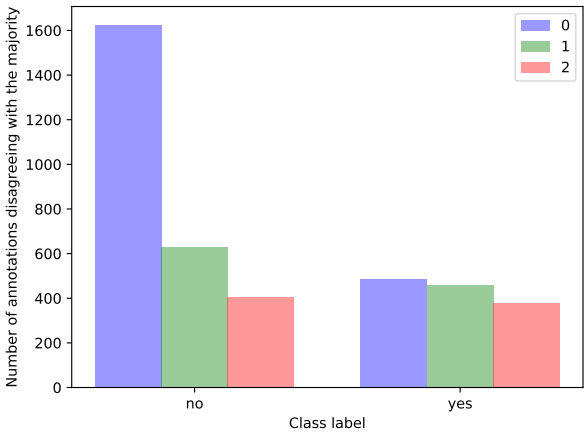



Figure 2: Number of disagreements per class label on all annotations.

To establish the state of the art for SherLIiC, we evaluate a number of baselines. Input to these baselines are either the dependency paths or the sentences that were presented to the crowd workers. Baselines that require a threshold to be used as binary classifiers are tuned on SherLIiC-dev.

**Lemma baseline.** Following Levy and Dagan (2016), this baseline classifies an InfCand as valid if the following holds true for the premise $p$ and hypothesis $h$ after lemmatization: (1) $p$ contains all of $h$'s content words,[3] (2) $p$'s and $h$'s predicates are identical, and (3) the relations' active/passive voice matches their arguments' alignment.

**Rule collection baselines.** `Berant I` (Berant et al., 2011) and `Berant II` (Berant, 2012)[4] are entailment graphs. `PPDB` is the largest collection (XXXL) of PPDB 2.0 (Pavlick et al., 2015).[5] `Patty` is a collection of relational patterns, consisting of ontological types, POS placeholders and words. We use the version extracted from Wikipedia with Freebase types (Nakashole et al., 2012). `Schoenmackers` is the rule collection released by Schoenmackers et al. (2010). `Chirps` is an ever-growing[6] predicate paraphrase database extracted via event coreference in news Tweets (Shwartz et al., 2017b). `All Rules` denotes the union of all of these rule bases. For rules with type (or POS) constraints, we ignore these constraints

[3] We use the stop word list of `nltk` (Loper and Bird, 2002).
[4] We use default threshold 0.
[5] We ignore stop words and punctuation for phrases.
[6] We use the version downloaded on May 28, 2019.

to boost recall. We will see that even with these recall-enhancing measures, the majority of our correct InfCands is not covered by existing rule bases.

**Word2vec baselines.** `word2vec` is based on Mikolov et al. (2013b)'s pre-trained word embeddings of size 300. We average them to obtain a vector representation of relations consisting of multiple words and use cosine similarity to judge a relation pair. `typed_rel_emb` (resp. `untyped_rel_emb`) is obtained by training word2vec skip-gram (Mikolov et al., 2013a) with vector size 300 and otherwise default parameters on a synthetic corpus representing the extensions of typed (resp. untyped) relations. The corpus is obtained by writing out one entity-relation-entity-triple per line, where each entity is prefixed with the argument slot it belongs to. `w2v+typed_rel` (resp. `w2v+untyped_rel`) produces its score by summing the scores of `word2vec` and `typed_rel_emb` (resp. `untyped_rel_emb`).

Some type signatures (tsgs) benefit more from type-informed methods than others. For example, the correct inference [INFLUENCER *is explaining in* WRITTEN_WORK ⇒ INFLUENCER *is writing in* WRITTEN_WORK] is detected by `w2v+typed_rel`, but not by `w2v+untyped_rel`. We therefore combine these two methods by using, for each tsg, the method that works better for that tsg on dev. (For tsgs not occurring in dev, we take the method that works better for the individual types occurring in the tsg. We use untyped embeddings if all else fails.) We refer to this combination as `w2v+tsg_rel_emb`.

**Knowledge graph embedding baselines.** As SherLIiC-TEG has the structure of a knowledge graph (KG), we also evaluate the two KG embedding methods `TransE` (Bordes et al., 2013) and `ComplEx` (Trouillon et al., 2016), as provided by the OpenKE framework (Han et al., 2018).

**Asymmetric baselines.** Entailment models built upon cosine similarity are symmetric whereas entailment is not. Therefore many asymmmetric measures based on the distributional inclusion hypothesis have been proposed (Kotlerman et al., 2010; Santus et al., 2014; Shwartz et al., 2017a; Roller et al., 2018). We consider `WeedsPrec` (Weeds et al., 2004) and `invCL` (Lenci and Benotto, 2012), which have strong empirical results on hypernym detection. We use cooccurrence counts with entity pairs as distributional representation of a relation.

**Supervised NLI models.** As LIiC is a special case of NLI, our dataset can also be used to evaluate the generalization capabilities of supervised models trained on large NLI datasets. We pick `ESIM` (Chen et al., 2017), a state-of-the-art supervised NLI model, trained on MultiNLI (Williams et al., 2018) as provided by the framework Jack the Reader (Weissenborn et al., 2018). Input to `ESIM` are the sentences from the annotation process with placeholders instantiated by entities randomly picked from the example lists that had also been shown to the crowd workers (cf. Fig. 1). As we want to measure `ESIM`'s capacity to detect entailment, we map its prediction of both *neutral* and *contradiction* to our *non-entailment* class.

**Sherlock+ESR.** We also evaluate the candidate scoring method inspired by Schoenmackers et al. (2010) that created the data in the first place. We again combine the three scores described in § 2 by multiplication. The low performance of `Sherlock+ESR` (cf. Table 3) is evidence that the dataset is not strongly biased in its favor and thus is promising as a general evaluation benchmark.

## 5 Experimental Results and Discussion

**Quantitative observations.** Table 3 summarizes the performance of our baselines on predicting the *entailment* class for SherLIiC-dev and -test.

Rule collections (lines 1–6) have recall between 0.119 and 0.308; the recall of their union (line 7) is only 0.483 on dev and 0.493 on test, showing that we found indeed new valid inferences missing from existing rule bases.

The state-of-the-art neural model `ESIM` does not generalize well from MultiNLI (its training set) to LIiC. In fact, it hardly improves on the baseline that always predicts *entailment* (`Always yes`). Our dataset was specifically designed to only contain good InfCands based on distributional features. So it poses a challenge to models that cannot make the fine semantic distinctions necessary for LIiC.

Turning to vector space models (lines 11–24), dense relation representations (lines 12, 13) predict entailment better than sparse models (lines 17–20) although they cannot use asymmetric measures.

KG embeddings (lines 21–24) do not seem at all appropriate for measuring the similarity of relations. First, their performance is very close to `Always yes`. Second, their F1-optimal thresholds are very low – even negative. This suggests that their relation vectors do not contain any helpful

| | Baseline | $\theta^*$ | dev | | | test | | |
|---|---|---|---|---|---|---|---|---|
| | | | P | R | F1 | P | R | F1 |
| 1 | Berant I | – | 0.699 | 0.154 | 0.252 | 0.762 | 0.126 | 0.216 |
| 2 | Berant II | – | 0.800 | 0.181 | 0.296 | 0.774 | 0.186 | 0.300 |
| 3 | PPDB | – | 0.631 | 0.211 | 0.317 | 0.621 | 0.240 | 0.347 |
| 4 | Patty | – | 0.795 | 0.187 | 0.303 | 0.779 | 0.153 | 0.256 |
| 5 | Schoenmackers | – | 0.780 | 0.139 | 0.236 | 0.849 | 0.119 | 0.208 |
| 6 | Chirps | – | 0.370 | 0.308 | 0.336 | 0.341 | 0.295 | 0.316 |
| 7 | All Rules | – | 0.418 | 0.483 | 0.448 | 0.404 | 0.493 | 0.444 |
| 8 | Lemma | – | **0.900** | 0.109 | 0.194 | **0.907** | 0.089 | 0.161 |
| 9 | Always yes | – | 0.332 | **1.000** | 0.499 | 0.333 | **1.000** | 0.499 |
| 10 | ESIM | – | 0.391 | 0.831 | 0.532 | 0.390 | 0.833 | 0.531 |
| 11 | word2vec | 0.321 | 0.556 | 0.625 | 0.589 | 0.520 | 0.606 | 0.559 |
| 12 | typed_rel_emb | 0.864 | 0.561 | 0.568 | 0.565 | 0.532 | 0.486 | 0.508 |
| 13 | untyped_rel_emb | 0.613 | 0.511 | 0.740 | 0.605 | 0.499 | 0.672 | 0.572 |
| 14 | w2v+typed_rel | 1.106 | 0.549 | 0.710 | 0.619 | 0.523 | 0.688 | 0.594 |
| 15 | w2v+untyped_rel | 0.884 | 0.565 | 0.740 | 0.641 | 0.528 | 0.695 | 0.600 |
| 16 | w2v+tsg_rel_emb | 0.884 | 0.566 | 0.776 | **0.655** | 0.518 | 0.727 | **0.605** |
| 17 | WeedsPrec (typed) | 0.073 | 0.335 | 0.994 | 0.501 | 0.333 | 0.988 | 0.498 |
| 18 | WeedsPrec (untyped) | 0.057 | 0.403 | 0.807 | 0.538 | 0.386 | 0.783 | 0.517 |
| 19 | invCL (typed) | 0.000 | 0.332 | **1.000** | 0.499 | 0.333 | **1.000** | 0.499 |
| 20 | invCL (untyped) | 0.148 | 0.362 | 0.876 | 0.512 | 0.357 | 0.863 | 0.505 |
| 21 | TransE (typed) | −0.922 | 0.336 | **1.000** | 0.503 | 0.333 | 0.991 | 0.498 |
| 22 | TransE (untyped) | −0.476 | 0.340 | 0.964 | 0.503 | 0.332 | 0.942 | 0.491 |
| 23 | ComplEx (typed) | −0.033 | 0.339 | 0.955 | 0.500 | 0.337 | 0.949 | 0.497 |
| 24 | ComplEx (untyped) | −0.030 | 0.340 | 0.952 | 0.501 | 0.334 | 0.939 | 0.493 |
| 25 | Sherlock+ESR | $9.460 \cdot 10^5$ | 0.504 | 0.592 | 0.544 | 0.491 | 0.526 | 0.508 |

Table 3: Precision, recall and F1 score on SherLIiC-dev and -test. All baselines run on top of Lemma. Thresholds ($\theta^*$) are F1-optimized on dev. Best result per column is set in bold.

information for the task. These methods were not developed to compare relations; the lack of useful information is still surprising and thus is a promising direction for future work on KG embeddings.

General purpose dense representations (word2vec, line 11) perform comparatively well, showing that, in principle, they cover the information necessary for LIiC. Embeddings trained on our relation extensions SherLIiC-TEG (line 13), however, can already alone achieve better performance than word2vec embeddings alone.

In general, type-informed relation embeddings seem to have a disadvantage compared to unrestricted ones (e.g., cf. lines 12 and 13) – presumably because type-informed baselines have training sets that are smaller (due to filtering) and sparser (since relations are split up according to type signatures). The combination of general word2vec and specialized relation embeddings (lines 14–16), however, consistently brings gains. This indicates that distributional word properties are complementary to the relation extensions our method extracts. So using both sources of information is promising for future research on modeling relational semantics.

w2v+tsg_rel_emb is the best-performing method. It combines typed and untyped relation embeddings as well as general-purpose word2vec embeddings. Even though one cannot rely on typed extensions only, this shows that incorporating type information is beneficial for good performance.

We use w2v+tsg_rel_emb to provide a noisy annotation for SherLIiC-InfCands. This is a useful resource because learning from noisy labels has been well studied (Frénay and Verleysen, 2014; Hendrycks et al., 2018) and is often beneficial.

**Qualitative observations.** Although SherLIiC's creation is based on the same method that was used to create Schoenmackers, SherLIiC is fundamentally different for several reasons: (1) The rule sets are complementary (cf. the low recall of 0.139 and 0.119 in Table 3). (2) The majority of rules in Schoenmackers has more than one premise, leaving only ~13k InfCands in Schoenmackers compared to ~960k in SherLIiC-InfCands that fit the format of NLI. (3) Schoenmackers is filtered more aggressively with the goal of maximizing the number of correct rules. This, however, makes it inadequate as a challenging benchmark be-

| | |
|---|---|
| (1) | PERSON[A] is REGION[B]'s ruler<br>⇒ PERSON[A] is dictator of REGION[B] |
| (2) | LOCATION[A] is fighting with ORGF[B]<br>⇒ LOCATION[A] is allied with ORGF[B] |
| (3) | ORGF[A] is coming into LOCATION[B]<br>⇒ ORGF[A] is remaining in LOCATION[B] |
| (4) | ORGF[A] is seeking from ORGF[B]<br>⇒ ORGF[B] is giving to ORGF[A] |
| (5) | LOCATION[A] is winning war against LOCATION[B]<br>⇒ LOCATION[A] is declaring war on LOCATION[B] |

Table 4: False positives for each of the three best-performing baselines taken from SherLIiC-dev. ORGF=organization founder.

| | | |
|---|---|---|
| nsubj–X–prep–of–obj<br>*A is an **ally** of B* | ⇔ | nsubj–X–poss<br>*A is B's **ally*** |
| nsubj–X–prep–in–obj<br>*A is the **capital** in B* | ⇔ | nsubj–X–poss<br>*A is B's **capital*** |
| nsubjpass–X–prep–by–obj<br>*A is **followed** by B* | ⇔ | obj–X–nsubj<br>*B **follows** A* |
| nsubj–one–prep–of–obj–X–obj<br>*A is one of the **countries in** B* | ⇔ | nsubj–X–obj<br>*A is a **country in** B* |
| nsubj–capital–conj–X–obj<br>*A is the capital and biggest **city in** B* | ⇒ | nsubj–X–obj<br>*A is a **city in** B* |
| nsubj–Xer–prep–of–obj<br>*A is a **teach**er of B* | ⇔ | nsubj–X–obj<br>A **teaches** B |
| nsubj–co-Xer–prep–of–obj<br>*A is a co-**found**er of B* | ⇒ | nsubj–X–obj<br>*A **founds** B* |
| nsubj–reX–obj<br>*A re**writes** B* | ⇒ | nsubj–X–obj<br>*A **writes** B* |
| nsubj–overX–obj<br>*A over**takes** B* | ⇒ | nsubj–X–obj<br>*A **takes** B* |
| nsubj–agree–xcomp–X–obj<br>*A agrees to **buy** B* | ⇒ | nsubj–X–obj<br>*A **buys** B* |
| nsubjpass–force–xcomp–X–obj<br>*A is forced to **leave** B* | ⇒ | nsubj–X–obj<br>*A **leaves** B* |
| nsubjpass–elect–xcomp–X–obj<br>*A is elected to be **governor** of B* | ⇔ | nsubj–X–obj<br>*A is **governor** of B* |
| nsubj–go–xcomp–X–obj<br>*A is going to **beat** B* | ⇒ | nsubj–X–obj<br>*A **beats** B* |
| nsubj–try–xcomp–X–obj<br>*A tries to **compete** with B* | ⇒ | nsubj–X–obj<br>*A **competes** with B* |
| nsubj–decide–xcomp–X–obj<br>*A decides to **move to** B* | ⇒ | nsubj–X–obj<br>*A **moves to** B* |
| nsubjpass–expect–xcomp–X–obj<br>*A is expected to **visit** B* | ⇒ | nsubj–X–obj<br>*A **visits** B* |

Table 5: Most frequent meta rules (top), character level meta rules (middle), and implicative verb meta rules (bottom). Bold: Words corresponding to X.

cause the performance of `Always yes` would be close to 100%. (4) SherLIiC is focused on events. When linking the relations from SherLIiC-TEG back to their surface forms in the corpus, 80% of them occur at least once in the progressive, which suggests that the large majority of our relations indeed represent events.

Taking a closer look at SherLIiC, we see that the data require a large variety of lexical knowledge even though their creation has been entirely automatic. Table 1 shows five positively labeled examples from SherLIiC-dev, each highlighting a different challenge for statistical models that is crucial for NLI. (1) is an instance of *troponymy*: "granting" is a manner or kind of "giving". This is the verbal equivalent to nominal hyponymy. (2) combines synonymy ("support" ⇔ "back") with morph. derivation. (3) can only be classified correctly if one knows that it is one of the typical actions of a president to represent their country. (4) requires knowledge about the typical course of events when interviewing someone. A typical interview involves asking questions. (5) can only be detected with common sense knowledge that goes even beyond that: you generally only claim something if you want it.

An error analysis of the three best-performing baselines (lines 14–16 in Table 3) reveals that none of them was able to detect the five correct InfCands from Table 1. Explicit modeling of one of the phenomena described above seems a promising direction for future research to improve recall. Table 4 shows five cases where InfCands were incorrectly labeled as *entailment*. (1) shows the importance of modeling directionality: every "dictator" is a "ruler" but not vice versa. (2) shows a well-known problem in representation learning from cooccurrence: antonyms tend to be close in the embedding space (Mohammad et al., 2008; Mrkšić et al., 2016). The other examples show other types of correlation that models relying entirely on distributional information will fall for: the outcome of events like "coming into a country" or "seeking something from someone" are in general uncertain although possible outcomes like "remaining in said country" (3) or "being given the object of desire" (4) will be highly correlated with them. Finally, better models will also take into account the simultaneity constraint: "winning a war" and "declaring a war" (5) rarely happen at the same time.

## 6 Meta Rules and Implicative Verbs

In addition to the annotated data, we also make available all ~960k SherLIiC-InfCands found by our unsupervised algorithm. SherLIiC-InfCands's distribution is similar enough to our labeled dataset

to be useful for domain adaptation, representation learning and other techniques when working on LIiC. It can also be investigated on its own in a purely unsupervised fashion as we will show now.

We can find easily interpretable patterns by looking for cases where premise and hypothesis of an InfCand have common parts. By masking these parts (X), we can abstract away from concrete instances and interesting meta rules emerge (Table 5). The most common patterns represent reasonable equivalent formulations, e.g., active/passive voice or "be $Y$'s $X \Leftrightarrow$ be $X$ of $Y$" (the *in*-variant coming from a lot of location-typed rule instances). The fifth most frequent pattern could still be formulated in an even more abstract way but shows already that the general principle of a conjunction $Y \wedge X$ implying one of its components $X$ can be learned.

If we search for meta rules whose X is part of a lemma (rather than a longer dependency path), we discover cases of derivational morphology such as agent nouns (e.g., *ruler*, *leader*) and sense preserving verb prefixes (e.g., *re-write*, *over-react*).

Finally, we observe several implicative verbs (verbs that entail their complement clauses) in their typical pattern $V$ *to* $X \Rightarrow X$. A lot of these verbs are not traditional implicatives, but are called *de facto implicatives* by Pavlick and Callison-Burch (2016) – who argue for the importance of data-driven approaches to detecting de facto implicatives. The meta rule discovery method just described is such a data-driven approach.

## 7 Related Work

**NLI challenge datasets.** A lot of work exists that aims at uncovering weaknesses in state-of-the-art NLI models. Several approaches are based on modifications of popular datasets, such as SNLI or MultiNLI. These modifications range from simple rule-based transformations (Naik et al., 2018) to rewritings generated by genetic algorithms (Alzantot et al., 2018) or adversarial neural networks (Zhao et al., 2018). Lalor et al. (2016) constructed an NLI test set by judging the difficulty of the sentence pairs in a small SNLI subset based on crowd-sourced human responses via Item Response Theory. These works are related as they, too, challenge existing NLI models with new data but orthogonal to ours as their goal is not to measure a model's knowledge about lexical inference in context.

Glockner et al. (2018) modified SNLI by replacing one word from a given sentence by a synonym, (co-)hyponym, hypernym or antonym to build a test set that requires NLI systems to use lexical knowledge. They rely on WordNet's lexical taxonomy. This, however, is difficult for verbs because their semantics depends more on context. Finally, Glockner et al. (2018)'s dataset has a strong bias for *contradiction* whereas our dataset is specifically designed to contain cases of *entailment*.

Our work is more closely related to the dataset by Levy and Dagan (2016), who frame relation entailment as the task of judging the appropriateness of candidate answers. Their hypothesis is that an answer is only appropriate if it entails the predicate of the question. This is often but by no means always true; certain questions imply additional information. Consider: "Which country annexed country[$B$]?" The answer candidate "country[$A$] administers country[$B$]" might be considered valid, given that it is unlikely that one country annexes $B$ and another country administers it. The inference *administer* $\Rightarrow$ *annex*, however, does not hold. Because of these difficulties, we follow the more traditional approach (Zeichner et al., 2012) of asking about consequences of a given fact (the premise).

**Relation extraction.** Some works (Schoenmackers et al., 2010; Berant, 2012; Zeichner et al., 2012) rely on the output from open information extraction systems (Banko et al., 2007; Fader et al., 2011). A more flexible approach is to represent relations as lexicalized paths in dependency graphs (Lin and Pantel, 2001; Szpektor et al., 2004), sometimes with semantic postprocessing (Shwartz et al., 2017b) and/or retransforming into textual patterns (Nakashole et al., 2012). We, too, choose the latter.

**Relation typing.** Typing relations has become standard in inference mining because of its usefulness for sense disambiguation (Schoenmackers et al., 2010; Nakashole et al., 2012; Yao et al., 2012; Lewis and Steedman, 2013). Still some resources only provide types for one argument slot of their binary relations (Levy and Dagan, 2016) or no types at all (Zeichner et al., 2012; Berant, 2012; Shwartz et al., 2017b). Our InfCands are typed in both argument slots, which both facilitates disambiguation and makes them more general.

Some works (Yao et al., 2012; Lewis and Steedman, 2013) learn distributions over latent type signatures for their relations via topic modeling. A large disadvantage of latent types is their lack of intuitive interpretability. By design, our KG types are meaningful and human-interpretable.

Schoenmackers et al. (2010) type common nouns based on cooccurrence with class nouns identified by Hearst patterns (Hearst, 1992) and later try to filter out unreasonable typings by using frequency thresholds and PMI. As KG entities are manually labeled with their correct types, we do not need this kind of heuristics. Furthermore, in contrast to this ad-hoc type system, KG types are the result of a KG design process. Notably, Freebase types function as interfaces, i.e., permit type-specific properties to be added, and are thus inherently motivated by relations between entities.

Lexical ontologies, such as WordNet (as used by Levy and Dagan, 2016) likewise lack this connection between relations and types. Moreover, relations between real-world entities are more often events than relations between common nouns. Thus, in contrast to existing resources that do not restrict relations to KG entities, SherLIiC contains more event-like relations.

Nakashole et al. (2012) also use KG types as context for their textual patterns. They simply create a new relation for each possible type combination for each entity occurring with a pattern and each possible type of this entity. It is unclear how the combinatorial explosion and the resulting sparsity affects pattern quality. Our approach of successively splitting a typewise heterogenous relation into its $k$ largest homogenous subrelations aims at finding only the most typical types for an action and our definition of type signature as intersection of all common types avoids unnecessary redundancy.

**Entailment candidate collection.** Distributional features are a common choice for paraphrase detection and relation clustering (Lin and Pantel, 2001; Szpektor et al., 2004; Sekine, 2005; Yao et al., 2012; Lewis and Steedman, 2013).

The two most important alternatives are bilingual pivoting (Ganitkevitch et al., 2013) – which identifies identically translated phrases in bilingual corpora – and event coreference in the news (Xu et al., 2014; Zhang et al., 2015; Shwartz et al., 2017b) – which relies on lexical variability in two articles or headlines referring to the same event. We specifically focus on distributional information for our InfCand collection because current models of lexical semantics are also mainly based on that (e.g., Grave et al., 2017). Our goal is not to build a resource free of typical mistakes made by distributional approaches but to provide a benchmark to study the progress on overcoming them (cf. § 5).

Another difference to aforementioned works is that we explicitly model unidirectional entailment as opposed to bidirectional synonymy (cf. Table 4, (1)). Here one can distinguish a learning-based approach (Berant, 2012), where an SVM classifier with various features is trained on lexical ontologies like WordNet, followed by the application of global transitivity constrains to enhance consistency, and probabilistic models of noisy set inclusion in the tradition of the distributional inclusion hypothesis (Schoenmackers et al., 2010; Nakashole et al., 2012). We adapt Sherlock, an instance of the latter, for its simplicity and effectiveness.

## 8 Conclusion

We presented SherLIiC, a new challenging testbed for LIiC and NLI, based on typed textual relations between named entities (NEs) from a KG. The restriction to NEs (as opposed to common nouns) allowed us to harness more event-like relations than previous similar collections as these naturally occur more often with NEs. The distributional similarity of both positive and negative examples makes SherLIiC a promising benchmark to track future NLI models' ability to go beyond shallow semantics relying primarily on distributional evidence. We showed that existing rule bases are complementary to SherLIiC and that current semantic vector space models as well as SOTA neural NLI models cannot achieve at the same time high precision and high recall on SherLIiC. Although SherLIiC's creation is entirely data-driven, it shows a large variety of linguistic challenges for NLI, ranging from lexical relations like troponymy, synonymy or morph. derivation to typical actions and common sense knowledge (cf. Table 1). The large unlabeled resources, SherLIiC-InfCands and SherLIiC-TEG, are potentially useful for further linguistic analysis (as we showed in § 6), as well as for data-driven models of lexical semantics, e.g., techniques such as representation learning and domain adaptation. We hope that SherLIiC will foster better modeling of lexical inference in context as well as progress in NLI in general.

## Acknowledgments

We gratefully acknowledge a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes). This work was supported by the BMBF as part of the project MLWin (01IS18050).

## References


Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Procs. of IJCAI*, volume 7, pages 2670–2676.

Jonathan Berant. 2012. *Global Learning of Textual Entailment Graphs*. Ph.D. thesis, The Blavatnik School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 610–619, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing textual entailment: Models and applications*. Morgan & Claypool Publishers.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Christiane Fellbaum. 2005. Wordnet and wordnets. In Keith Brown et al., editor, *Encyclopedia of Language and Linguistics*, second edition, pages 665–670. Elsevier, Oxford.

Benoît Frénay and Michel Verleysen. 2014. Classification in the Presence of Label Noise: A Survey. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 25(5):845–869.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0).

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of tricks for efficient text classification. In *EACL (2)*, pages 427–431. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. 2018. Using trusted data to train deep networks on labels corrupted by severe noise.

In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10477–10486. Curran Associates, Inc.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. 2010. Directional distributional similarity for lexical inference. *Nat. Lang. Eng.*, 16(4):359–389.

John Lalor, Hao Wu, and hong yu. 2016. Building an evaluation scale using item response theory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 648–657, Austin, Texas. Association for Computational Linguistics.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 75–79, Montréal, Canada. Association for Computational Linguistics.

Omer Levy and Ido Dagan. 2016. Annotating relation inference in context via question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Mike Lewis and Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Dekang Lin and Patrick Pantel. 2001. DIRT: Discovery of Inference Rules from Text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 323–328, New York, NY, USA. ACM Press.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 982–991, Honolulu, Hawaii. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145, Jeju Island, Korea. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Ellie Pavlick and Chris Callison-Burch. 2016. Tense manages to predict implicative behavior in verbs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2225–2229, Austin, Texas. Association for Computational Linguistics.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China. Association for Computational Linguistics.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association*

*for Computational Linguistics (Volume 2: Short Papers)*, pages 358–363, Melbourne, Australia. Association for Computational Linguistics.

Wesley C Salmon. 1971. *Statistical explanation and statistical relevance*, volume 69. University of Pittsburgh Pre.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 38–42, Gothenburg, Sweden. Association for Computational Linguistics.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: 2010*, pages 1088–1098.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between NE pairs. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017a. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 65–75, Valencia, Spain. Association for Computational Linguistics.

Vered Shwartz, Gabriel Stanovsky, and Ido Dagan. 2017b. Acquiring predicate paraphrases from news tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 155–160, Vancouver, Canada. Association for Computational Linguistics.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP 2004*, pages 41–48, Barcelona, Spain. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2071–2080.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of Coling 2004*, pages 1015–1021, Geneva, Switzerland. COLING.

Dirk Weissenborn, Pasquale Minervini, Tim Dettmers, Isabelle Augenstein, Johannes Welbl, Tim Rocktäschel, Matko Bošnjak, Jeff Mitchell, Thomas Demeester, Pontus Stenetorp, and Sebastian Riedel. 2018. Jack the Reader – A Machine Reading Framework. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) System Demonstrations*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

XTAG Research Group. 2001. A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 712–720, Stroudsburg, PA, USA. Association for Computational Linguistics.

Naomi Zeichner, Jonathan Berant, and Ido Dagan. 2012. Crowdsourcing inference-rule evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 156–160, Jeju Island, Korea. Association for Computational Linguistics.

Congle Zhang, Stephen Soderland, and Daniel S. Weld. 2015. Exploiting parallel news streams for unsupervised event extraction. *Transactions of the Association for Computational Linguistics*, 3:117–129.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations*.

## A Relation Filter Heuristics

In order to be kept as a relation, a dependency path must fulfill all of the following criteria:

1. It starts or ends with `nsubj` or `nsubjpass`.
2. It starts or ends with one of the following labels: `nsubj`, `nsubjpass`, `iobj`, `dobj`, `pobj`, `appos`, `poss`, `rcmod`, `infmod`, `partmod`.
3. It is not longer than 7 words and 8 dependency labels.

4. At least one of the presumable lemmas contains at least 3 letters.

5. It does not have the same dependency label at both ends.

6. It does not contain any of the following labels: `parataxis`, `pcomp`, `csubj`, `advcl`, `ccomp`.

7. It does not contain immediate repetitions of words or dependency labels.

# Chapter 3

# Language Models for Lexical Inference in Context

# Language Models for Lexical Inference in Context

**Martin Schmitt**[1] and **Hinrich Schütze**[2]
Center for Information and Language Processing (CIS)
LMU Munich, Germany
[1]martin@cis.lmu.de [2]inquiries@cislmu.org

## Abstract

Lexical inference in context (LIiC) is the task of recognizing textual entailment between two very similar sentences, i.e., sentences that only differ in one expression. It can therefore be seen as a variant of the natural language inference task that is focused on lexical semantics. We formulate and evaluate the first approaches based on pretrained language models (LMs) for this task: (i) a few-shot NLI classifier, (ii) a relation induction approach based on handcrafted patterns expressing the semantics of lexical inference, and (iii) a variant of (ii) with patterns that were automatically extracted from a corpus. All our approaches outperform the previous state of the art, showing the potential of pretrained LMs for LIiC. In an extensive analysis, we investigate factors of success and failure of our three approaches.[1]

## 1 Introduction

Lexical inference (LI) denotes the task of deciding whether or not an entailment relation holds between two lexical items. It is therefore related to the detection of other lexical relations like hyponymy between nouns (Hearst, 1992), e.g., *dog* ⇒ *animal*, or troponymy between verbs (Fellbaum and Miller, 1990), e.g., *to traipse* ⇒ *to walk*. Lexical inference in context (LIiC) adds the problem of disambiguating the pair of lexical items in a given context before reasoning about the inference question. This type of LI is particularly interesting for entailments between verbs and verbal expressions because their meaning – and therefore their implications – can drastically change with different arguments. Consider, e.g., *run* ⇒ *lead* in a PERSON / COMPANY context ("Bezos runs Amazon") vs. *run* ⇒ *execute* in a COMPUTER / SOFTWARE context ("My mac runs macOS"). LIiC is thus also closely related to the task of natural language inference (NLI) – also called recognizing textual entailment (Dagan et al., 2013) – and can be seen as a focused variant of it. Besides the important use case of evaluating NLI systems, this kind of predicate entailment has also been shown useful for question answering (Schoenmackers et al., 2010), event coreference (Shwartz et al., 2017; Meged et al., 2020), and link prediction in knowledge graphs (Hosseini et al., 2019).

Despite its NLI nature, previous systems for LIiC have primarily been models of lexical similarity (Levy and Dagan, 2016) or models based on verb argument inclusion (Hosseini et al., 2019). The reason is probably that supervised NLI models need large amounts of training data, which is unavailable for LIiC, and that systems trained on available large-scale NLI benchmarks (e.g., Williams et al., 2018) have been reported to insufficiently cover lexical phenomena (Glockner et al., 2018; Schmitt and Schütze, 2019).

Recently, transfer learning has become ubiquitous in NLP; Transformer (Vaswani et al., 2017) language models (LMs) pretrained on large amounts of textual data (Devlin et al., 2019a; Liu et al., 2019) form the basis of a lot of current state-of-the-art models. Besides zero- and few-shot capabilities (Radford et al., 2019; Brown et al., 2020), pretrained LMs have also been found to acquire factual and relational knowledge during pretraining (Petroni et al., 2019; Bouraoui et al., 2020). The entailment relation certainly stands out among previously explored semantic relations – such as the relation between a country and its capital – because it is very rarely stated explicitly and often involves reasoning about both the meaning of verbs and additional knowledge (Schmitt and Schütze, 2019). It is unclear whether implicit clues during pretraining are enough to learn about LIiC and what the best way is to harness any such implicit knowledge.

Regarding these questions, we make the follow-

[1]Our code is publicly available: https://github.com/mnschmit/lm-lexical-inference



*Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1267–1280
April 19 - 23, 2021. ©2021 Association for Computational Linguistics

ing contributions: (1) This work is the first to explore the use of pretrained LMs for the LIiC task. (2) We formulate three approaches and evaluate them using the publicly available pretrained RoBERTa LM (Liu et al., 2019; Wolf et al., 2019): (i) a few-shot NLI classifier, (ii) a relation induction approach based on handcrafted patterns expressing the semantics of lexical inference, and (iii) a variant of (ii) with patterns that were automatically extracted from a corpus. (3) We introduce the concept of antipatterns, patterns that express non-entailment, and evaluate their usefulness for LIiC. (4) In our experiments on two established LIiC benchmarks, Levy/Holt's dataset (Levy and Dagan, 2016; Holt, 2018) and SherLIiC (Schmitt and Schütze, 2019), all our approaches consistently outperform previous work, thus setting a new state of the art for LIiC. (5) In contrast to previous work on relation induction (Bouraoui et al., 2020), automatically retrieved patterns do not outperform handcrafted ones for LIiC. A qualitative analysis of patterns and errors identifies possible reasons for this finding.

## 2 Related Work

**Lexical inference.** There has been a lot of work on lexical inference for nouns, notably hypernymy detection, resulting in a variety of benchmarks (Kotlerman et al., 2010; Kiela et al., 2015) and methods (Shwartz et al., 2015; Vulić and Mrkšić, 2018). Although there has been work on predicate entailment before (Lin and Pantel, 2001; Lewis and Steedman, 2013), Levy and Dagan (2016) were the first to create a general benchmark for evaluating entailment between verbs. In their evaluation, neither resource-based approaches (Pavlick et al., 2015; Berant et al., 2011) nor vector space models (Levy and Goldberg, 2014) achieved satisfying results. Holt (2018) later published a re-annotated version, which was readily adopted by later work. Hosseini et al. (2018) put global constraints on top of directed local similarity scores (Weeds and Weir, 2003; Lin, 1998; Szpektor and Dagan, 2008) based on distributional features of the predicates. Hosseini et al. (2019) replaced these scores by transition probabilities in a bipartite graph where edge weights are computed by a link prediction model.

When Schmitt and Schütze (2019) created the SherLIiC benchmark, they also mainly focused on resource- and vector-based models for evaluation. Their best model combines general-purpose word2vec representations (Mikolov et al., 2013) with a vector representation of the arguments that co-occur with a predicate.

All these works (i) base the probability of entailment validity on the similarity of the verbs and (ii) compute this similarity via (expected) co-occurrence of verbs and their arguments. Our work differs in that our models solely reason about the sentence surface in an end-to-end NLI task without access to previously observed argument pairs. This is possible because our models have learned about these surface forms during pretraining.

**Patterns and entailment.** Pattern-based approaches have long been known for hypernymy detection (Hearst, 1992). Recent work combined them with vector space models (Mirkin et al., 2006; Roller and Erk, 2016; Roller et al., 2018). While there are effective patterns, such as *X is a Y*, that are indicative for entailment between nouns, there is little work on comparable patterns for verbs. Schwartz et al. (2015) mine symmetric patterns for lexical similarity and achieve good results for verbs. Entailment, however, is not symmetric.

Chklovski and Pantel (2004) handcrafted 35 patterns to distinguish 6 semantic relations for pairs of distributionally similar verbs. Some of their classes like strength (*taint :: poison*) or antonymy (*ban :: allow*) can be indicators of entailment and non-entailment but are, in general, much more narrowly defined than the patterns we use in our approach. Another difference to our work is that verb pairs are scored based on co-occurrence counts on the web, while we employ an LM, which does not depend on a valid entailment pair actually appearing together in a document.

**Patterns and language models.** Amrami and Goldberg (2018) were the first to manipulate LM predictions with a simple pattern to enhance the quality of substitute words in a given context for word sense induction. Petroni et al. (2019) found that large pretrained LMs can be queried for factual knowledge, when presented with appropriate pattern-generated cloze-style sentences. This zero-shot factual knowledge has later been shown to be quite fragile (Kassner and Schütze, 2020). So we rather focus on approaches that fine-tune an LM on at least a few samples. Forbes et al. (2019) train a binary classifier on top of a fine-tuned BERT (Devlin et al., 2019a) to predict the truth value of hand-written statements about objects and their properties. While their experiments investigate BERT's

physical common sense reasoning, we focus on the different phenomenon of entailment between two actions expressed by verbs in context.

Schick and Schütze (2020) used handcrafted patterns and LMs for few-shot text classification. Based on manually defined label-token correspondences, the predicted classification label is determined by the token an LM estimates as most probable at a masked position in the cloze-style pattern. We differentiate entailment and non-entailment via compatibility scores for patterns and antipatterns and not via different predicted tokens.

Addressing relation induction, Bouraoui et al. (2020) propose an automatic way of finding, given a relation, LM patterns that are likely to express it. They train a binary classifier per relation on the sentences generated by these patterns. While some of the relations they consider are related to verbal entailment (e.g., *cook activity-goal eat*), most of them concern common sense (e.g., *library location-activity reading*) or encyclopedic knowledge (e.g., *Paris capital-of France*). We adapt their method for the automatic retrieval of promising patterns for LIiC, but find that handcrafted patterns that capture the generality of the entailment relation still have an advantage over automatic patterns for LIiC. Another important novelty we introduce is the use of antipatterns. While Bouraoui et al. (2020) have to use negative samples for training their classifiers, they only consider patterns that exemplify the desired relation. In contrast, we also use antipatterns that exemplify what the entailment relation is **not**. We believe that antipatterns are particularly useful for entailment detection because they can help identify other kinds of semantic relations that often pose a challenge to vector space models (Levy and Dagan, 2016; Schmitt and Schütze, 2019).

## 3 Proposed Approaches

### 3.1 NLI classifier

Building an NLI classifier on top of a pretrained LM usually means taking an aggregate sequence representation of the concatenated premise and hypothesis as input features of a neural network classifier (Devlin et al., 2019b). For RoBERTa (Liu et al., 2019), this representation is the final hidden state of a special $\langle \mathtt{s} \rangle$ token that is prepended to the input sentences, which in turn are separated by a separator token $\langle / \mathtt{s} \rangle$. Let $\Lambda$ be the function that maps such an input $x = x_1 \langle / \mathtt{s} \rangle x_2$ to the aggregate representation $\Lambda(x) \in \mathbb{R}^d$. Following (Devlin et al., 2019b; Liu et al., 2019), we then feed these features to a 2-layer feed-forward neural network with tanh activation:

$$\begin{aligned} h(x) &= \tanh(\mathrm{drop}(\Lambda(x))W_1 + b_1) \\ P_{\text{NLI}}(y \mid x) &= \sigma(\mathrm{drop}(h(x))W_2 + b_2) \end{aligned} \quad (1)$$

where drop applies dropout with a probability of 0.1, $\sigma$ is the softmax function, and $W_1 \in \mathbb{R}^{d \times d}, W_2 \in \mathbb{R}^{d \times 2}, b_1 \in \mathbb{R}^d, b_2 \in \mathbb{R}^2$ are learnable parameters. Note that $W_1$ and $b_1$ are still part of the LM's pretrained parameters; so we only train $W_2$ and $b_2$ from scratch.[2] The actual classification decision uses a threshold $\vartheta$:

$$D^{\vartheta}_{\text{NLI}}(x_1, x_2) = \begin{cases} 1, & \text{if } P_{\text{NLI}}(y = 1 \,|\, x_1, x_2) > \vartheta \\ 0, & \text{otherwise} \end{cases}$$

The traditional choice for the threshold is $\vartheta = 0.5$ because that means $D^{\vartheta}_{\text{NLI}}(x_1, x_2) = 1$ iff $P_{\text{NLI}}(y = 1 \mid x_1, x_2) > P_{\text{NLI}}(y = 0 \mid x_1, x_2)$. We nevertheless keep $\vartheta$ as a hyperparameter to be tuned on held-out development data.

We train the NLI approach by minimizing the negative log-likelihood $\mathcal{L}_{\text{NLI}}$ of the training data $\mathcal{T}$:

$$\mathcal{L}_{\text{NLI}}(\mathcal{T}) = \sum_{(x_1, x_2, y) \in \mathcal{T}} -\log(P_{\text{NLI}}(y \mid x_1, x_2))$$

### 3.2 Pattern-based classifier

This approach puts the input sentences $x_1, x_2$ together in a pattern-based textual context and trains a classifier to distinguish between felicitous and infelicitous utterances.[3] In contrast to previous approaches (Forbes et al., 2019; Bouraoui et al., 2020), we also consider antipatterns that exemplify what kind of semantic relatedness we are not interested in, and combine probabilities for patterns and antipatterns in the final classification.

**Finding suitable patterns.** A simple handcrafted pattern to check for the validity of an inference $x_1 \Rightarrow x_2$ is "$x_2$ *because* $x_1$.". An analoguos antipattern is "*It is not sure that* $x_2$ *just because* $x_1$.". Based on similar considerations, we manually design 5 patterns and 5 antipatterns (see Table 4). We will refer to the approach using these handcrafted patterns as MANPAT.

Bouraoui et al. (2020) argue that text produced by simple, handcrafted patterns is artificial and

[2] We follow the official implementation; cf. Jacob Devlin's comment on issue 43 in the BERT GitHub repository, https://github.com/google-research/bert/issues/43, (accessed 19 January 2021).

[3] Bouraoui et al. (2020) called this natural vs. unusual.

therefore suboptimal for LMs pretrained on naturally occurring text. To adapt their setup to verbal expressions used in LIiC, we identify suitable patterns (antipatterns) by searching a large text corpus[4] for sentences that contain both elements of valid (invalid) entailment pairs. In a second step, we score each of these patterns (antipatterns) according to the number of valid (invalid) entailment pairs $x_1, x_2$ that can be found by querying an LM for the $k$ most probable completions when $x_1$ or $x_2$ is inserted in the pattern and its counterpart is masked. For example, consider the entailment pair *rule* ⇒ *control* and the pattern "*Catchers* ***prem*** *the field; they* ***hypo*** *the plays and tell everyone where to be.*" extracted from a description of softball. Predicting *rule* from "*Catchers* ⟨mask⟩ *the field; they control the plays and tell everyone where to be.*" and predicting *control* from "*Catchers rule the field; they* ⟨mask⟩ *the plays and tell everyone where to be.*" would result in one point each. Approaches called AUTPAT$_n$ use the $n$ patterns with the most points obtained in that manner. See §4 for more details on our experimental setup.

**Pattern-based predictions.** The probability $P_{\text{FEL}}(z \mid x)$ of sentence $x$ to be felicitous ($z=1$) or infelicitous ($z=0$) is estimated like $P_{\text{NLI}}$ in Eq. (1), except that $x$ is not the concatenation of two sentences but a single pattern-generated utterance.

Given a set of patterns $\Phi$ and a set of antipatterns $\Psi$, the score $s$ to judge an input $x_1, x_2$ is the difference between the maximum probability $m_{\text{pos}}$ that any pattern forms a felicitous statement and the maximum probability $m_{\text{neg}}$ that any antipattern forms a felicitous statement:

$$m_{\text{pos}} = \max_{\varphi \in \Phi} P_{\text{FEL}}(z = 1 \mid \varphi(x_1, x_2))$$
$$m_{\text{neg}} = \max_{\psi \in \Psi} P_{\text{FEL}}(z = 1 \mid \psi(x_1, x_2))$$
$$s(x_1, x_2) = m_{\text{pos}} - m_{\text{neg}}$$

As in NLI, the final decision uses a threshold $\vartheta$:

$$D^{\vartheta}_{\text{PAT}}(x_1, x_2) = \begin{cases} 1, & \text{if } s(x_1, x_2) > \vartheta \\ 0, & \text{otherwise} \end{cases}$$

This corresponds to requiring that $m_{\text{pos}}$ be higher than $m_{\text{neg}}$ by a margin $\vartheta$, i.e., $D^{\vartheta}_{\text{PAT}}(x_1, x_2) = 1$ iff $m_{\text{pos}} > m_{\text{neg}} + \vartheta$.

As Bouraoui et al. (2020) did not use antipatterns, they defined $m_{\text{neg}}$ as the maximum probability for any pattern to form an infelicitous statement.

[4]We use the Wikipedia dump from Jan 15th 2011.

| | | Levy/Holt | SherLIiC |
|---|---|---|---|
| dev$_1$ | train | 4,388 | 797 |
| | dev$_2$ | 1,098 | 201 |
| test | | 12,921 | 2,990 |

Table 1: Data split sizes as used in our experiments.

To estimate the usefulness of antipatterns, we evaluate both possibilities, marking systems that use both patterns and antipatterns with $\Phi\Psi$ and those that only use patterns with $\Phi$.

The use of a threshold is another novel component, i.e., Bouraoui et al. (2020) virtually set $\vartheta = 0$. We discuss the influence of $\vartheta$ in §5.

We train all pattern-based approaches by minimizing the negative log-likelihood $\mathcal{L}_{\text{PAT}}$ that patterns $\Phi$ produce felicitous statements for valid entailments ($y = 1$) and infelicitous statements for invalid entailments ($y = 0$) from the training data $\mathcal{T}$, and vice versa for antipatterns $\Psi$:

$$\mathcal{L}_{\text{PAT}}(\mathcal{T}, \Phi, \Psi) = \sum_{(x_1, x_2, y) \in \mathcal{T}} \mathcal{L}_{\Phi}(x_1, x_2, y) + \mathcal{L}_{\Psi}(x_1, x_2, 1 - y)$$

with

$$\mathcal{L}_{\Omega}(x_1, x_2, y) = -\frac{1}{|\Omega|} \sum_{\omega \in \Omega} \log(P_{\text{FEL}}(z = y \mid \omega(x_1, x_2)))$$

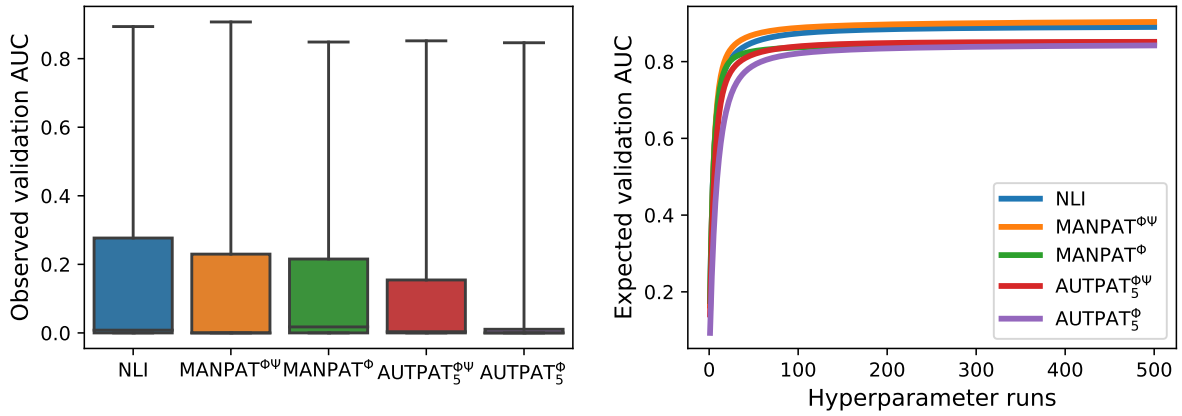
## 4 Experiments

We evaluate on two benchmarks: (i) Levy/Holt's dataset (Levy and Dagan, 2016; Holt, 2018) and (ii) SherLIiC (Schmitt and Schütze, 2019). For both filtering and classification, we employ RoBERTa-base (Liu et al., 2019). For classification only, we also report results for RoBERTa-large.

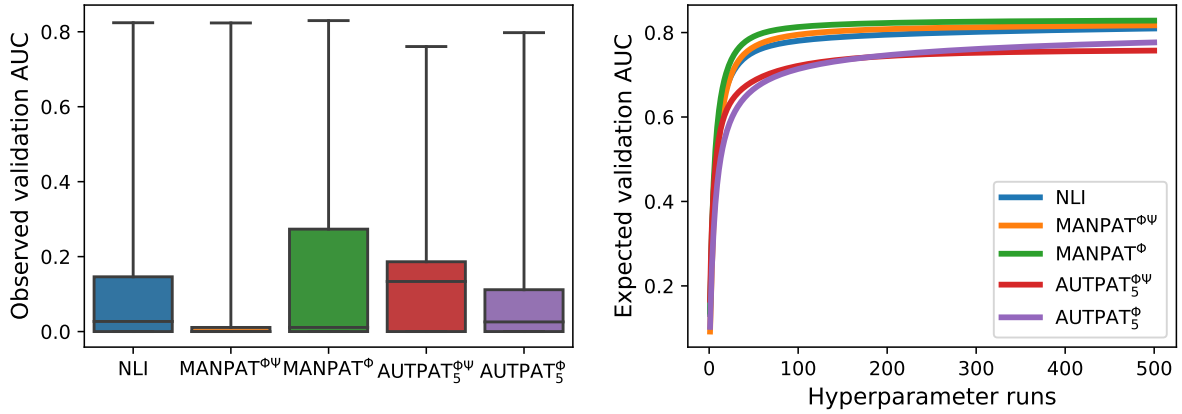
### 4.1 Data processing

For both datasets, previous work has established a dev/test split. For Levy/Holt, it was defined in (Hosseini et al., 2018); for SherLIiC, we use the original one from (Schmitt and Schütze, 2019). For comparison with previous work, we keep the test portion as is and split the dev portion further into 80% for training and 20% for development. We call the new, smaller dev sets dev$_2$ and the original dev sets dev$_1$. See Table 1 for data split sizes.

**Levy/Holt.** An instance in Levy/Holt has two sentences, each consisting of two shared noun




(a) Levy/Holt dev$_2$ RoBERTa-base




(b) SherLIiC dev$_2$ RoBERTa-base

Figure 1: Validation performance distribution of different datasets across different hyperparameter runs (left) and expected validation performance per number of tested hyperparameter configurations as proposed by Dodge et al. (2019) (right). Performance is measured as the area under the precision-recall curve for precision values $\geq 0.5$. The Boxes represent 75% of the respective data points; a black line indicates the median, whiskers extend to the maximum value.

phrases (the arguments) and a verbal expression, in which the two sentences differ. As the verbal expressions can contain auxiliaries or negation, they often consist of multiple tokens. Originally, one argument is replaced with a WordNet (Miller, 1995) type in one of the sentences to make the entailment more general during annotation, but we use a version of the dataset provided by Hosseini et al. (2018) where both sentences have concretely instantiated arguments. For example, consider Table 6 (c). *Athena* was masked as the WordNet synset *deity* during benchmark annotation but we use the original sentences as shown in Table 6 for all classifiers without further modification.

For the automatic pattern search in AUTPAT, we look for sentences that mention verbatim the two verbal expressions of any instance from dev$_1$. For the ranking, we take the last token of a verbal expression as representative for the whole. This has the advantage that we can query the LM with a single ⟨mask⟩ token and compare a single token to the $k = 100$ most probable predictions. We take the last token because it usually is the main verb.

**SherLIiC.** For classification, we use SherLIiC's automatically generated sentences that were used for annotation during benchmark creation. The arguments in SherLIiC are entity types from Freebase (Bollacker et al., 2008). As such, they can be replaced by any Freebase entity with matching type. For example, consider Table 6 (a); the arguments *Germany* and *Côte d'Ivoire* were originally masked as *location[A]* and *location[B]* during annotation, but annotators also saw three randomly chosen instantiations for both *A* (*Germany / Syria / USA*) and *B* (*Côte d'Ivoire / UK / Italy*) for context. From the three examples provided in SherLIiC for each argument, we choose the first one to form sentences with concretely instantiated arguments.

For the automatic pattern search in AUTPAT, we make use of the greater flexibility offered by the

| | | AUC | P | R | F1 |
|---|---|---|---|---|---|
| baselines | | | | | |
| Hosseini et al. (2018) | | 16.5 | – | – | – |
| Hosseini et al. (2019) | | **18.7** | – | – | – |
| RoBERTa-base | | | | | |
| NLI | ($\vartheta = 0.0052$) | 72.6 | 68.7 | **75.3** | 71.9 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = -0.0909$) | **76.9** | **78.7** | 66.4 | **72.0** |
| MANPAT$^{\Phi}$ | ($\vartheta = 0.5793$) | 71.2 | 74.4 | 61.2 | 67.1 |
| AUTPAT$_5^{\Phi\Psi}$ | ($\vartheta = -0.1428$) | 63.7 | 71.0 | 58.8 | 64.3 |
| AUTPAT$_5^{\Phi}$ | ($\vartheta = -0.0592$) | 65.4 | 68.0 | 63.3 | 65.5 |
| RoBERTa-large | | | | | |
| NLI | ($\vartheta = 0.0016$) | 75.5 | 73.5 | 73.7 | 73.6 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = 0.1156$) | **83.9** | **84.8** | 70.1 | **76.7** |
| MANPAT$^{\Phi}$ | ($\vartheta = -0.8457$) | 77.8 | 67.9 | **81.5** | 74.1 |
| AUTPAT$_5^{\Phi\Psi}$ | ($\vartheta = -0.0021$) | 70.4 | 75.7 | 60.7 | 67.4 |
| AUTPAT$_5^{\Phi}$ | ($\vartheta = -0.9197$) | 66.5 | 61.8 | 74.4 | 67.5 |

Table 2: Levy/Holt test. AUC denotes the area under the precision-recall curve for precision $\geq 0.5$. All results in %. Bold means best result per column and block.

| | | AUC | P | R | $F_1$ |
|---|---|---|---|---|---|
| baselines | | | | | |
| Lemma | | – | **90.7** | 8.9 | 16.1 |
| w2v+untyped_rel | | – | 52.8 | 69.5 | 60.0 |
| w2v+tsg_rel_emb | | – | 51.8 | **72.7** | **60.5** |
| RoBERTa-base | | | | | |
| NLI | ($\vartheta = 0.3878$) | 65.8 | **67.0** | 66.1 | 66.5 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = -0.3324$) | 66.4 | 60.9 | 78.8 | 68.7 |
| MANPAT$^{\Phi}$ | ($\vartheta = -0.4812$) | **69.2** | 62.0 | **81.2** | **70.3** |
| AUTPAT$_5^{\Phi\Psi}$ | ($\vartheta = -0.4694$) | 67.4 | 61.8 | 75.6 | 68.0 |
| AUTPAT$_5^{\Phi}$ | ($\vartheta = -0.7042$) | 67.3 | 56.6 | 82.6 | 67.2 |
| AUTCUR$_5^{\Phi}$ | ($\vartheta = -0.7524$) | **69.5** | 56.3 | **89.6** | **69.2** |
| AUTARG$_5^{\Phi}$ | ($\vartheta = -0.7461$) | 65.2 | **61.9** | 75.6 | 68.1 |
| RoBERTa-large | | | | | |
| NLI | ($\vartheta = 0.0025$) | 68.3 | 60.5 | **85.5** | 70.9 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = -0.0956$) | **74.4** | **66.0** | 80.8 | **72.6** |
| MANPAT$^{\Phi}$ | ($\vartheta = -0.6641$) | 64.6 | 58.1 | 79.0 | 67.0 |
| AUTPAT$_5^{\Phi\Psi}$ | ($\vartheta = -0.9889$) | 68.6 | 61.9 | 75.5 | 68.0 |
| AUTPAT$_5^{\Phi}$ | ($\vartheta = -0.5355$) | 56.8 | 61.5 | 66.1 | 63.7 |

Table 3: SherLIiC test. Baseline results from (Schmitt and Schütze, 2019). Table format: see Table 2.

lemmatized representations in SherLIiC. As we are interested in statements that can be made in any way in a text, we search for sentences that mention the two predicates of a SherLIiC $\text{dev}_1$ instance in any inflected form. For the ranking, we again consider the predicate representative for the whole verbal expression. We thus use the predicate lemma and otherwise proceed as described above.

### 4.2 Training details

We train all our classifiers for 5 epochs with Adam (Kingma and Ba, 2015) and a mini-batch size of 10 (resp. 2) for RoBERTa-base (resp. -large). We randomly sample 500 configurations for the remaining hyperparameters (see Appendix A). For a fair comparison, we evaluate all our approaches with the same configurations.

## 5 Results and Discussion

### 5.1 Hyperparameter robustness

Following previous work (Hosseini et al., 2018, 2019), we use the area under the precision-recall curve (AUC) restricted to precision values $\geq 0.5$ as criterion for model selection.

Fig. 1 (left) shows the distribution of $\text{dev}_2$ performance for 500 randomly sampled runs with RoBERTa-base. Most hyperparameters perform poorly, suggesting that hyperparameter search is crucial. For Levy/Holt, NLI is strong whereas for SherLIiC handcrafted MANPAT$^{\Phi}$ patterns have a clearer advantage. For SherLIiC, the combination of automatically generated patterns and antipatterns AUTPAT$_5^{\Phi\Psi}$ exhibits the highest median performance and the second-highest upper quartile, making it together with MANPAT$^{\Phi}$ the most robust to different hyperparameters, although its top performance is lower compared to the others. For all methods, only very few hyperparameter sets achieve top performances. For both datasets, however, a well-performing configuration is found after fewer than 100 sampled runs (Fig. 1, right). Considering that AUTPAT requires an LM to rank thousands of patterns, these results suggest that, for LIiC, available GPU hours should be spent on automatic hyperparameter rather than pattern search. With its manually written patterns, MANPAT does not need additional GPU hours for pattern search and still, on average, performs better.

### 5.2 Best hyperparameter configurations

For the best found configuration for each method, we not only report AUC, which provides a general picture of a scoring method's precision-recall tradeoff, but also the concrete precision, recall, and F1 for the actual classification after applying a threshold $\vartheta$. For this we tune $\vartheta$ on $\text{dev}_2$ for optimal F1. Tables 2 and 3 show the results.

On both datasets, our methods outperform all previous work (sometimes by a large margin), thus establishing a new state of the art. For SherLIiC+RoBERTa-base, the strong but simple NLI system is consistently outperformed by all pattern-based approaches, showing that well-

| *Automatically retrieved patterns (with SherLIiC dev$_1$)* | | prem | hypo |
|---|---|---|---|
| rank 1 | In North America, where the "atypical" forms of community-**hypo** pneumonia are becoming more common, macrolides (such as azithromycin), and doxycycline have displaced amoxicillin as first-line outpatient treatment for community-**prem** pneumonia. | acquired | acquired |
| rank 5 | This area now consists of … the Yukon Territory (**prem** 1898) … and Nunavut (**hypo** 1999). | created | created in |
| rank 12 | For example, … 訪問 "**prem**" is composed of 訪 "to visit" and 問 "to **hypo**". | interview | ask |
| *Handcrafted patterns* | | | |
| (a) | PARGL **prem** PARGR, which means that HARGL **hypo** HARGR. | | |
| (b) | It is not the case that HARGL **hypo** HARGR, let alone that PARGL **prem** PARGR. | | |
| (c) | HARGL **hypo** HARGR because PARGL **prem** PARGR. | | |
| (d) | PARGL **prem-negated** PARGR because HARGL **hypo-negated** HARGR. | | |
| (e) | HARGL **hypo-negated** HARGR, which means that PARGL **prem-negated** PARGR. | | |

Table 4: Examples of automatically retrieved and ranked AUTPAT patterns (top) and handcrafted MANPAT patterns (bottom). prem/hypo = original fillers as found in the corpus. PARGL/HARGL = placeholder for left argument of premise/hypothesis; PARGR/HARGR = right argument.

chosen patterns and antipatterns can be helpful for LIiC. For SherLIiC+RoBERTa-large and also generally on Levy/Holt's dataset, NLI is more competitive, but the combination of handcrafted patterns and antipatterns MANPAT$^{\Phi\Psi}$ still performs better in these cases.

The use of antipatterns does not consistently lead to better performance for all combinations of dataset, LM variant (base vs. large), and pattern set (MANPAT vs. AUTPAT). They do, however, consistently bring gains for some combinations, e.g., MANPAT on Levy/Holt and AUTPAT on SherLIiC. Moreover, antipatterns are essential for achieving top performance, i.e., the new state of the art, on both datasets.

Most of the threshold values $\vartheta$ (tuned on dev$_2$) are far from their traditional values, 0.5 for NLI and 0.0 for patterns. NLI classifiers' probability estimates are often too confident, resulting in values close to 0 and 1. To "correct" cases where a very small value is assigned to a valid entailment, optimal thresholds are often close to 0 instead of 0.5. Analogously, most pattern-based approaches opt for a negative $\vartheta$, which means that instead of requiring a margin between $m_{\text{pos}}$ and $m_{\text{neg}}$ (boosting precision), they make more positive predictions and boost recall. Low recall is a key problem in LIiC (cf. Levy and Dagan (2016)). Tuning a threshold increases the models' flexibility in this aspect.

# 6 Analysis

## 6.1 Number of patterns

§5 shows that automatic patterns do not beat handcrafted patterns for LIiC. However, automatic patterns have one major advantage: in contrast to manual patterns, their number can be easily increased. We therefore investigate the impact of the hyperparameter $n$ for AUTPAT$_n$.

| | ΦΨ | | Φ | |
|---|---|---|---|---|
| $n$ | AUC | F1 | AUC | F1 |
| 5 | 67.4 | 68.0 | 67.3 | 67.2 |
| 15 | **70.0** | **68.7** | **73.1** | **69.4** |
| 25 | 63.5 | 67.3 | 69.0 | 68.7 |
| 50 | 66.3 | 65.6 | 67.4 | 67.6 |

Table 5: RoBERTa-base+AUTPAT$_n$ results on SherLIiC test for different $n$ values. Hyperparameters were tuned for the corresponding AUTPAT$_5$ method on dev$_2$.

Table 5 shows that too many patterns is as bad as too few. AUTPAT$_{15}$ is the sweetspot: on SherLIiC, it outperforms all other RoBERTa-base methods on AUC and closely approaches the otherwise best method MANPAT$^{\Phi}$ on F1.

## 6.2 Pattern analysis

Handcrafted patterns mostly outperform automatic ones (§5). A larger number $n$ of patterns only has a small effect (§6.1). We therefore take a closer look at automatic and manual patterns. Table 4 shows all handcrafted and a sample of highly ranked automatic patterns.

It is striking how specific the automatically retrieved contexts are; especially for the highest ranks (exemplified by ranks 1 and 5) only a narrow set of verbs seems plausible from a human perspective. It is only at rank 12 that we find a more general context and it arguably even displays some semantic reasoning. There certainly are verbs that are not compatible with the meaning of *visit*, but this context allows for a wide range of plausible verbs

and even mentions composition of meaning.

The handcrafted patterns, in contrast, all capture some general aspect of entailment, which might be the reason they generalize better. Moreover, they also have placeholder slots for the verb arguments, which could be an advantage as these represent a verb's original context. Only accepting corpus sentences in which the verbs occur with the same arguments as in the dataset is too restrictive.

We therefore conduct the following experiment: We manually go through the 100 highest-ranked automatically created patterns and identify 5 contexts that could accommodate arguments without changing the overall sentence structure. We also try to pick patterns that are different enough from each other to avoid redundancy. As a baseline, the method AUTCUR$_5^\Phi$ uses these manually curated patterns as is. We then rewrite the patterns such that they include placeholders for verb arguments, e.g., "*The original aim of de Garis' work was to* ***prem*** *the field of "brain building" (a term of his invention) and to "****hypo*** *a trillion dollar industry within 20 years".*" becomes "*The original aim of their work was that "*PARGL ***prem*** PARGR*" and that "*HARGL ***hypo*** HARGR *within 20 years".*" with PARGL / PARGR (HARGL / HARGR) the placeholder for the left / right argument of the premise (hypothesis). See Table 14 in the appendix for the complete list. AUTARG$_5^\Phi$ is based on these rewritten patterns. We try the same 500 hyperparameter configurations as for the other RoBERTa-base approaches and include results for the best configuration (chosen on dev$_2$) in Table 3. We find that manually curating automatically ranked patterns helps performance. AUTCUR$_5^\Phi$ outperforms AUTPAT$_5^\Phi$ on AUC and F1, reducing the gap to handcrafted patterns (i.e., MANPAT$^\Phi$). This is probably due to the variety we enforced when handpicking the patterns.

Surprisingly, adding arguments decreases performance. Possibly, our modifications make the patterns less fluent or the arguments that are filled into the placeholders during training and evaluation do not fit well into the contexts, which still are rather specific.

### 6.3 Error analysis

Table 6 displays a selection of the dev$_2$ sets of our two benchmarks along with the predictions of all our approaches.

The first four examples indicate how NLI differs from pattern approaches. Example (a) involves the common sense knowledge that occupying a territory implies remaining there. This might be learned from patterns more easily as these patterns might resemble contexts – seen during pretraining – that describe how long a military force remained during an occupation. Putting the inference candidate (b) into a pattern-generated context avoids being fooled by the high similarity of the two sentences. Only the handcrafted patterns can make sense of the important details in this construction.

In contrast, (c) and (d) are difficult for our pattern approaches whereas NLI gets them right. We hypothesize that the problem stems from linking the two sentences into one. An entailment pattern ideally represents a derivation of the hypothesis from the premise. One may wrongly conclude that (c) *Athena was the goddess of Athens only because she was worshiped there*, by neglecting the possibility that there are others that are equally worshiped. In the same way, (d) is unlikely to be found in an argumentative text. While it is clear that there can be no beating without a fight, one would hardly argue that *Pyrrhus fought the romans because they beat him*. This particular reasoning calls for additional explanations like *Pyrrhus must have fought the romans because I know that they beat him*. This analysis serves as inspiration for further improve-

| | | |
|---|---|---|
| (a) | *Germany is occupying Côte d'Ivoire* ⇒ *Germany is remaining in Côte d'Ivoire* | |
| Sh | truth: 1 NLI: 0 MANPAT: 1 / 0 AUTPAT: 1 / 1 | |
| (b) | *Ford awarded him the medal* ⇒ *Ford was awarded a medal* | |
| L/H | truth: 0 NLI: 1 MANPAT: 0 / 0 AUTPAT: 1 / 1 | |
| (c) | *Athena was worshiped in Athens* ⇒ *Athena was the goddess of Athens* | |
| L/H | truth: 0 NLI: 0 MANPAT: 0 / 1 AUTPAT: 1 / 1 | |
| (d) | *Pyrrhus was beaten by the romans* ⇒ *Pyrrhus fought the romans* | |
| L/H | truth: 1 NLI: 1 MANPAT: 0 / 0 AUTPAT: 0 / 1 | |
| (e) | *England national rugby union team is playing against Denver Broncos* ⇒ *England national rugby union team is beating Denver Broncos* | |
| Sh | truth: 0 NLI: 1 MANPAT: 1 / 1 AUTPAT: 1 / 1 | |
| (f) | *Polk negotiated with Britain* ⇒ *Polk made peace with Britain* | |
| L/H | truth: 0 NLI: 1 MANPAT: 1 / 1 AUTPAT: 1 / 1 | |

Table 6: Qualitative error analysis of the RoBERTa-base models from Tables 2 and 3 on the dev$_2$ split of SherLIiC (Sh) and Levy/Holt (L/H). Pattern-based predictions are listed in the format ΦΨ / Φ. Correct predictions are green; errors are underlined and red.

ments of entailment patterns.

The last two examples (e) and (f) are difficult for all approaches. It seems to be a particular challenge to identify open situations like a sports match or a negotiation where multiple outcomes are possible and distinguish them from cases where one particular outcome is inevitable.

## 7 Conclusion

We proposed and evaluated three approaches to the task of lexical inference in context (LIiC) based on pretrained language models (LMs). In particular, we found that putting an inference candidate into a pattern-generated context mostly increases performance compared to a standard sequence classification approach. Concrete performance, however, also depends on the particular dataset, used LM (variant), and pattern set. We introduced the concept of antipatterns, which express the negative class of a binary classification, and found that they often lead to performance gains for LIiC. We set a new state of the art for LIiC and conducted an extensive analysis of our approaches. Notably, we found that automatically created patterns can perform nearly as well as handcrafted ones if we either use the right number $n$ of patterns or manually identify the right subset of them. Promising directions for future work are the investigation of alternative automatic pattern generation methods or a better modeling of the remaining challenges we described in our error analysis (§6.3).

## Acknowledgments

We gratefully acknowledge a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes). This work was supported by the BMBF as part of the project MLWin (01IS18050).

## References

Asaf Amrami and Yoav Goldberg. 2018. Word sense induction with neural biLM and symmetric patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4867, Brussels, Belgium. Association for Computational Linguistics.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 610–619, Portland, Oregon, USA. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from bert. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Computing Research Repository*, arXiv:2005.14165.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 33–40, Barcelona, Spain. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing textual entailment: Models and applications*. Morgan & Claypool Publishers.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding. *Computing Research Repository*, arXiv:1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Christiane Fellbaum and George A. Miller. 1990. Folk psychology or semantic entailment? comment on rips and conrad (1989). *Psychological Review*, 97(4):565–570.

Maxwell Forbes, Ari Holtzman, and Yejin Choi. 2019. Do neural language representations learn physical commonsense? In *Proceedings of the 41th Annual Meeting of the Cognitive Science Society (CogSci 2019)*, pages 1753–1759. cognitivesciencesociety.org.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

Xavier R. Holt. 2018. Probabilistic models of relational implication. Master's thesis, Macquarie University.

Mohammad Javad Hosseini, Nathanael Chambers, Siva Reddy, Xavier R. Holt, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2018. Learning typed entailment graphs with global soft constraints. *Transactions of the Association for Computational Linguistics*, 6:703–717.

Mohammad Javad Hosseini, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2019. Duality of link prediction and entailment graph induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4736–4746, Florence, Italy. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. 2015. Exploiting image generality for lexical entailment detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 119–124, Beijing, China. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389.

Omer Levy and Ido Dagan. 2016. Annotating relation inference in context via question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 249–255, Berlin, Germany. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, Montreal, Quebec, Canada. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. DIRT: Discovery of Inference Rules from Text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 323–328, New York, NY, USA. ACM Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Computing Research Repository*, arXiv:1907.11692.

Yehudit Meged, Avi Caciularu, Vered Shwartz, and Ido Dagan. 2020. Paraphrasing vs coreferring: Two sides of the same coin. *Computing Research Repository*, arXiv:2004.14979.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computing Research Repository*, arXiv:1301.3781.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 579–586, Sydney, Australia. Association for Computational Linguistics.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2163–2172, Austin, Texas. Association for Computational Linguistics.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–363, Melbourne, Australia. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *Computing Research Repository*, arXiv:2001.07676.

Martin Schmitt and Hinrich Schütze. 2019. SherLIiC: A typed event-focused lexical inference benchmark for evaluating natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 902–914, Florence, Italy. Association for Computational Linguistics.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098, Cambridge, MA. Association for Computational Linguistics.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 258–267, Beijing, China. Association for Computational Linguistics.

Vered Shwartz, Omer Levy, Ido Dagan, and Jacob Goldberger. 2015. Learning to exploit structured resources for lexical inference. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 175–184, Beijing, China. Association for Computational Linguistics.

Vered Shwartz, Gabriel Stanovsky, and Ido Dagan. 2017. Acquiring predicate paraphrases from news tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 155–160, Vancouver, Canada. Association for Computational Linguistics.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 849–856, Manchester, UK. Coling 2008 Organizing Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1134–1145, New Orleans, Louisiana. Association for Computational Linguistics.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 81–88.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

## A Hyperparameters

We train all our classifiers for 5 epochs with the Adam optimizer (Kingma and Ba, 2015) and a mini-

batch size of 10 or 2 instances for RoBERTa-base and -large, respectively. For AUTPAT$_n$ approaches with $n > 5$, we distribute the available patterns and antipatterns into chunks of size 5 for training to save memory. During evaluation, the predictions are based on all the patterns and antipatterns.

We randomly sample 500 configurations for the remaining hyperparameters, i.e., initial learning rate lr, weight decay $\lambda$ (L2 regularization), and the number of batches $c$ the gradient is accumulated before each optimization step, which virtually increases the batch size by a factor of $c$. The hyperparameters are sampled from the following intervals: $\text{lr} \in [10^{-8}, 5 \cdot 10^{-2}]$, $\lambda \in [10^{-5}, 10^{-1}]$, $c \in \{1, 2, \ldots, 10\}$. lr and $\lambda$ are sampled uniformly in log-space. For a fair comparison, we use the same 500 random configurations for all of our approaches.

As usual for Transformer models, we apply a learning rate schedule: lr decreases linearly such that it reaches 0 at the end of the last epoch. We do not employ warm-up.

The best configurations can be seen in Tables 8 and 10 for Levy/Holt's dataset and in Tables 9 and 11 for SherLIiC.

## B Results on development sets

See Tables 12 and 13.

## C Varying $n$ in training and evaluation

Another approach to make use of different values of $n$ in AUTPAT$_n$ systems is to vary $n$ from training to evaluation. Figure 2 is a visualization of the performance impact of this procedure. The base point for the visualization (in white) is the AUC performance of AUTPAT$_5^\Phi$. We see that training with $n = 50$ almost always leads to a performance drop (marked in blue) w.r.t. this number. It seems generally to be catastrophic to evaluate a model with patterns that were not seen during training, indicating that there is no generalization from seen patterns to unseen patterns even if they were chosen by the same method and can thus be expected to be – at least to some extent – similar. In general, this evaluation suggests that modifying $n$ after the training always leads to a drop in performance.

## D Transfer between Datasets

Table 7 shows results on the question how well a model trained on one dataset performs on the other. For this, we assume that the target dataset is not available at all, i.e., we do not use it at all – neither for finding patterns in AUTPAT nor for tuning the threshold $\vartheta$. We thus use the standard $\vartheta$ values, i.e., 0.5 for NLI and 0.0 for the pattern-based methods.

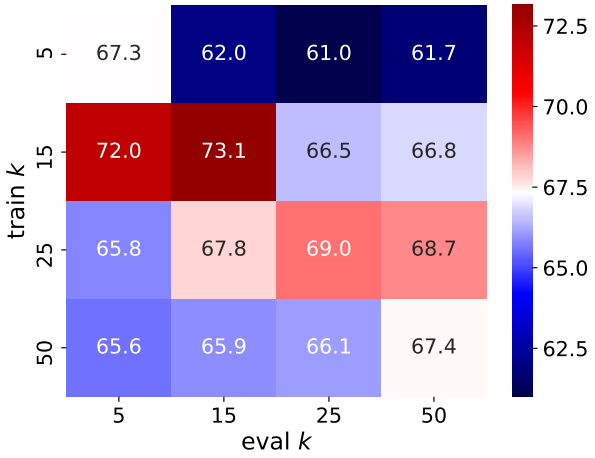



Figure 2: RoBERTa-base+AUTPAT$_k^\Phi$ performance on SherLIiC test for different $k$ values during training and evaluation. Same hyperparameters used for all models (as in Table 5). Blue marks drops, red marks gains in performance w.r.t. AUTPAT$_5^\Phi$.

| | AUC | P | R | $F_1$ |
|---|---|---|---|---|
| RoBERTa-base | | | | |
| NLI | 38.4 | 52.7 | 57.1 | **54.8** |
| MANPAT$^{\Phi\Psi}$ | **46.1** | **64.0** | 45.4 | 53.1 |
| MANPAT$^{\Phi}$ | 32.4 | 32.4 | **94.5** | 48.2 |
| AUTPAT$_5^{\Phi\Psi}$ | 18.7 | 40.5 | 35.0 | 37.6 |
| AUTPAT$_5^{\Phi}$ | 21.3 | 28.3 | 62.3 | 38.9 |
| RoBERTa-large | | | | |
| NLI | 37.8 | 31.0 | 96.4 | 46.9 |
| MANPAT$^{\Phi\Psi}$ | **70.4** | 39.6 | 95.3 | **56.0** |
| MANPAT$^{\Phi}$ | 38.9 | 25.6 | **98.3** | 40.6 |
| AUTPAT$_5^{\Phi\Psi}$ | 33.6 | **61.6** | 36.0 | 45.5 |
| AUTPAT$_5^{\Phi}$ | 9.3 | 30.7 | 76.6 | 43.8 |

(a) SherLIiC train → Levy/Holt test.

| | AUC | P | R | $F_1$ |
|---|---|---|---|---|
| RoBERTa-base | | | | |
| NLI | 63.3 | 62.8 | **68.4** | **65.5** |
| MANPAT$^{\Phi\Psi}$ | **69.1** | **80.5** | 42.1 | 55.3 |
| MANPAT$^{\Phi}$ | 68.4 | 80.1 | 24.2 | 37.2 |
| AUTPAT$_5^{\Phi\Psi}$ | 60.3 | 71.5 | 54.5 | 61.9 |
| AUTPAT$_5^{\Phi}$ | 58.9 | 68.6 | 55.7 | 61.5 |
| RoBERTa-large | | | | |
| NLI | 65.6 | 73.8 | 53.0 | 61.7 |
| MANPAT$^{\Phi\Psi}$ | 69.6 | 84.7 | 35.7 | 50.3 |
| MANPAT$^{\Phi}$ | **72.2** | **89.3** | 30.3 | 45.2 |
| AUTPAT$_5^{\Phi\Psi}$ | 62.1 | 68.1 | **57.3** | **62.3** |
| AUTPAT$_5^{\Phi}$ | 63.8 | 75.8 | 44.2 | 55.8 |

(b) Levy/Holt train → SherLIiC test.

Table 7: Transfer learning. Table format: see Table 2.

| | NLI | MANPAT$^{\Phi\Psi}$ | MANPAT$^{\Phi}$ | AUTPAT$^{\Phi\Psi}_5$ | AUTPAT$^{\Phi}_5$ |
|---|---|---|---|---|---|
| lr | $2.72 \cdot 10^{-5}$ | $2.47 \cdot 10^{-5}$ | $6.68 \cdot 10^{-6}$ | $3.82 \cdot 10^{-5}$ | $2.11 \cdot 10^{-5}$ |
| $\lambda$ | $1.43 \cdot 10^{-3}$ | $2.98 \cdot 10^{-4}$ | $1.07 \cdot 10^{-5}$ | $4.02 \cdot 10^{-5}$ | $1.65 \cdot 10^{-5}$ |
| $c$ | 1 | 2 | 1 | 2 | 3 |

Table 8: Levy/Holt; RoBERTa-base.

| | NLI | MANPAT$^{\Phi\Psi}$ | MANPAT$^{\Phi}$ | AUTPAT$^{\Phi\Psi}_5$ | AUTPAT$^{\Phi}_5$ | AUTCUR$^{\Phi}_5$ | AUTARG$^{\Phi}_5$ |
|---|---|---|---|---|---|---|---|
| lr | $6.34 \cdot 10^{-6}$ | $3.87 \cdot 10^{-5}$ | $2.28 \cdot 10^{-5}$ | $3.92 \cdot 10^{-5}$ | $2.53 \cdot 10^{-5}$ | $1.28 \cdot 10^{-5}$ | $2.47 \cdot 10^{-5}$ |
| $\lambda$ | $1.35 \cdot 10^{-3}$ | $1.43 \cdot 10^{-5}$ | $6.52 \cdot 10^{-2}$ | $2.18 \cdot 10^{-4}$ | $1.02 \cdot 10^{-5}$ | $8.23 \cdot 10^{-3}$ | $2.98 \cdot 10^{-4}$ |
| $c$ | 1 | 4 | 2 | 1 | 1 | 1 | 2 |

Table 9: SherLIiC; RoBERTa-base.

| | NLI | MANPAT$^{\Phi\Psi}$ | MANPAT$^{\Phi}$ | AUTPAT$^{\Phi\Psi}_5$ | AUTPAT$^{\Phi}_5$ |
|---|---|---|---|---|---|
| lr | $6.68 \cdot 10^{-6}$ | $4.55 \cdot 10^{-6}$ | $4.92 \cdot 10^{-6}$ | $6.68 \cdot 10^{-6}$ | $8.13 \cdot 10^{-6}$ |
| $\lambda$ | $1.07 \cdot 10^{-5}$ | $3.90 \cdot 10^{-4}$ | $3.61 \cdot 10^{-4}$ | $1.07 \cdot 10^{-5}$ | $6.05 \cdot 10^{-2}$ |
| $c$ | 1 | 2 | 3 | 1 | 2 |

Table 10: Levy/Holt; RoBERTa-large.

| | NLI | MANPAT$^{\Phi\Psi}$ | MANPAT$^{\Phi}$ | AUTPAT$^{\Phi\Psi}_5$ | AUTPAT$^{\Phi}_5$ |
|---|---|---|---|---|---|
| lr | $6.68 \cdot 10^{-6}$ | $1.29 \cdot 10^{-5}$ | $9.14 \cdot 10^{-6}$ | $6.34 \cdot 10^{-6}$ | $4.55 \cdot 10^{-6}$ |
| $\lambda$ | $1.07 \cdot 10^{-5}$ | $2.49 \cdot 10^{-4}$ | $6.09 \cdot 10^{-5}$ | $1.35 \cdot 10^{-3}$ | $3.90 \cdot 10^{-4}$ |
| $c$ | 1 | 3 | 4 | 1 | 2 |

Table 11: SherLIiC; RoBERTa-large.

| | | dev$_1$ | | | | dev$_2$ | | | | test | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AUC | P | R | F1 | AUC | P | R | F1 | AUC | P | R | F1 |
| baselines | | | | | | | | | | | | | |
| Hosseini et al. (2018) | | – | – | – | – | – | – | – | – | 16.5 | – | – | – |
| Hosseini et al. (2019) | | – | – | – | – | – | – | – | – | **18.7** | – | – | – |
| RoBERTa-base | | | | | | | | | | | | | |
| NLI | ($\vartheta = 0.0052$) | 94.9 | 87.4 | 91.1 | 89.2 | 88.8 | 78.1 | **90.3** | 83.8 | 72.6 | 68.7 | **75.3** | 71.9 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = -0.0909$) | **96.5** | **87.7** | **96.2** | **91.8** | **89.4** | **81.4** | 88.5 | **84.8** | **76.9** | **78.7** | 66.4 | **72.0** |
| MANPAT$^{\Phi}$ | ($\vartheta = 0.5793$) | 91.8 | 80.2 | 90.1 | 84.9 | 84.7 | 77.5 | 81.1 | 79.3 | 71.2 | 74.4 | 61.2 | 67.1 |
| AUTPAT$^{\Phi\Psi}_5$ | ($\vartheta = -0.1428$) | 95.0 | 83.4 | 95.6 | 89.1 | 87.7 | 79.2 | 85.7 | 82.3 | 63.7 | 71.0 | 58.8 | 64.3 |
| AUTPAT$^{\Phi}_5$ | ($\vartheta = -0.0592$) | 87.4 | 78.0 | 90.0 | 83.6 | 83.3 | 76.3 | 81.6 | 78.8 | 65.4 | 68.0 | 63.3 | 65.5 |
| RoBERTa-large | | | | | | | | | | | | | |
| NLI | ($\vartheta = 0.0016$) | 96.9 | 90.1 | 97.1 | **93.5** | 87.7 | 82.6 | 87.6 | **85.0** | 75.5 | 73.5 | 73.7 | 73.6 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = 0.1156$) | **97.1** | **91.4** | 95.4 | 93.4 | **88.9** | **84.0** | 84.8 | 84.4 | **83.9** | **84.8** | 70.1 | **76.7** |
| MANPAT$^{\Phi}$ | ($\vartheta = -0.8457$) | 92.2 | 76.1 | **97.3** | 85.4 | 84.4 | 72.5 | **91.2** | 80.8 | 77.8 | 67.9 | **81.5** | 74.1 |
| AUTPAT$^{\Phi\Psi}_5$ | ($\vartheta = -0.0021$) | 95.0 | 86.0 | 91.9 | 88.8 | 84.7 | 78.9 | 81.1 | 80.0 | 70.4 | 75.7 | 60.7 | 67.4 |
| AUTPAT$^{\Phi}_5$ | ($\vartheta = -0.9197$) | 92.4 | 75.5 | 95.3 | 84.2 | 83.5 | 70.6 | 88.5 | 78.5 | 66.5 | 61.8 | 74.4 | 67.5 |

Table 12: Full results on the Levy/Holt dataset. The dev and test sets as created by Hosseini et al. (2018) are called dev$_1$ and test. The portion of dev$_1$ that serves as our validation set is called dev$_2$. AUC denotes the area under the precision-recall curve for precision values $\geq 0.5$. All results in %.

| | | dev$_1$ | | | | dev$_2$ | | | | test | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AUC | P | R | $F_1$ | AUC | P | R | $F_1$ | AUC | P | R | $F_1$ |
| baselines | | | | | | | | | | | | | |
| Lemma | | – | **90.0** | 10.9 | 19.4 | – | – | – | – | – | **90.7** | 8.9 | 16.1 |
| w2v+untyped_rel | | – | 56.5 | 74.0 | 64.1 | – | – | – | – | – | 52.8 | 69.5 | 60.0 |
| w2v+tsg_rel_emb | | – | 56.6 | **77.6** | **65.5** | – | – | – | – | – | 51.8 | **72.7** | **60.5** |
| RoBERTa-base | | | | | | | | | | | | | |
| NLI | ($\vartheta = 0.3878$) | 81.3 | **79.1** | 80.1 | 79.6 | 81.5 | **84.2** | 70.6 | **76.8** | 65.8 | **67.0** | 66.1 | 66.5 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = -0.3324$) | 76.2 | 68.6 | 85.8 | 76.2 | **82.4** | 70.0 | **82.4** | 75.7 | 66.4 | 60.9 | 78.8 | 68.7 |
| MANPAT$^{\Phi}$ | ($\vartheta = -0.4812$) | **88.4** | 75.5 | **93.1** | **83.4** | 84.1 | 73.0 | 79.4 | 76.1 | **69.2** | 62.0 | **81.2** | **70.3** |
| AUTPAT$_5^{\Phi\Psi}$ | ($\vartheta = -0.4694$) | 87.0 | 77.8 | 88.8 | 82.9 | 71.2 | 68.4 | 76.5 | 72.2 | 67.4 | 61.8 | 75.6 | 68.0 |
| AUTPAT$_5^{\Phi}$ | ($\vartheta = -0.7042$) | 86.8 | 64.1 | 91.8 | 75.5 | 74.0 | 65.5 | 83.8 | 73.6 | 67.3 | 56.6 | 82.6 | 67.2 |
| AUTCUR$_5^{\Phi}$ | ($\vartheta = -0.7524$) | 82.6 | 61.7 | 92.8 | 74.1 | 75.6 | 60.6 | 88.2 | 71.9 | 69.5 | 56.3 | 89.6 | 69.2 |
| AUTARG$_5^{\Phi}$ | ($\vartheta = -0.7461$) | 77.4 | 69.3 | 84.0 | 76.0 | 73.6 | 68.9 | 75.0 | 71.8 | 65.2 | 61.9 | 75.6 | 68.1 |
| AUTPAT$_{15}^{\Phi\Psi}$ | ($\vartheta = -0.5263$) | 95.3 | 87.0 | 93.1 | 89.9 | 73.0 | 65.4 | 75.0 | 69.9 | 70.0 | 60.4 | 79.7 | 68.7 |
| AUTPAT$_{15}^{\Phi}$ | ($\vartheta = -0.6422$) | **95.4** | 85.3 | 94.6 | 89.7 | **75.8** | 69.2 | **79.4** | **74.0** | **73.1** | **63.0** | 77.4 | **69.4** |
| AUTPAT$_{25}^{\Phi\Psi}$ | ($\vartheta = -0.0014$) | 95.0 | **92.0** | 93.7 | **92.8** | 66.1 | **70.0** | 72.1 | 71.0 | 63.5 | 62.1 | 73.4 | 67.3 |
| AUTPAT$_{25}^{\Phi}$ | ($\vartheta = -0.6496$) | 88.1 | 72.3 | 90.0 | 80.2 | 73.0 | 67.5 | 79.4 | 73.0 | 69.0 | 60.5 | **79.4** | 68.7 |
| AUTPAT$_{50}^{\Phi\Psi}$ | ($\vartheta = -0.9163$) | 93.2 | 72.8 | 92.8 | 81.5 | 67.1 | 63.0 | 75.0 | 68.5 | 66.3 | 54.3 | **82.8** | 65.6 |
| AUTPAT$_{50}^{\Phi}$ | ($\vartheta = -0.9500$) | 94.2 | 79.1 | **94.9** | 86.3 | 69.3 | 66.3 | 77.9 | 71.6 | 67.4 | 57.3 | 82.5 | 67.6 |
| RoBERTa-large | | | | | | | | | | | | | |
| NLI | ($\vartheta = 0.0025$) | **92.3** | **79.7** | **93.7** | **86.1** | 75.7 | 66.7 | **82.4** | 73.7 | 68.3 | 60.5 | **85.5** | 70.9 |
| MANPAT$^{\Phi\Psi}$ | ($\vartheta = -0.0956$) | 89.3 | 77.3 | 88.5 | 82.5 | **80.8** | **74.7** | 77.9 | **76.3** | **74.4** | **66.0** | 80.8 | **72.6** |
| MANPAT$^{\Phi}$ | ($\vartheta = -0.6641$) | 78.0 | 67.4 | 84.9 | 75.1 | 72.2 | 67.1 | 77.9 | 72.1 | 64.6 | 58.1 | 79.0 | 67.0 |
| AUTPAT$_5^{\Phi\Psi}$ | ($\vartheta = -0.9889$) | 86.5 | 73.8 | 86.7 | 79.7 | 73.6 | 70.4 | 73.5 | 71.9 | 68.6 | 61.9 | 75.5 | 68.0 |
| AUTPAT$_5^{\Phi}$ | ($\vartheta = -0.5355$) | 71.6 | 64.3 | 71.9 | 67.9 | 64.5 | 71.4 | 66.2 | 68.7 | 56.8 | 61.5 | 66.1 | 63.7 |

Table 13: Full results on SherLIiC. The original dev and test sets are called dev$_1$ and test. The portion of dev$_1$ that serves as our validation set is called dev$_2$. AUC denotes the area under the precision-recall curve for precision values $\geq 0.5$. Baseline results from (Schmitt and Schütze, 2019). All results in %.

| | | | |
|---|---|---|---|
| (1) | The original aim of de Garis' work was to **prem** the field of "brain building" (a term of his invention) and to "**hypo** a trillion dollar industry within 20 years". | → | The original aim of their work was that "PARGL **prem** PARGR" and that "HARGL **hypo** HARGR within 20 years". |
| (2) | Critic Roger Ebert stated that Gellar and co-star Ryan Phillippe "**prem** a convincing emotional charge" and that Gellar is "effective as a bright girl who knows exactly how to **hypo** her act as a tramp". | → | Critic Roger Ebert stated that PARGL and co-star Ryan Phillippe "**prem** PARGR" and that HARGL is "effective as a bright girl who knows exactly how she **hypo** HARGR as a tramp". |
| (3) | Well-known professional competitions in the past have included the World Professional Championships (**hypo** Landover, Maryland), the Challenge Of Champions, the Canadian Professional Championships and the World Professional Championships (**prem** in Jaca, Spain). | → | Well-known professional competitions in the past have included HARGL (**hypo** HARGR), the Challenge Of Champions, the Canadian Professional Championships and PARGL (**prem** PARGR). |
| (4) | They also had sharpshooter Steve Kerr, whom they **hypo** via free agency before the 1993–94 season, Myers, and centers Luc Longley (**prem** via trade in 1994 from the Minnesota Timberwolves) and Bill Wennington. | → | HARGL also had sharpshooter HARGR, whom they **hypo** via free agency before the 1993–94 season, Myers, and centers PARGR (whom PARGL **prem** via trade in 1994 from the Minnesota Timberwolves) and Bill Wennington. |
| (5) | Because the 6x86 was more efficient on an instructions-per-cycle basis than Intel's Pentium, and because Cyrix sometimes **hypo** a faster bus speed than either Intel or AMD, Cyrix and competitor AMD co-**prem** the controversial PR system in an effort to compare its products more favorably with Intel's. … | → | Because the 6x86 was more efficient on an instructions-per-cycle basis than Intel's Pentium, and because HARGL sometimes **hypo** HARGR, PARGL and competitor AMD co-**prem** PARGR in an effort to compare its products more favorably with Intel's. … |

Table 14: Five manually selected patterns from the 100 highest-ranked automatically extracted patterns from SherLIiC dev$_1$ (used in AUTCUR$_5^{\Phi}$) and their rewritten counterparts (used in AUTARG$_5^{\Phi}$). PARGL (HARGL) stands for the left argument of the premise (hypothesis); PARGR (HARGR) for the right one.

# Chapter 4

# Continuous Entailment Patterns for Lexical Inference in Context

# Continuous Entailment Patterns for Lexical Inference in Context

**Martin Schmitt** and **Hinrich Schütze**
Center for Information and Language Processing (CIS)
LMU Munich, Germany
martin@cis.lmu.de

## Abstract

Combining a pretrained language model (PLM) with textual patterns has been shown to help in both zero- and few-shot settings. For zero-shot performance, it makes sense to design patterns that closely resemble the text seen during self-supervised pretraining because the model has never seen anything else. Supervised training allows for more flexibility. If we allow for tokens outside the PLM's vocabulary, patterns can be adapted more flexibly to a PLM's idiosyncrasies. Contrasting patterns where a "token" can be any continuous vector vs. those where a discrete choice between vocabulary elements has to be made, we call our method *CONtinuous pAtterNs* (CONAN). We evaluate CONAN on two established benchmarks for lexical inference in context (LIiC) a.k.a. predicate entailment, a challenging natural language understanding task with relatively small training sets. In a direct comparison with discrete patterns, CONAN consistently leads to improved performance, setting a new state of the art. Our experiments give valuable insights into the kind of pattern that enhances a PLM's performance on LIiC and raise important questions regarding our understanding of PLMs using text patterns.[1]

## 1 Introduction

Lexical inference in context (LIiC) – also called predicate entailment – is a variant of natural language inference (NLI) or recognizing textual entailment (Dagan et al., 2013) with focus on the lexical semantics of verbs and verbal expressions (Levy and Dagan, 2016; Schmitt and Schütze, 2019). Its goal is to detect entailment between two very similar sentences, i.e., sentences that share subject and object and only differ in the predicate, e.g., PERSON($A$) *runs* ORG($B$) → PERSON($A$) *leads* ORG($B$). NLI models that were not specifically trained with lexical knowledge have been reported to struggle with this task (Glockner et al., 2018; Schmitt and Schütze, 2019), making LIiC an important evaluation criterion for general language understanding. Other use cases for this kind of lexical entailment knowledge include question answering (Schoenmackers et al., 2010; McKenna et al., 2021), event coreference (Shwartz et al., 2017; Meged et al., 2020), and link prediction in knowledge graphs (Hosseini et al., 2019).

Although LIiC is an inherently directional task, symmetric cosine similarity in a vector space, such as word2vec (an, 2013), has long been the state of the art for this task. Only recently transfer learning with pretrained Transformer (Vaswani et al., 2017) language models (Devlin et al., 2019), has led to large improvements for LIiC. Schmitt and Schütze (2021) combine natural language (NL) patterns with a pretrained language model (PLM) and not only set a new state of the art but also beat baselines without access to such patterns.

Empirical findings suggest that a good pattern can be worth 100s of labeled training instances (Le Scao and Rush, 2021), making pattern approaches interesting for low-resource tasks such as LIiC. But beyond the intuition that patterns serve as some sort of task instruction (Schick and Schütze, 2021a), little is known about the reasons for their success. Recent findings that (i) PLMs can fail to follow even simple instructions (Efrat and Levy, 2020), that (ii) PLMs can behave drastically different with paraphrases of the same pattern (Elazar et al., 2021), and that (iii) performance increases if we train a second model to rewrite an input pattern with the goal of making it more comprehensible for a target PLM (Haviv et al., 2021), strongly suggest that patterns do not make sense to PLMs in the same way as they do to humans.

Our work sheds light on the interaction of patterns and PLMs and proposes a new method of improving pattern-based models fully automatically.

[1] Our code is publicly available: https://github.com/mnschmit/conan



*Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6952–6959
November 7–11, 2021. ©2021 Association for Computational Linguistics

On two popular LIiC benchmarks, our model (i) establishes a new state of the art without the need for handcrafting patterns or automatically identifying them in a corpus and (ii) does so more efficiently thanks to shorter patterns. Our best model only uses 2 tokens per pattern.

## 2 The CONAN Model

**Continuous patterns.** LIiC is a binary classification task, i.e., given a premise $\mathbf{p} = p_1 p_2 \dots p_{|\mathbf{p}|}$ and a hypothesis $\mathbf{h} = h_1 h_2 \dots h_{|\mathbf{h}|}$ a model has to decide whether $\mathbf{p}$ entails $\mathbf{h}$ ($y = 1$) or not ($y = 0$). A template-based approach to this task surrounds $\mathbf{p}$ and $\mathbf{h}$ with tokens $t_1 t_2 \dots t_m$ to bias the classifier for entailment detection, e.g., "$\mathbf{p}$, *which means that* $\mathbf{h}$".While in most approaches that leverage a PLM $\mathcal{M}$ these tokens come from the PLM's vocabulary, i.e., $\forall i.\, t_i \in \Sigma_{\mathcal{M}}$, we propose a model based on CONtinuous pAtterNs (CONAN), i.e., surround the embedding representation of $\mathbf{p}$ and $\mathbf{h}$ with continuous vectors that may be close to but do not have to match the embedding of any vocabulary entry.

For this, we first extend the PLM's vocabulary by a finite set $C = \{ c_1, c_2, \dots, c_{|C|} \}$ of fresh tokens, i.e., $\Sigma = \Sigma_{\mathcal{M}} \cup C$ with $C \cap \Sigma_{\mathcal{M}} = \emptyset$. Then, we distinguish two methods of surrounding $\mathbf{p}$ and $\mathbf{h}$ with these special tokens: $\alpha$ sets them both $\alpha$round and between $\mathbf{p}$ and $\mathbf{h}$ (Eq. (1)) while $\beta$ only sets them $\beta$etween the two input sentences (Eq. (2)).

$$\alpha_k(\mathbf{p}, \mathbf{h}) = c_1 \dots c_a \mathbf{p} c_{a+1} \dots c_{a+b} \mathbf{h} c_{a+b+1} \dots c_k$$
$$\text{with } a = \lfloor k/3 \rfloor,\ b = \lfloor k/3 \rfloor + (k \bmod 3) \quad (1)$$

$$\beta_k(\mathbf{p}, \mathbf{h}) = \mathbf{p} c_1 \dots c_k \mathbf{h} \quad (2)$$

Note that $\alpha$ divides its $k$ tokens into three parts as equally as possible where any remaining tokens go between $\mathbf{p}$ and $\mathbf{h}$ if $k$ is not a multiple of 3. In particular, this means that the same templates are produced by $\alpha$ and $\beta$ for $k \leq 2$. We chose this behavior as a generalization of the standard approach to fine-tuning a PLM for sequence classification (such as NLI) where there is only one special token and it separates the two input sequences. The template produced by $\alpha_1$ and $\beta_1$ is very similar to this. A major difference is that the embeddings for $C$ tokens are randomly initialized whereas the standard separator token has a pretrained embedding.

**Pattern-based classifier.** Given $\gamma \in \{ \alpha, \beta \}$, we estimate the probability distribution $P(\hat{y} \mid \mathbf{p}, \mathbf{h})$ with a linear classifier on top of the pooled sequence representation produced by the PLM $\mathcal{M}$:

$$P(\hat{y} \mid \mathbf{p}, \mathbf{h}) = \sigma(\mathcal{M}(\gamma(\mathbf{p}, \mathbf{h}))W + b) \quad (3)$$

where $W \in \mathbb{R}^{d \times 2}, b \in \mathbb{R}^2$ are learnable parameters, $\sigma$ is the softmax function, and applying $\mathcal{M}$ means encoding the whole input sequence in a single $d$-dimensional vector according to the specifics of the PLM. For BERT (Devlin et al., 2019) and its successor RoBERTa (Liu et al., 2019), this implies a dense pooler layer with tanh activation over the contextualized token embeddings and picking the first of these embeddings (i.e., [CLS] for BERT and ⟨s⟩ for RoBERTa).[2] For training, we apply dropout with a probability of 0.1 to the output of $\mathcal{M}(\cdot)$.

**Inference with multiple patterns.** Previous work (Bouraoui et al., 2020; Schmitt and Schütze, 2021) combined multiple patterns with the intuition that different NL patterns can capture different aspects of the task. This intuition makes also sense for CONAN. We conjecture that an efficient use of the model parameters requires different continuous patterns to learn different representations, which can detect different types of entailment. Following the aforementioned work, we form our final score $s$ by combining the probability estimates from different patterns $\Gamma$ by comparing the maximum probability for the two classes $0, 1$ over all patterns:

$$m_1(\mathbf{p}, \mathbf{h}) = \max_{\gamma \in \Gamma} P(\hat{y} = 1 \mid \gamma(\mathbf{p}, \mathbf{h}))$$
$$m_0(\mathbf{p}, \mathbf{h}) = \max_{\gamma \in \Gamma} P(\hat{y} = 0 \mid \gamma(\mathbf{p}, \mathbf{h}))$$
$$s(\mathbf{p}, \mathbf{h}) = m_1(\mathbf{p}, \mathbf{h}) - m_0(\mathbf{p}, \mathbf{h}) \quad (4)$$

In conclusion, a CONAN model $\gamma_k^n$ is characterized by three factors: (i) The type of pattern $\gamma \in \{ \alpha, \beta \}$, (ii) the number of patterns $n \in \mathbb{N}$, and (iii) the number of tokens $k \in \mathbb{N}$ per pattern.

**Training.** While multiple patterns are combined for decision finding during inference, we treat all patterns separately during training – as did previous work (Schmitt and Schütze, 2021). So, given a set of patterns $\Gamma$, we minimize the negative log-likelihood of the training data $\mathcal{T}$, i.e.,

$$\sum_{(\mathbf{p}, \mathbf{h}, y) \in \mathcal{T}} \sum_{\gamma \in \Gamma} -\log(P(\hat{y} = y \mid \gamma(\mathbf{p}, \mathbf{h}))) \quad (5)$$

In practice, we apply mini-batching to both $\mathcal{T}$ and $\Gamma$ and thus compute this loss only for a fraction of the available training data and patterns at a time. In

[2] Cf. Jacob Devlin's comment on issue 43 in the official BERT repository on GitHub, https://github.com/google-research/bert/issues/43.

| | train | dev | test | total |
|---|---|---|---|---|
| SherLIiC | 797 | 201 | 2,990 | 3,988 |
| Levy/Holt | 4,388 | 1,098 | 12,921 | 18,407 |

Table 1: Number of labeled instances per data split.

this case, we normalize the loss by averaging over the training samples and patterns in the mini-batch.

## 3 Experiments

We conduct experiments on two established LIiC benchmarks, SherLIiC (Schmitt and Schütze, 2019) and Levy/Holt (Levy and Dagan, 2016; Holt, 2018), using the data splits as defined in (Schmitt and Schütze, 2021) for comparison. Both benchmarks contain a majority of negative examples (SherLIiC: 67%, Levy/Holt: 81%), making the detection of the entailment (i.e., the minority) class a particular challenge. See Table 1 for dataset and split sizes. Note that Levy/Holt is nearly 5 times bigger than SherLIiC and still has less than 5k train samples.

Following (Schmitt and Schütze, 2021), we use RoBERTa as underlying PLM and also use the same hyperparameters whenever possible for comparison. Also following Schmitt and Schütze (2021), we instantiate the typed placeholders $A, B$ in SherLIiC with Freebase (Bollacker et al., 2008) entities, making sure that $A$ and $B$ are not assigned the same entity. See Appendix A for full training details.

We evaluate model performance with two metrics: (i) The area under the precision-recall curve for precision values $\geq 0.5$ (AUC) as threshold-less metric using only the score $s$ defined in the previous section and (ii) the F1 score of actual classification decisions after tuning a decision threshold $\vartheta$ on the respective dev portion of the data. Our implementation is based on (Wolf et al., 2019).

## 4 Results

**Choosing $n$ and $k$.** The number $n$ of patterns and the number $k$ of $C$ tokens per pattern are essential hyperparameters of a CONAN model.

Fig. 1 shows the impact on performance (measured in AUC) on SherLIiC dev for different $n$-$k$-combinations. We observe that using too many, too long patterns harms performance w.r.t. the base case $n = k = 1$. Best results are obtained with either a small number of patterns or tokens or both. Comparing the $\alpha$ and $\beta$ settings, we notice that,

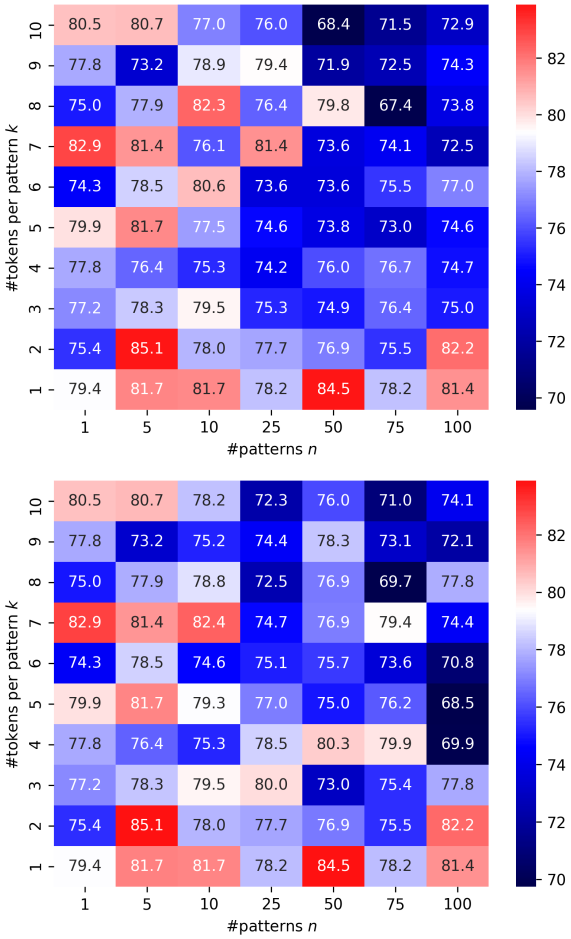



Figure 1: AUC on SherLIiC dev for different CONAN models; top = $\alpha$, bottom = $\beta$; white/red/blue = similar to/better than/worse than $n = k = 1$. Note that $\alpha_k = \beta_k$ for $k \leq 2$.

rounded to one decimal, they produce identical results for both $n = 1$ and $n = 5$ patterns, suggesting that the particular position of the $C$ tokens does not matter much in these settings for SherLIiC. Even with $n = 10$ patterns, the two methods only begin to differ with $k \geq 5$ tokens per pattern. Evaluation on the Levy/Holt data (see Fig. 2 in Appendix B) shows more variation between $\alpha$ and $\beta$ but, otherwise, confirms the trend that small $n$ and $k$ yield better performance.

Our results offer an explanation for the empirical finding in (Schmitt and Schütze, 2021) that patterns retrieved from a corpus lead to worse performance than handcrafted ones because the latter are generally shorter. CONAN models do not only yield better performance, they also provide an automatic way to test pattern properties, such as length, w.r.t. effect on performance for a given task.

**Test performance.** On both SherLIiC (Table 2)

| | | AUC | P | R | F1 |
|---|---|---|---|---|---|
| RoBERTa-base | | | | | |
| MANPAT$^{\Phi}$ | (baseline) | 69.2 | 62.0 | 81.2 | 70.3 |
| AUTPAT$^{\Phi}_{15}$ | (baseline) | 73.1 | 63.0 | 77.4 | 69.4 |
| CONAN$^{5}_{2}$ | ($\vartheta = -0.0768$) | 73.2 | 65.8 | **84.1** | **73.8** |
| CONAN$^{50}_{1}$ | ($\vartheta = -0.0108$) | **74.0** | **70.1** | 78.0 | **73.8** |
| CONAN-$\alpha^{1}_{7}$ | ($\vartheta = -0.1875$) | 66.9 | 68.7 | 68.4 | 68.5 |
| CONAN-$\beta^{1}_{7}$ | ($\vartheta = -0.1875$) | 66.9 | 68.7 | 68.4 | 68.5 |
| RoBERTa-large | | | | | |
| NLI | (baseline) | 68.3 | 60.5 | 85.5 | 70.9 |
| MANPAT$^{\Phi\Psi}$ | (baseline) | 74.4 | 66.0 | 80.8 | 72.6 |
| CONAN$^{5}_{2}$ | ($\vartheta = -0.6838$) | **75.9** | **67.9** | 81.1 | **73.9** |
| CONAN$^{50}_{1}$ | ($\vartheta = -0.9999$) | 73.9 | 64.8 | 81.5 | 72.2 |
| CONAN-$\alpha^{1}_{7}$ | ($\vartheta = -0.8797$) | 62.6 | 63.1 | 76.4 | 69.1 |
| CONAN-$\beta^{1}_{7}$ | ($\vartheta = -0.9556$) | 68.6 | 60.8 | **86.0** | 71.2 |

Table 2: SherLIiC test. AUC denotes the area under the precision-recall curve for precision $\geq 0.5$. All results in %. Bold means best result per column and block. All baselines from (Schmitt and Schütze, 2021). For $k \leq 2$, we simply write CONAN$^{n}_{k}$ because $\alpha = \beta$.

| | | AUC | P | R | F1 |
|---|---|---|---|---|---|
| RoBERTa-base | | | | | |
| NLI | (baseline) | 72.6 | 68.7 | 75.3 | 71.9 |
| MANPAT$^{\Phi\Psi}$ | (baseline) | 76.9 | 78.7 | 66.4 | 72.0 |
| CONAN$^{5}_{2}$ | ($\vartheta = -0.4809$) | **77.6** | **79.1** | 67.5 | **72.8** |
| CONAN$^{5}_{1}$ | ($\vartheta = -0.9003$) | 74.9 | 67.1 | **77.5** | 71.9 |
| CONAN-$\alpha^{5}_{3}$ | ($\vartheta = -0.8985$) | 73.6 | 75.0 | 66.7 | 70.6 |
| CONAN-$\beta^{5}_{3}$ | ($\vartheta = -0.9289$) | 76.4 | 76.6 | 69.2 | 72.7 |
| RoBERTa-large | | | | | |
| MANPAT$^{\Phi\Psi}$ | (baseline) | 83.9 | 84.8 | 70.1 | 76.7 |
| MANPAT$^{\Phi}$ | (baseline) | 77.8 | 67.9 | **81.5** | 74.1 |
| CONAN$^{5}_{2}$ | ($\vartheta = -0.1315$) | **85.9** | 81.7 | 78.1 | **79.9** |
| CONAN$^{5}_{1}$ | ($\vartheta = -0.9750$) | 85.2 | 77.2 | 80.3 | 78.7 |
| CONAN-$\alpha^{5}_{3}$ | ($\vartheta = -0.9212$) | 84.4 | **82.2** | 74.9 | 78.4 |
| CONAN-$\beta^{5}_{3}$ | ($\vartheta = -0.9585$) | 85.3 | 78.8 | 77.3 | 78.0 |

Table 3: Levy/Holt test. All baselines from (Schmitt and Schütze, 2021). See Table 2 for table format.

and Levy/Holt (Table 3), and across model sizes (base and large), CONAN$^{5}_{2}$ (using either $\alpha$ or $\beta$ because they are identical for $k = 2$) outperforms all other models including the previous state of the art by Schmitt and Schütze (2021), who fine-tune RoBERTa both without patterns (NLI) and using handcrafted (MANPAT) or automatically retrieved corpus patterns (AUTPAT). We report their two best systems for each benchmark.

We take the performance increase with continuous patterns as a clear indicator that the flexibility offered by separating pattern tokens from the rest of the vocabulary allows RoBERTa to better adapt to the task-specific data even with only few labeled training instances in the challenging LIiC task.

| | AUC | P | R | F1 |
|---|---|---|---|---|
| SherLIiC train → Levy/Holt test | | | | |
| Schm&Schü (2021) (base) | 38.4 | **52.7** | 57.1 | 54.8 |
| Schm&Schü (2021) (large) | **70.4** | 39.6 | 95.3 | **56.0** |
| CONAN$^{5}_{2}$ (base) | 54.3 | 38.2 | 89.8 | 53.5 |
| CONAN$^{5}_{2}$ (large) | 61.5 | 34.5 | **96.3** | 50.8 |
| Levy/Holt train → SherLIiC test | | | | |
| Schm&Schü (2021) (base) | 63.3 | 62.8 | 68.4 | 65.5 |
| Schm&Schü (2021) (large) | 62.1 | 68.1 | 57.3 | 62.3 |
| CONAN$^{5}_{2}$ (base) | 64.9 | 63.4 | 68.9 | 66.1 |
| CONAN$^{5}_{2}$ (large) | **70.1** | **69.0** | **69.5** | **69.2** |

Table 4: Transfer experiments ($\vartheta = 0$). Best models from (Schmitt and Schütze, 2021) according to F1 score.

## 5 Analysis and Discussion

**Nearest neighbors.** To further investigate how RoBERTa makes use of the flexibility of $C$ tokens, we compute their nearest neighbors in the space of original vocabulary tokens based on cosine similarity for our models in Tables 2 and 3. We always find the $C$ tokens to be very dissimilar from any token in the original vocabulary, the highest cosine similarity being 0.15. And even among themselves, $C$ tokens are very dissimilar, nearly orthogonal, with 0.08 being the highest cosine similarity here. RoBERTa seems to indeed take full advantage of the increased flexibility to put the $C$ tokens anywhere in the embedding space. This further backs our hypothesis that the increased flexibility is beneficial for performance.

**Influence of additional parameters.** One might argue that the vocabulary extension and the resulting new randomly initialized token embeddings lead to an unfair advantage for CONAN models because the parameter count increases. While more parameters do generally lead to increased model capacity, the number of new parameters is so small compared to the total number of parameters in RoBERTa that we consider it improbable that the new parameters are alone responsible for the improved performance. Of all models in Tables 2 and 3, CONAN$^{50}_{1}$ introduces the most additional model parameters, i.e., $1 \cdot 50 \cdot 768 = 38400$ for RoBERTa-base. Given that even the smaller RoBERTa-base model still has a total of 125M parameters, the relative parameter increase is maximally 0.03%, which, we argue, is negligible.

**Transfer between datasets.** The experiments summarized in Table 4 investigate the hypothesis that CONAN's better adaptation to the fine-tuning data might worsen its generalization abilities to other

LIiC benchmarks. For this, we train our best model CONAN$_2^5$ on SherLIiC to test it on Levy/Holt and vice versa. In this scenario, we assume that the target dataset is not available at all. So there is no way to adapt to a slightly different domain other than learning general LIiC reasoning. We thus set $\vartheta = 0$ in these experiments.

We find that with the very few train samples in SherLIiC the risk of overfitting to SherLIiC is indeed higher. When trained on Levy/Holt with around 4.4k train samples, however, CONAN clearly improves generalization to the SherLIiC domain.

## 6 Related Work

**PLMs and text patterns.** GPT-2 (Radford et al., 2019) made the idea popular that a PLM can perform tasks without access to any training data when prompted with the right NL task instructions. With GPT-3, Brown et al. (2020) adapted this idea to few-shot settings where the task prompt is extended by a few training samples. While this kind of few-shot adaptation with a frozen PLM only works with very big models, Schick and Schütze (2021b) achieve similar performance with smaller models by fine-tuning the PLM on the available training data and putting them into NL templates. Recently, Schmitt and Schütze (2021) investigated the use of PLMs for LIiC. Compared to a standard sequence classification fine-tuning approach, they were able to improve the PLM RoBERTa's performance by putting an entailment candidate into textual contexts that only make sense for either a valid or invalid example. Patterns like "$y$ because $x$." (valid) or "It does not mean that $y$ just because $x$." (invalid) make intuitive sense to humans and outperform standard RoBERTa on LIiC.

A large problem with all these approaches, however, is to find well-functioning patterns, for which numerous solutions have been proposed (Shin et al., 2020; Haviv et al., 2021; Bouraoui et al., 2020; Jiang et al., 2020; Gao et al., 2021; Reynolds and McDonell, 2021). We argue that it is not optimal to constrain pattern search to the space of NL sequences if the primary goal is better task performance, and therefore abandon this constraint.

**PLMs and continuous patterns.** Li and Liang (2021) and Hambardzumyan et al. (2021) contemporaneously introduced the idea of mixing the input token embeddings of a PLM with other continuous vectors that do not correspond to vocabulary elements. In the spirit of GPT-2 (see above), they keep the PLM's parameters frozen and only fine-tune the embeddings of the "virtual tokens" to the target task. While this line of research offers certain appeals of its own, e.g., reusability of the frozen PLM weights, this is not the focus of our work. In pursuit of the best possible performance, we instead compare the use of continuous vs. NL patterns in the process of fine-tuning all PLM parameters and find that even carefully chosen NL patterns can be outperformed by our automatically learned ones.

Contemporaneously to our work, Liu et al. (2021) fine-tune entire PLMs with continuous patterns for SuperGLUE (Wang et al., 2019). Besides reformulating the SuperGLUE tasks as cloze tasks, while we keep formalizing our task as classification, Liu et al. (2021) also add more complexity by computing the continuous token representations with an LSTM (Hochreiter and Schmidhuber, 1997) and adding certain "anchor tokens", such as a question mark, at manually chosen places. CONAN does not use any manual pattern design and embeds continuous tokens with a simple lookup table.

Another contemporaneous work by Lester et al. (2021) tests the influence of model size on the performance of a frozen PLM with trained continuous prompts. Their prompt ensembling is akin to our combining multiple patterns during inference (cf. §2). The key difference is that, instead of making predictions with different patterns and taking the majority vote, we rather compare the scores for different patterns to make our prediction.

## 7 Conclusion

We presented CONAN, a method that improves fine-tuning performance of a PLM with continuous patterns. CONAN does not depend on any manual pattern design and is efficient as the shortest possible patterns with good performance can be found automatically. It provides an automatic way of systematically testing structural properties of patterns, such as length, w.r.t. performance changes. In our experiments on two established LIiC benchmarks, CONAN outperforms previous work using NL patterns and sets a new state of the art.

## Acknowledgments

We gratefully acknowledge a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes). This work was supported by the BMBF as part of the project MLWin (01IS18050).

## References


Tomas Mikolov an. 2013. Efficient estimation of word representations in vector space. *ArXiv preprint*, abs/1301.3781.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7456–7463. AAAI Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing textual entailment: Models and applications*. Morgan & Claypool Publishers.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *ArXiv preprint*, abs/2010.11982.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *ArXiv preprint*, abs/2102.01017.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. BERTese: Learning to speak to BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3618–3623, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Xavier R. Holt. 2018. Probabilistic models of relational implication. Master's thesis, Macquarie University.

Mohammad Javad Hosseini, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2019. Duality of link prediction and entailment graph induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4736–4746, Florence, Italy. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *ArXiv preprint*, abs/2104.08691.

Omer Levy and Ido Dagan. 2016. Annotating relation inference in context via question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 249–255, Berlin, Germany. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *ArXiv preprint*, abs/2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Nick McKenna, Liane Guillou, Mohammad Javad Hosseini, Sander Bijl de Vroe, and Mark Steedman. 2021. Multivalent entailment graphs for question answering. *ArXiv preprint*, abs/2104.07846.

Yehudit Meged, Avi Caciularu, Vered Shwartz, and Ido Dagan. 2020. Paraphrasing vs coreferring: Two sides of the same coin. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4897–4907, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI, San Francisco, California, United States.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. *ArXiv preprint*, abs/2102.07350.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Martin Schmitt and Hinrich Schütze. 2019. SherLIiC: A typed event-focused lexical inference benchmark for evaluating natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 902–914, Florence, Italy. Association for Computational Linguistics.

Martin Schmitt and Hinrich Schütze. 2021. Language models for lexical inference in context. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1267–1280, Online. Association for Computational Linguistics.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098, Cambridge, MA. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Vered Shwartz, Gabriel Stanovsky, and Ido Dagan. 2017. Acquiring predicate paraphrases from news tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 155–160, Vancouver, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771.

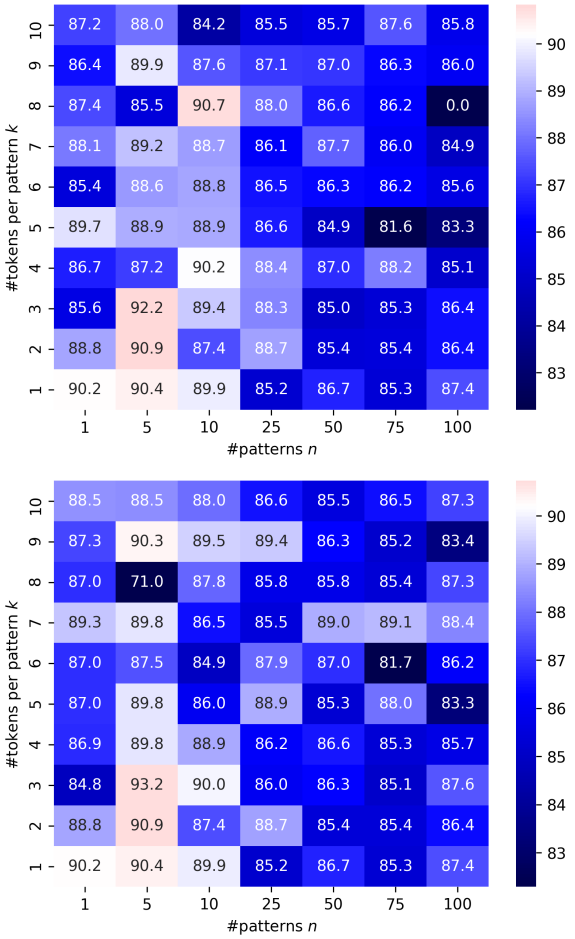



Figure 2: AUC on Levy/Holt dev for different CONAN models. Same format as Fig. 1.

## A Training Details

We train our model for 5 epochs on a single GeForce RTX 2080 Ti GPU, with Adam (Kingma and Ba, 2015) and a mini-batch size of 10 (resp. 2) training instances for RoBERTa-base (resp. -large) and mini-batches of 5 patterns. We adopt well-functioning values from (Schmitt and Schütze, 2021) for all non-CONAN-specific hyperparameters, i.e., learning rate $lr$, weight decay $\lambda$, and accumulated batches $c$ before a gradient update: Concretely, we set $lr = 2.28 \cdot 10^{-5}, \lambda = 6.52 \cdot 10^{-2}, c = 2$ for evaluating RoBERTa-base on SherLIiC, $lr = 1.29 \cdot 10^{-5}, \lambda = 2.49 \cdot 10^{-4}, c = 3$ for RoBERTa-large on SherLIiC, $lr = 2.72 \cdot 10^{-5}, \lambda = 1.43 \cdot 10^{-3}, c = 1$ for RoBERTa-base on Levy/Holt, and $lr = 4.55 \cdot 10^{-6}, \lambda = 3.90 \cdot 10^{-4}, c = 2$ for RoBERTa-large on Levy/Holt.

## B More Dev Results

See Fig. 2 for evaluation results on Levy/Holt dev.

# Chapter 5

# An Unsupervised Joint System for Text Generation from Knowledge Graphs and Semantic Parsing

# An Unsupervised Joint System for Text Generation from Knowledge Graphs and Semantic Parsing

**Martin Schmitt**[1] **Sahand Sharifzadeh**[2] **Volker Tresp**[2,3] **Hinrich Schütze**[1]

[1]Center for Information and Language Processing (CIS), LMU Munich
[2]Department of Informatics, LMU Munich
[3]Siemens AG Munich
martin@cis.lmu.de

## Abstract

Knowledge graphs (KGs) can vary greatly from one domain to another. Therefore supervised approaches to both graph-to-text generation and text-to-graph knowledge extraction (semantic parsing) will always suffer from a shortage of domain-specific parallel graph-text data; at the same time, adapting a model trained on a different domain is often impossible due to little or no overlap in entities and relations. This situation calls for an approach that (1) does not need large amounts of annotated data and thus (2) does not need to rely on domain adaptation techniques to work well in different domains. To this end, we present the *first approach to unsupervised text generation from KGs* and show simultaneously how it can be used for *unsupervised semantic parsing*. We evaluate our approach on WebNLG v2.1 and a new benchmark leveraging scene graphs from Visual Genome. Our system outperforms strong baselines for both text↔graph conversion tasks without any manual adaptation from one dataset to the other. In additional experiments, we investigate the impact of using different unsupervised objectives.[1]

## 1 Introduction

Knowledge graphs (KGs) are a general-purpose approach for storing information in a structured, machine-accessible way (Van Harmelen et al., 2008). They are used in various fields and domains to model knowledge about topics as different as lexical semantics (Fellbaum, 2005; van Assem et al., 2006), common sense (Speer et al., 2017; Sap et al., 2019), biomedical research (Wishart et al., 2018) and visual relations in images (Lu et al., 2016).

This ubiquity of KGs necessitates interpretability because diverse users – both experts and non-experts – work with them. Even though, in principle, a KG is human-interpretable, non-experts may have difficulty making sense of it. Thus, there is a need for methods, such as automatic natural language generation ("graph→text"), that support them.

Semantic parsing, i.e., the conversion of a text to a formal meaning representation, such as a KG, ("text→graph") is equally important because it makes information that only exists in text form accessible to machines, thus assisting knowledge base engineers in KG creation and completion.

As KGs are so flexible in expressing various kinds of knowledge, separately created KGs vary a lot. This unavoidably leads to a shortage of training data for both graph↔text tasks. We therefore propose an unsupervised model that (1) easily adapts to new KG domains and (2) only requires unlabeled (i.e., non-parallel) texts and graphs from the target domain, together with a few fact extraction heuristics, but no manual annotation.

To show the effectiveness of our approach, we conduct experiments on the latest release (v2.1) of the WebNLG corpus (Shimorina and Gardent, 2018) and on a new benchmark we derive from *Visual Genome* (Krishna et al., 2016). While both of these datasets contain enough annotations to train supervised models, we evaluate our unsupervised approach by ignoring these annotations. The datasets are particularly well-suited for our evaluation as both graphs and texts are completely human-generated. Thus for both our tasks, models are evaluated with natural, i.e., human-generated targets.

Concretely, we make the following contributions: (1) We present the first unsupervised non-template approach to text generation from KGs (graph→text). (2) We jointly develop a new unsupervised approach to semantic parsing that automatically adjusts to a target KG schema (text→graph). (3) In contrast to prior unsupervised graph→text and text→graph work, our model does not re-

[1]https://github.com/mnschmit/unsupervised-graph-text-conversion



*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7117–7130,
November 16–20, 2020. ©2020 Association for Computational Linguistics

quire manual adaptation to new domains or graph schemas. (4) We provide a thorough analysis of the impact of different unsupervised objectives, especially the ones we newly introduce for text↔graph conversion. (5) We create a new large-scale dataset for text↔graph transformation tasks in the visual domain.

## 2 Related Work

**graph → text.** Our work is the first attempt at fully unsupervised text generation from KGs. In this respect it is only comparable to traditional rule- or template-based approaches (Kukich, 1983; McRoy et al., 2000). However, in contrast to these approaches, which need to be manually adapted to new domains and KG schemas, our method is generally applicable to all kinds of data without modification.

There is a large body of literature about supervised text generation from structured data, notably about the creation of sports game summaries from statistical records (Robin, 1995; Tanaka-Ishii et al., 1998). Recent efforts make use of neural encoder-decoder mechanisms (Wiseman et al., 2017; Puduppully et al., 2019). Although text creation from relational databases is related and our unsupervised method is, in principle, also applicable to this domain, in our work we specifically address text creation from graph-like structures such as KGs.

One recent work on supervised text creation from KGs is (Bhowmik and de Melo, 2018). They generate a short description of an entity, i.e., a single KG node, based on a set of facts about the entity. We, however, generate a description of the whole KG, which involves multiple entities and their relations. Koncel-Kedziorski et al. (2019) generate texts from whole KGs. They, however, do not evaluate on human-generated KGs but automatically generated ones from the scientific information extraction tool SciIE (Luan et al., 2018). Their supervised model is based on message passing through the topology of the incidence graph of the KG input. Such graph neural networks (Kipf and Welling, 2017; Veličković et al., 2018) have been widely adopted in supervised graph-to-text tasks (Beck et al., 2018; Damonte and Cohen, 2019; Ribeiro et al., 2019, 2020).

Even though Marcheggiani and Perez-Beltrachini (2018) report that graph neural networks can make better use of graph input than RNNs for supervised learning, for our unsupervised approach we follow the line of research that uses RNN-based sequence-to-sequence models (Cho et al., 2014; Sutskever et al., 2014) operating on serialized triple sets (Gardent et al., 2017b; Trisedya et al., 2018; Gehrmann et al., 2018; Castro Ferreira et al., 2019; Fan et al., 2019). We make this choice because learning a common semantic space for both texts and graphs by means of a shared encoder and decoder is a central component of our model. It is a nontrivial, separate research question whether and how encoder-decoder parameters can effectively be shared for models working on both sequential and non-sequential data. We thus leave the adaptation of our approach to graph neural networks for future work.

**text → graph.** Converting a text into a KG representation, our method is an alternative to prior work on open information extraction (Niklaus et al., 2018) with the advantage that the extractions, though trained without labeled data, automatically adjust to the KGs used for training. It is therefore also related to relation extraction in the unsupervised (Yao et al., 2011; Marcheggiani and Titov, 2016; Simon et al., 2019) and distantly supervised setting (Riedel et al., 2010; Parikh et al., 2015). However, these systems merely predict a single relation between two given entities in a single sentence, while we translate a whole text into a KG with potentially multiple facts.

Our text→graph task is therefore most closely related to semantic parsing (Kamath and Das, 2019), but we convert statements into KG facts whereas semantic parsing typically converts a question into a KG or database query. Poon and Domingos (2009) proposed the first unsupervised approach. They, however, still need an additional KG alignment step, i.e., are not able to directly adjust to the target KG. Other approaches overcome this limitation but only in exchange for the inflexibility of manually created domain-specific lexicons (Popescu et al., 2004; Goldwasser et al., 2011). Poon (2013)'s approach is more flexible but still relies on preprocessing by a dependency parser, which generally means that language-specific annotations to train such a parser are needed. Our approach is end-to-end, i.e., does not need any language-specific preprocessing during inference and only depends on a POS tagger used in the rule-based text→graph system to bootstrap training.

**Unsupervised sequence generation.** Our unsu-

pervised training regime for both text↔graph tasks is inspired by (Lample et al., 2018b). They used self-supervised pretraining and backtranslation for unsupervised translation from one language to another. We adapt these principles and their noise model to our tasks, and introduce two new noise functions specific to text↔graph conversion.

## 3 Preliminaries

### 3.1 Data structure

We formalize a KG as a labeled directed multigraph $(V, E, s, t, l)$ where entities are nodes $V$ and edges $E$ represent relations between entities. The lookup functions $s, t : E \to V$ assign to each edge its source and target node. The labeling function $l$ assigns labels to nodes and edges where node labels are entity names and edge labels come from a predefined set $\mathcal{R}$ of relation types.

An equivalent representation of a KG is the set of its facts. A fact is a triple consisting of an edge's source node (the subject), the edge itself (the predicate), and its target node (the object). So the set of facts $\mathcal{F}$ of a KG can be obtained from its edges:

$$\mathcal{F} := \{\, (s(e), e, t(e)) \mid e \in E \,\}.$$

Applying $l$ to all triple elements and writing out $\mathcal{F}$ in an arbitrary order generates a serialization that makes the KG accessible to sequence models otherwise used only for text. This has the advantage that we can train a sequence encoder to embed text and KGs in the same semantic space. Specifically, we serialize a KG by writing out its facts separated with end-of-fact symbols (`EOF`) and elements of each fact with special `SEP` symbols. We thus define our task as a sequence-to-sequence (seq2seq) task.

### 3.2 Scene Graphs

The Visual Genome (VG) repository is a large collection of images with associated manually annotated **scene graphs**; see Fig. 1. A scene graph formally describes image objects with their attributes, e.g., (hydrant, `attr`, yellow), and their relations to other image objects, e.g., (woman, in, shorts). Each scene graph is organized into smaller subgraphs, known as **region graphs**, representing a subpart of a more complex larger picture that is interesting on its own. Each region graph is associated with an English text, the **region description**. Texts and graphs were not automatically produced from each other, but were collected from crowdworkers who

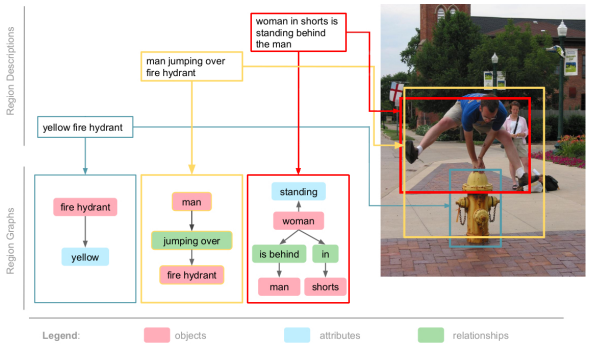



Figure 1: Region graphs and textual region descriptions in Visual Genome (VG). Image regions serve as common reference for text and graph creation but are disregarded in our work. We solely focus on the pairs of corresponding texts and graphs. Illustration adapted from (Krishna et al., 2016).

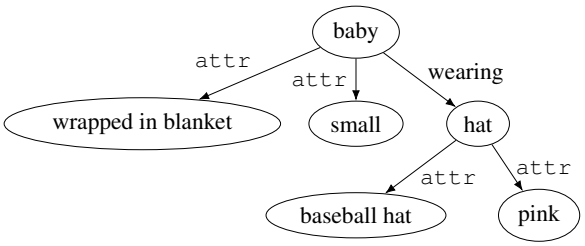



Figure 2: Example graph in our new VG benchmark.

were presented an image region and then generated text and graph. So although the graphs were not specifically designed to closely resemble the texts, they describe the same image region. This semantic correspondence makes scene graph↔text conversion an interesting and challenging problem because text and graph are not simple translations of each other.

Scene graphs are formalized in the same way as other KGs: $V$ here contains image objects and their attributes, and $\mathcal{R}$ contains all types of visual relationships and the special label `attr` for edges between attribute and non-attribute nodes. Fig. 2 shows an example.

VG scene graphs have been used before for traditional KG tasks, such as KG completion (Wan et al., 2018), but we are the first to use them for a text↔graph conversion dataset.

## 4 Approaches

### 4.1 Rule-based systems

We propose a rule-based system as unsupervised baseline for each of the text↔graph tasks. Note that they both assume that the texts are in English.

**R$^{\text{graph}\to\text{text}}$.** From a KG serialization, we remove

| noise function | behavior |
|---|---|
| `swap` | applies a random permutation $\sigma$ of words or facts with $\forall i \in \{1, \ldots, n\}, \lvert\sigma(i) - i\rvert \leq k$; $k = 3$ for text, $k = +\infty$ for knowledge graphs. |
| `drop` | removes each fact/word with a probability of $p_{\text{drop}}$. |
| `blank` | replaces each fact/word with a probability of $p_{\text{blank}}$ by a special symbol `blanked`. |
| `repeat` | inserts repetitions with a probability of $p_{\text{repeat}}$ in a sequence of facts/words. |
| `rule` | generates a noisy translation by applying $\text{R}^{\text{graph}\to\text{text}}$ to a graph or $\text{R}^{\text{text}\to\text{graph}}$ to a text. |

Table 1: Noise functions and their behavior on graphs and texts.

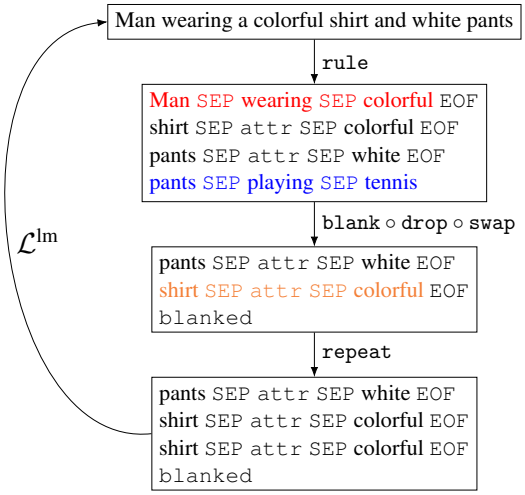



Figure 3: Example noisy training instance for the graph→text task in the composed noise setting. The fact highlighted in red is removed by `drop`, the one in blue is replaced with `blanked` by `blank`, the one in orange is repeated by `repeat`.

`SEP` symbols and replace `EOF` symbols by the word *and*. The special label `attr` is mapped to *is*. This corresponds to a template-based enumeration of all KG facts. See Table 5 for an example.

**$\text{R}^{\text{text}\to\text{graph}}$.** After preprocessing a text with NLTK's default POS tagger (Loper and Bird, 2004) and removing stop words, we apply two simple heuristics to extract facts: (1) Each verb becomes a predicate; *is* creates facts with predicate `attr`. The content words directly before and after such a predicate word become subject and object. (2) Adjectives $a$ form attributes, i.e., build facts of the form $(X, \texttt{attr}, a)$ where $X$ is filled with the first noun after $a$. These heuristics are similar in nature to a rudimentary parser. See Table 8 for an example.

### 4.2 Neural seq2seq systems

Our main system is a neural seq2seq architecture. We equip the standard encoder-decoder model with attention (Bahdanau et al., 2014) and copy mechanism (Gu et al., 2016). Allowing the model to directly copy from the source to the target side is beneficial in data to text generation (Wiseman et al., 2017; Puduppully et al., 2019). The encoder (resp. decoder) is a bidirectional (resp. unidirectional) LSTM (Hochreiter and Schmidhuber, 1997). Dropout (Hinton et al., 2012) is applied at the input of both encoder and decoder (Britz et al., 2017). We combine this model with the following concepts:

**Multi-task model.** In unsupervised machine translation, systems are trained for both translation directions (Lample et al., 2018b). In the same way, we train our system for both conversion tasks text↔graph, sharing encoder and decoder. To tell the decoder which type of output should be produced (text or graph), we initialize the cell state of the decoder with an embedding of the desired output type. The hidden state of the decoder is initialized with the last state of the encoder as usual.

**Noisy source samples.** Lample et al. (2018a) introduced denoising auto-encoding as pretraining and auxiliary task to train the decoder to produce well-formed output and make the encoder robust to noisy input. The training examples for this task consist of a noisy version of a sentence as source and the original sentence as target. We adapt this idea and propose the following noise functions for the domains of graphs and texts: `swap`, `drop`, `blank`, `repeat`, `rule`. Table 1 describes their behavior. `swap`, `drop` and `blank` are adapted from (Lample et al., 2018a) with facts in graphs taking the role of words in text. As order should be irrelevant in a set of facts, we drop the locality constraint in the `swap` permutation for graphs by setting $k = +\infty$.

Denoising samples generated by `repeat` requires the model to learn to remove redundant information in a set of facts. In the case of text, `repeat` mimics a behavior often observed with insufficiently trained neural models, i.e., repeating words considered important.

Unlike the other noise functions, `rule` does not "perturb" its input, but rather noisily backtranslates

it. We will see in Section 7 that bootstrapping with these noisy translations is essential.

We consider two fundamentally different noise injection regimes: (1) The **composed noise** setting is an adaptation of Lample et al. (2018a)'s noise model (`blank`∘`drop`∘`swap`) where our newly introduced noise functions `rule` and `repeat` are added to the start and end of the pipeline, i.e., all data samples are treated equally with the same noise function $C_{\text{comp}} :=$ `repeat`∘`blank`∘`drop`∘`swap`∘`rule`. Figure 3 shows an example. (2) In the **sampled noise** setting, we do not use all noise functions at once but sample a single one per data instance.

### 4.3 Training regimes

We denote the sets of graphs and corresponding texts by $\mathcal{G}$ and $\mathcal{T}$. The set of available supervised examples $(x, y) \in \mathcal{G} \times \mathcal{T}$ is called $\mathcal{S} \subset \mathcal{G} \times \mathcal{T}$. $P_g$ and $P_t$ are probabilistic models that generate, conditioned on any input, a graph ($g$) or a text ($t$).

**Unsupervised training.** We first obtain a language model for both graphs and text by training one epoch with the denoising auto-encoder objective:

$$\mathcal{L}^{\text{denoise}} = \mathop{\mathbb{E}}_{x\sim\mathcal{G}}[-\log P_g(x|C(x))] + \mathop{\mathbb{E}}_{y\sim\mathcal{T}}[-\log P_t(y|C(y))]$$

where $C \in \{C_{\text{comp}}\}$ for composed noise and $C \in \{\texttt{swap}, \texttt{blank}, \texttt{drop}, \texttt{repeat}, \texttt{rule}\}$ for sampled noise. In this pretraining epoch only, we use all possible noise functions individually on all available data. As sampled noise incorporates five different noise functions and composed noise only one, this results in five times more pretraining samples for sampled noise than for composed noise.

In subsequent epochs, we additionally consider $\mathcal{L}^{\text{back}}$ as training signal:

$$\mathcal{L}^{\text{back}} = \mathop{\mathbb{E}}_{x\sim\mathcal{G}}[-\log P_g(x|z^*(x))] + \mathop{\mathbb{E}}_{y\sim\mathcal{T}}[-\log P_t(y|w^*(y))]$$
$$z^*(x) = \arg\max_z P_t(z|x)$$
$$w^*(y) = \arg\max_w P_g(w|y)$$

This means that, in each iteration, we apply the current model to backtranslate a text (graph) to obtain a potentially imperfect graph (text) that we can use as noisy source with the clean original input being the target. This gives us a pseudo-parallel training instance for the next iteration – recall that we address unsupervised generation, i.e., without access to parallel data.

The total loss in these epochs is $\mathcal{L}^{\text{back}} + \mathcal{L}^{\text{denoise}}$, where now $\mathcal{L}^{\text{denoise}}$ only samples one possible type of noise independently for each data instance.

**Supervised training.** Our intended application is an unsupervised scenario. For our two datasets, however, we have labeled data (i.e., a "parallel corpus") and so can also compare our model to its supervised variant. Although supervised performance is generally better, it serves as a reference point and gives us an idea of the impact of supervision as opposed to factors like model architecture and hyperparameters. The supervised loss is simply defined as follows:

$$\mathcal{L}^{\text{sup}} = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{S}}[-\log P_t(y|x) - \log P_g(x|y)]$$

| | VG | $\text{VG}_{\text{ball}}$ | WebNLG |
|---|---|---|---|
| train split size | 2,412,253 | 151,790 | 34,338 |
| val split size | 323,478 | 21,541 | 4,313 |
| test split size | 324,664 | 20,569 | 4,222 |
| #relation types | 36,506 | 5,167 | 373 |
| avg #facts in graph | 2.7 | 2.5 | 3.0 |
| avg #tokens in text | 5.4 | 5.5 | 22.8 |
| avg % text tokens in graph | 49.3 | 50.6 | 49.4 |
| avg % graph tokens in text | 52.3 | 54.7 | 75.6 |

Table 2: Statistics of WebNLG v2.1 and our newly created benchmark VG; $\text{VG}_{\text{ball}}$ is a subset of VG representing images from ball sports events. Data split sizes are given as number of graph-text pairs.

## 5 Experiments

### 5.1 Data

For our experiments, we randomly split the VG images 80/10/10 into train/val/test. We then remove all graphs from train that also occur in one of the images in val or test. Finally, we unify graph serialization duplicates with different texts to single instances with multiple references for graph→text and proceed analogously with text duplicates for text→graph. For WebNLG v2.1, we use the data splits as provided. Following (Gardent et al., 2017a), we resolve the camel case of relation names and remove underscores from entity names in a preprocessing step. For both datasets, the order of facts in graph serializations corresponds to the order of triples in the original dataset. Because of VG's enormous size and limited computation power, we additionally create a closed-domain ball

| graph → text | Visual Genome | | | | | | WebNLG | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BLEU | | METEOR | | CHRF++ | | BLEU | | METEOR | | CHRF++ | |
| | val | test | val | test | val | test | val | test | val | test | val | test |
| $R^{graph \to text}$ | 5.9 | 5.9 | 28.2 | 28.1 | 43.4 | 43.3 | 18.3 | 18.3 | 33.5 | 33.6 | 55.0 | 55.2 |
| Ours w/ sampled noise | 19.8 | 19.5 | 31.4 | 31.2 | 50.9 | 50.7 | **39.1** | **37.7** | **35.4** | **35.5** | **61.9** | **62.1** |
| Ours w/ composed noise | **23.2** | **23.2** | **33.0** | **32.9** | **53.7** | **53.6** | 30.8 | 30.5 | 30.2 | 30.0 | 53.1 | 52.8 |
| Ours *supervised* | 26.5 | 26.4 | 32.3 | 32.2 | 53.7 | 53.6 | 35.1 | 34.4 | 39.6 | 39.5 | 64.1 | 64.0 |

Table 3: Results for unsupervised and supervised text generation. Note that training a supervised model on millions of labeled samples is usually not an option. Best unsupervised models are identified by best BLEU on $\mathcal{V}_{100}$. BLEU and METEOR are computed with scripts from (Lin et al., 2018); the CHRF++ script is from (Popović, 2017b).

sports subset of VG, called $VG_{ball}$, which we can use to quickly conduct additional experiments (see Section 7). We identify all images where at least one region graph contains at least one fact that mentions an object ending with *ball* and take all regions from them (keeping data splits the same). In contrast to alternatives like random subsampling, we consider this domain-focused construction more realistic.

Table 2 shows relevant statistics for all datasets. While VG and WebNLG have similar statistics, VG is around 70 times larger than WebNLG, which makes it an interesting benchmark for future research, both supervised and unsupervised. Apart from size, there are two important differences: (1) The VG graph schema has been freely defined by crowd workers and thus features a large variety of different relations. (2) The percentage of graph tokens occurring in the text, a measure important for the text→graph task, is lower for VG than for WebNLG. Thus, VG graphs contain more details than their corresponding texts, which is a characteristic feature of the domain of image captions: they mainly describe the salient image parts.

### 5.2 Training details

We train all models with the Adam optimizer (Kingma and Ba, 2015) for maximally 30 epochs. We stop supervised models early when $\mathcal{L}^{sup}$ does not decrease on val for 10 epochs. Unsupervised models are stopped after 5 iterations on VG because of its big size and limited computational resources. All hyperparameters and more details are described in Appendices A and B. Our implementation is based on AllenNLP (Gardner et al., 2017).

In unsupervised training, input graphs and texts are the same as in supervised training – only the gold target sides are ignored. While it is an artificial setup to split paired data and treat them as unpaired, this not only makes the supervised and unsupervised settings more directly comparable, but also ensures that the text data resemble the evaluation texts in style and domain. For the purpose of experiments on a benchmark, this seems appropriate to us. For a concrete use case, it would be an important first step to find adequate texts that showcase the desired language style and that are about a similar topic as the KGs that are to be textualized. As KGs are rarely the only means of storing information, e.g., in an industrial context, such texts should not be hard to come by in practice.

| # | sampled noise | | | | composed noise | | | |
|---|---|---|---|---|---|---|---|---|
| | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val | test | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val | test |
| 1 | **80.4** | 7.8 | 10.1 | 9.9 | **72.2** | 15.9 | 19.8 | 19.7 |
| 2 | 50.7 | 7.2 | 9.2 | 9.1 | 41.2 | 14.0 | 15.2 | 15.1 |
| 3 | 67.6 | 19.5 | 19.4 | 19.2 | 61.0 | 22.7 | **23.5** | **23.4** |
| 4 | 56.4 | **21.2** | **19.8** | **19.5** | 51.9 | 22.2 | 21.4 | 21.3 |
| 5 | 62.9 | 20.0 | 19.6 | 19.4 | 60.5 | **24.5** | 23.2 | 23.2 |

Table 4: BLEU scores on VG for our unsupervised models evaluated for graph→text at different iterations. $\mathcal{U}$ is calculated on all unlabeled data used for training. $\mathcal{V}_{100}$ is a 100-size random sample from val. All results are computed with scripts from (Lin et al., 2018).

## 6 Results and Discussion

### 6.1 Text generation from graphs

**Model selection.** Table 4 shows how performance of our unsupervised model changes at every back-translation iteration, measured in BLEU (Papineni et al., 2002), a common metric for natural language generation. For model selection, we adopt the two methods proposed by Lample et al. (2018b), i.e., a small validation set (we take a 100-size random subset of val, called $\mathcal{V}_{100}$) and a fully unsupervised criterion ($\mathcal{U}$) where BLEU compares an unlabeled sample with its back-and-forth translation. We confirm their finding that $\mathcal{U}$ is not reliable for neural

| | |
|---|---|
| (a) Reference text | a baseball cap on a baby's head |
| (b) $R^{graph \to text}$ | baby is small and baby is wrapped in blanket and hat is pink and hat is baseball hat and baby wearing hat |
| (c) Unsuperv. neural model | small baby wrapped in blanket with pink baseball hat |
| (d) Superv. neural model | baby wearing a pink hat |

Table 5: Texts generated from graph in Fig. 2.

| | sampled noise | | | | composed noise | | | |
|---|---|---|---|---|---|---|---|---|
| # | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val | test | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val | test |
| 1 | 19.1 | 1.0 | 1.2 | 1.2 | 17.0 | 2.0 | 2.2 | 2.2 |
| 2 | **71.0** | **21.7** | **19.1** | **18.8** | 49.3 | **22.1** | **22.1** | **21.7** |
| 3 | 58.2 | 19.3 | 18.6 | 18.3 | 45.9 | 18.7 | 19.7 | 19.4 |
| 4 | 62.3 | 18.3 | **19.1** | **18.8** | **54.4** | 19.9 | 20.8 | 20.5 |
| 5 | 63.7 | 19.8 | 19.0 | 18.7 | 49.0 | 18.8 | 19.0 | 18.8 |

Table 6: F1 scores on VG for our models from Table 4 evaluated on text→graph at different iterations.

| text → graph | VG | | WebNLG | |
|---|---|---|---|---|
| | val | test | val | test |
| $R^{text \to graph}$ | 13.4 | 13.1 | 0.0 | 0.0 |
| Stanford SG Parser | 19.5 | 19.3 | 0.0 | 0.0 |
| Ours w/ sampled noise | 19.1 | 18.8 | **38.5** | **39.1** |
| Ours w/ composed noise | **22.1** | **21.7** | 32.5 | 33.1 |
| Ours *supervised* | 23.5 | 23.0 | 52.8 | 52.8 |

Table 7: F1 scores of facts extracted by our unsupervised semantic parsing (text→graph) systems and our model trained with supervision.

text generation models whereas $\mathcal{V}_{100}$ correlates better with performance on the larger test sets. We use $\mathcal{V}_{100}$ for model selection in the rest of this paper.

**Quantitative evaluation.** Table 3 shows BLEU, METEOR (Banerjee and Lavie, 2005) and CHRF++ (Popović, 2017a) for our unsupervised models and the rule baseline $R^{graph \to text}$, which is in many cases, i.e., if parallel graph-text data are scarce, the only alternative.

First, we observe that $R^{graph \to text}$ performs much better on WebNLG than VG, indicating that our new benchmark poses a tougher challenge. Second, our unsupervised models consistently outperform this baseline on all metrics and on both datasets, showing that our method produces textual descriptions much closer to human-generated ones. Third, noise composition, the general default in unsupervised machine translation, does not always perform better than noise sampling. Thus, it is worthwhile to try different noise settings for new tasks or datasets.

Surprisingly, supervised and unsupervised models perform nearly on par. Real supervision does not seem to give much better guidance in training than our unsupervised regime, as measured by our three metrics on two different datasets. Some metric-dataset combinations even favor one of the unsupervised models. Our qualitative observations provide a possible explanation for that.

**Qualitative observations.** Taking a look at example generations (Table 5), we also see qualitatively how much easier it is to grasp the content of our natural language summarization than reading through a simple enumeration of KG facts. We find that the unsupervised model (c) seems to output the KG information in a more complete manner than its supervised counterpart (d). The supervision probably introduces a bias present in the training data that image captions focus on salient image parts and therefore the supervised model is encouraged to omit information. As it never sees a corresponding text-graph pair together, the unsupervised model cannot draw such a conclusion.

### 6.2 Graph extraction from texts

We evaluate semantic parsing (text→graph) performance by computing the micro-averaged F1 score of extracted facts. If there are multiple reference graphs (cf. Section 5.1), an extracted fact is considered correct if it occurs in at least one reference graph. For the ground truth number of facts to be extracted from a given text, we take the maximum number of facts of all its reference graphs.

**Model selection.** Table 6 shows that (compared to text generation quality) $\mathcal{U}$ is more reliable for text→graph performance. For sampled noise, it correctly identifies the best iteration, whereas for composed noise it chooses second best. In both noise settings, $\mathcal{V}_{100}$ perfectly chooses the best model.

**Quantitative observations.** Table 7 shows a comparison of our unsupervised models with two rule-based systems, our $R^{text \to graph}$ and the highly domain-specific Stanford Scene Graph Parser (SSGP) by Schuster et al. (2015).

We choose these two baselines to adequately represent the state of the art in the unsupervised setting. Recall from Section 2 that the only previous unsupervised works either cannot adapt to a target graph schema (open information extraction), which means their precision and recall of retrieved facts is always 0, or have been created for SQL query

| Input sentence | Man wearing a colorful shirt and white pants playing tennis |
|---|---|
| Reference (RG) | (shirt, attr, colorful)<br>(pants, attr, white)<br>(man, wearing, shirt)<br>(man, wearing, pants) |
| $R^{text \to graph}$ | (Man, wearing, colorful)<br>(shirt, attr, colorful)<br>(pants, attr, white)<br>(pants, playing, tennis) |
| Stanford Scene Graph Parser | (shirt, play, tennis) ,<br>(pants, play, tennis) ,<br>(shirt, attr, colorful) ,<br>(pants, attr, white) |
| Unsuperv. model w/ composed noise | (pants, attr, colorful)<br>(pants, attr, white)<br>(man, wearing, shirt)<br>(man, playing, tennis) |
| Supervised model | (shirt, attr, colorful)<br>(pants, attr, white)<br>(Man, wearing, shirt)<br>(Man, wearing, pants) |

Table 8: Example fact extractions and evaluation wrt reference graph (RG). Green: correct ($\in$ RG). Yellow: acceptable fact, but $\notin$ RG. Red: incorrect ($\notin$ RG).

generation from natural language questions (Poon, 2013), a related task that is yet so different that an adaptation to triple set generation from natural language statements is nontrivial. While rule-based systems do not automatically adapt to new graph schemas either, $R^{text \to graph}$ and SSGP were at least designed with the scene graph domain in mind.

Although SSGP was not optimized to match the scene graphs from VG, its rules were still engineered to cover typical idiosyncrasies of textual image descriptions and corresponding scene graphs. Besides, we evaluate it with lemmatized reference graphs because it only predicts lemmata as predicates. All this gives it a major advantage over the other presented systems but it is nonetheless outperformed by our best unsupervised model – even on VG. This shows that our automatic method can beat even hand-crafted domain-specific rules.

Both $R^{text \to graph}$ and SSGP fail to predict any fact from WebNLG. The DBpedia facts from WebNLG often contain multi-token entities while $R^{text \to graph}$ only picks single tokens from the text. Likewise, SSGP models multi-token entities as two nodes

| | $VG_{ball}$ | | WebNLG | |
|---|---|---|---|---|
| | g→t BLEU | t→g F1 | g→t BLEU | t→g F1 |
| No noise | 0.9 | 0.0 | 14.8 | 0.0 |
| sample all noise funs | **19.9** | 17.3 | **39.1** | **38.5** |
| compose all noise funs | 19.6 | **19.0** | 30.8 | 32.5 |
| use only `rule` | **19.5** | **18.5** | 37.4 | **31.0** |
| use only `swap` | 0.9 | 0.0 | 13.1 | 0.0 |
| use only `drop` | 0.9 | 0.0 | **39.9** | 30.1 |
| use only `blank` | 0.9 | 0.0 | 14.9 | 0.0 |
| use only `repeat` | 1.1 | 0.0 | 15.7 | 0.0 |
| sample all but `rule` | 0.9 | 0.0 | 14.9 | 0.0 |
| sample all but `swap` | 19.2 | 17.0 | 39.6 | **37.3** |
| sample all but `drop` | 19.5 | 16.0 | 39.2 | 35.3 |
| sample all but `blank` | 19.9 | **17.5** | **41.0** | 37.0 |
| sample all but `repeat` | **20.4** | 16.6 | 36.7 | 37.1 |
| comp. all but `rule` | 0.9 | 0.0 | 13.5 | 0.0 |
| comp. all but `swap` | 20.2 | 16.3 | 35.9 | 40.8 |
| comp. all but `drop` | 21.5 | 18.6 | 36.4 | 41.1 |
| comp. all but `blank` | 20.2 | 16.3 | 34.8 | 40.4 |
| comp. all but `repeat` | **21.1** | **20.1** | **38.5** | **42.3** |

Table 9: Ablation study of our models on val of $VG_{ball}$ and WebNLG v2.1. Models selected based on $\mathcal{V}_{100}$. Bold: best performance per column and block. Underlined: worse than corresponding rule-based system.

with an `attr` relation. This illustrates the importance of automatic adaptation to the target KG. Although our system uses $R^{text \to graph}$ during unsupervised training and is similarly not adapted to the WebNLG dataset, it performs significantly better.

Supervision helps more on WebNLG than on VG. The poor performance of $R^{text \to graph}$ on WebNLG is probably a handicap for unsupervised learning.

**Qualitative observations.** Table 8 shows example facts extracted by different systems. $R^{text \to graph}$ and SSGP are both fooled by the proximity of the noun *pants* and the verb *play* whereas our model correctly identifies *man* as the subject. It, however, fails to identify *shirt* as an entity and associates the two attributes *colorful* and *white* to *pants*. Only the supervised model produces perfect output.

### 6.3 Noise and translation completeness

Sampled noise only creates training pairs that either are complete rule-based translations or reconstruction pairs from a noisy graph to a complete graph or a noisy text to a complete text. In contrast, composed noise can introduce translations from a noisy text to a complete graph or vice versa and thus encourage a system to omit input information (cf. Fig. 3). This difference is mirrored nicely in the results of our unsupervised systems for both tasks: composed noise performs better on VG where omit-

ted information in an image caption is common and sampled noise works better on WebNLG where the texts describe their graphs completely.

## 7 Noise Ablation Study

Our unsupervised objectives are defined by different types of noise models. Hence, we examine their impact in a noise ablation study. Table 9 shows results for text→graph and graph→text on the validation splits of $VG_{ball}$ and WebNLG.

For both datasets and tasks, introducing variation via noise functions is crucial for the success of unsupervised learning. The model without noise (i.e., $C(x) = x$) fails completely as do all models lacking `rule` as type of noise, the only exception being the only-`drop` system on WebNLG. Even though `drop` seems to work equally well in this one case, the simple translations delivered by our rule-based systems clearly provide the most useful information for the unsupervised models – notably in combination with the other noise functions: removing `rule` and keeping all other types of noise (cf. "sample all but `rule`" and "comp. all but `rule`") performs much worse than leaving out `drop`.

We hypothesize that our two rule systems provide two important pieces of information: (1) $R^{graph \to text}$ helps distinguish data format tokens from text tokens and (2) $R^{text \to graph}$ helps find probable candidate words in a text that form facts for the data output. As opposed to machine translation, where usually every word in a sentence is translated into a fluent sentence in the target language, identifying words that probably form a fact is more important in data-to/from-text generation.

We moreover observe that our unsupervised models always improve on the rule-based systems even when `rule` is the only type of noise: graph→text BLEU increases from 6.2/18.3 to 19.5/37.4 on $VG_{ball}$/WebNLG and text→graph F1 from 14.4/0.0 to 18.5/31.0.

Finally, our ablation study makes clear that there is no best noise model for all datasets and tasks. We therefore recommend experimenting with both different sets of noise functions and noise injection regimes (sampled vs. composed) for new data.

## 8 Conclusion

We presented the first fully unsupervised approach to text generation from KGs and a novel approach to unsupervised semantic parsing that automatically adapts to a target KG. We showed the effectiveness of our approach on two datasets, WebNLG v2.1 and a new text↔graph benchmark in the visual domain, derived from Visual Genome. We quantitatively and qualitatively analyzed our method on text↔graph conversion. We explored the impact of different unsupervised objectives in an ablation study and found that our newly introduced unsupervised objective using rule-based translations is essential for the success of unsupervised learning.

## Acknowledgments

We thank the anonymous reviewers for their helpful comments and gratefully acknowledge a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes). This work was supported by by the BMBF as part of the project MLWin (01IS18050).

## References

Mark van Assem, Aldo Gangemi, and Guus Schreiber. 2006. Conversion of wordnet to a standard rdf/owl representation. In *Proceedings of the Fifth Edition of the International Conference on Language Resources and Evaluation (LREC 2006)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computing Research Repository*, arXiv:1409.0473.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Rajarshi Bhowmik and Gerard de Melo. 2018. Generating fine-grained open vocabulary entity type descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 877–888, Melbourne, Australia. Association for Computational Linguistics.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale Seq2Seq models to multi-document inputs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4184–4194, Hong Kong, China. Association for Computational Linguistics.

Christiane Fellbaum. 2005. Wordnet and wordnets. In Keith Brown et al., editor, *Encyclopedia of Language and Linguistics*, second edition, pages 665–670. Elsevier, Oxford.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform. *Computing Research Repository*, arXiv:1803.07640.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56, Tilburg University, The Netherlands. Association for Computational Linguistics.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1486–1495, Portland, Oregon, USA. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *Computing Research Repository*, arXiv:1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing. In *Automated Knowledge Base Construction (AKBC)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

*Language Technologies*, Minneapolis, Minnesota. Association for Computational Linguistics.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Computing Research Repository*, arXiv:1602.07332.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

Tsung-Yi Lin, Xinlei Chen, Hao Fang, and Ramakrishna Vedantam. 2018. GitHub repository: tylin/coco-caption (Microsoft COCO caption evaluation). https://github.com/tylin/coco-caption.

Edward Loper and Steven Bird. 2004. Nltk: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. 2016. Visual relationship detection with language priors. In *European Conference on Computer Vision*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *Transactions of the Association for Computational Linguistics*, 4:231–244.

Susan W. McRoy, Songsak Channarukul, and Syed S. Ali. 2000. YAG: A template-based generator for real-time systems. In *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*, pages 264–267, Mitzpe Ramon, Israel. Association for Computational Linguistics.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 756–766, Denver, Colorado. Association for Computational Linguistics.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 933–943, Sofia, Bulgaria. Association for Computational Linguistics.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore. Association for Computational Linguistics.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 141–147, Geneva, Switzerland. COLING.

Maja Popović. 2017a. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Maja Popović. 2017b. GitHub repository: m-popovic/chrf (chrF). https://github.com/m-popovic/chrF.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-Text Generation with Content Selection and Planning. In *Proceedings of the 33rd Conference on Artificial Intelligence*.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3181–3192, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Computing Research Repository*, arXiv:2001.11003.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jacques Pierre Robin. 1995. *Revision-based Generation of Natural Language Summaries Providing Historical Background: Corpus-based Analysis, Design, Implementation and Evaluation*. Ph.D. thesis, Columbia University, New York, NY, USA. UMI Order No. GAX95-33653.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3027–3035.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the Fourth Workshop on Vision and Language*, pages 70–80, Lisbon, Portugal. Association for Computational Linguistics.

Anastasia Shimorina and Claire Gardent. 2018. Handling rare items in data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Tilburg University, The Netherlands. Association for Computational Linguistics.

Étienne Simon, Vincent Guigue, and Benjamin Piwowarski. 2019. Unsupervised information extraction: Regularizing discriminative approaches with relation distribution losses. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1378–1387, Florence, Italy. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 4444–4451. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Kumiko Tanaka-Ishii, Koiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1282–1288, Montreal, Quebec, Canada. Association for Computational Linguistics.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637, Melbourne, Australia. Association for Computational Linguistics.

Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. 2008. *Handbook of knowledge representation*, volume 1. Elsevier.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Hai Wan, Yonghao Luo, Bo Peng, and Wei-Shi Zheng. 2018. Representation learning for scene graph completion via jointly structural and visual embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 949–956. International Joint Conferences on Artificial Intelligence Organization.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. 2018. DrugBank 5.0: a major update to the DrugBank database

for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466, Edinburgh, Scotland, UK. Association for Computational Linguistics.

## A Hyperparameters

We use the following settings for all our experiments: learning rate of $10^{-4}$, word embeddings of size 300, an LSTM hidden size of 250, a dropout rate of 0.2 and a batch size of 10. Following Lample et al. (2018b), we set $p_{\text{blank}} = p_{\text{repeat}} = 0.2$, $p_{\text{drop}} = 0.1$. For inference, we decode greedily with a maximum number of 40 decoding steps. To speed up unsupervised learning, we increase the batch size to 64 when creating backtranslations.

## B Model details

We train with homogeneous batches of one target output type (text or graph) at a time. We use a single GeForce GTX 1080 GPU for training and inference. In this environment, pure training takes approximately 9 ms per instance and inference, which also means backtranslation, takes approximately 21 ms per instance. This means that unsupervised learning approximately needs 30 ms per instance. WebNLG models use 10.6 million parameters, VG models have 60.7 million parameters. The difference is due to a larger vocabulary size of 70,800 for VG compared to 8,171 for WebNLG.

## C Results of all iterations on WebNLG

See Table 10 for all intermediate graph→text results of unsupervised training on WebNLG and Table 11 for text→graph. We find similar trends as for VG (Tables 4 and 6) except for $\mathcal{U}$ being a less reliable performance indicator for text→graph in the sampled noise setting.

| | sampled noise | | | composed noise | | |
|---|---|---|---|---|---|---|
| # | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val |
| 1 | 91.7 | 12.8 | 13.0 | 23.0 | 15.9 | 15.5 |
| 2 | **94.0** | 14.7 | 15.8 | 53.2 | 22.2 | 20.7 |
| 3 | 85.2 | 25.5 | 26.0 | 71.0 | 23.2 | 22.8 |
| 4 | 65.9 | 27.7 | 28.8 | 75.2 | 25.3 | 26.2 |
| 5 | 65.5 | 31.4 | 30.7 | 69.2 | 25.9 | 27.2 |
| 6 | 58.1 | 31.5 | 31.0 | 71.5 | 27.6 | 27.7 |
| 7 | 48.0 | 31.3 | 32.3 | **79.2** | 29.0 | 27.7 |
| 8 | 48.3 | 32.8 | 33.4 | 52.5 | 28.1 | 27.5 |
| 9 | 37.5 | 33.2 | 34.0 | 57.1 | 30.5 | 30.0 |
| 10 | 42.1 | 32.8 | 33.4 | 52.4 | 30.6 | 29.9 |
| 11 | 38.7 | 34.7 | 34.8 | 59.9 | 32.0 | **31.6** |
| 12 | 38.7 | 36.4 | 36.2 | 42.1 | 30.4 | 30.8 |
| 13 | 39.3 | 33.5 | 35.1 | 50.0 | 30.7 | 30.7 |
| 14 | 40.5 | 36.9 | 36.6 | 46.7 | 30.9 | 30.7 |
| 15 | 41.8 | 36.5 | 37.5 | 48.2 | 31.1 | 30.3 |
| 16 | 43.2 | 36.9 | 38.0 | 43.7 | 30.3 | 29.6 |
| 17 | 39.1 | 35.6 | 36.6 | 43.1 | 29.0 | 29.7 |
| 18 | 38.5 | 37.5 | 38.3 | 31.1 | 29.7 | 29.8 |
| 19 | 38.8 | 37.8 | 38.4 | 39.5 | 29.0 | 29.8 |
| 20 | 37.5 | 37.2 | 38.6 | 36.2 | 31.3 | 29.8 |
| 21 | 36.4 | 36.8 | 38.4 | 35.2 | 30.0 | 30.8 |
| 22 | 44.8 | 36.3 | 39.7 | 37.6 | 32.4 | 30.7 |
| 23 | 40.8 | 35.8 | 38.2 | 39.6 | 31.4 | 30.3 |
| 24 | 35.8 | 39.2 | 39.6 | 39.6 | 32.4 | 30.3 |
| 25 | 40.6 | 38.5 | 39.5 | 37.0 | 33.2 | 30.9 |
| 26 | 36.8 | 38.9 | 40.3 | 41.3 | 32.3 | 30.2 |
| 27 | 44.1 | 39.7 | **40.6** | 37.3 | 33.0 | 30.4 |
| 28 | 39.3 | 36.9 | 38.9 | 39.0 | **34.7** | 30.8 |
| 29 | 36.1 | 37.6 | 38.6 | 41.5 | 31.0 | 30.6 |
| 30 | 38.7 | **40.7** | 39.1 | 42.9 | 30.6 | 30.0 |

Table 10: BLEU scores on WebNLG for our unsupervised models evaluated for graph→text at different iterations. $\mathcal{U}$ is calculated on all unlabeled data used for training. $\mathcal{V}_{100}$ is a 100-size random sample from val. All results are computed with scripts from (Lin et al., 2018).

| | sampled noise | | | composed noise | | |
|---|---|---|---|---|---|---|
| # | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val | $\mathcal{U}$ | $\mathcal{V}_{100}$ | val |
| 1 | **69.4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 64.0 | 0.0 | 0.1 | 16.2 | 1.2 | 1.6 |
| 3 | 35.6 | 0.9 | 0.3 | 7.5 | 3.3 | 3.0 |
| 4 | 47.8 | 2.6 | 2.3 | 37.5 | 5.5 | 5.5 |
| 5 | 39.2 | 5.7 | 3.4 | 35.3 | 7.0 | 6.6 |
| 6 | 39.2 | 6.2 | 5.6 | 44.9 | 9.7 | 8.0 |
| 7 | 45.8 | 9.8 | 7.9 | 58.3 | 8.0 | 10.3 |
| 8 | 50.0 | 12.6 | 10.0 | 51.1 | 14.0 | 12.8 |
| 9 | 54.9 | 13.6 | 12.9 | 53.1 | 12.5 | 14.0 |
| 10 | 58.3 | 14.9 | 14.3 | 51.1 | 15.9 | 16.8 |
| 11 | 62.5 | 19.3 | 17.8 | 53.8 | 15.6 | 17.3 |
| 12 | 54.2 | 20.3 | 18.2 | 58.3 | 16.7 | 18.0 |
| 13 | 57.1 | 23.1 | 20.2 | 47.8 | 19.8 | 20.6 |
| 14 | 37.5 | 25.5 | 21.4 | 49.0 | 20.6 | 22.1 |
| 15 | 48.0 | 25.7 | 22.4 | 54.2 | 23.0 | 22.8 |
| 16 | 52.0 | 27.9 | 24.3 | 46.2 | 22.5 | 25.4 |
| 17 | 50.0 | 26.7 | 25.1 | 35.6 | 26.8 | 26.8 |
| 18 | 48.0 | 32.1 | 27.7 | 52.2 | 27.8 | 27.7 |
| 19 | 56.0 | 32.3 | 28.9 | 58.3 | 26.4 | 28.1 |
| 20 | 60.0 | 31.0 | 30.1 | 55.3 | 26.4 | 29.2 |
| 21 | 51.0 | 32.3 | 30.4 | 59.3 | 27.6 | 30.7 |
| 22 | 55.3 | 34.9 | 32.0 | **62.5** | 31.7 | 32.0 |
| 23 | 44.9 | 34.3 | 32.7 | 54.9 | 34.0 | 32.6 |
| 24 | 58.8 | 38.4 | 33.7 | 61.2 | 31.5 | 32.4 |
| 25 | 46.8 | 39.6 | 34.1 | 58.3 | 33.3 | 33.1 |
| 26 | 53.8 | 40.6 | 36.3 | 54.2 | **34.4** | 32.5 |
| 27 | 62.5 | 41.8 | 36.4 | 50.0 | 33.9 | 33.3 |
| 28 | 55.3 | 41.0 | 37.4 | 40.8 | 32.6 | **33.7** |
| 29 | 56.0 | 40.7 | 37.0 | 58.8 | 29.5 | **33.7** |
| 30 | 59.6 | **41.9** | **38.5** | 53.8 | 31.6 | 33.4 |

Table 11: F1 scores on WebNLG for our unsupervised models evaluated for text→graph at different iterations. $\mathcal{U}$ is calculated on all unlabeled data used for training. $\mathcal{V}_{100}$ is a 100-size random sample from val.

# Chapter 6

# Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs

# Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs

**Martin Schmitt**[1] **Leonardo F. R. Ribeiro**[2] **Philipp Dufter**[1] **Iryna Gurevych**[2] **Hinrich Schütze**[1]

[1]Center for Information and Language Processing (CIS), LMU Munich
[2]Research Training Group AIPHES and UKP Lab, Technische Universität Darmstadt
martin@cis.lmu.de

## Abstract

We present *Graformer*, a novel Transformer-based encoder-decoder architecture for graph-to-text generation. With our novel graph self-attention, the encoding of a node relies on all nodes in the input graph – not only direct neighbors – facilitating the detection of global patterns. We represent the relation between two nodes as the length of the shortest path between them. Graformer learns to weight these node-node relations differently for different attention heads, thus virtually learning differently connected views of the input graph. We evaluate Graformer on two popular graph-to-text generation benchmarks, AGENDA and WebNLG, where it achieves strong performance while using many fewer parameters than other approaches.[1]

## 1 Introduction

A knowledge graph (KG) is a flexible data structure commonly used to store both general world knowledge (Auer et al., 2008) and specialized information, e.g., in biomedicine (Wishart et al., 2018) and computer vision (Krishna et al., 2017). Generating a natural language description of such a graph (KG→text) makes the stored information accessible to a broader audience of end users. It is therefore important for KG-based question answering (Bhowmik and de Melo, 2018), data-to-document generation (Moryossef et al., 2019; Koncel-Kedziorski et al., 2019) and interpretability of KGs in general (Schmitt et al., 2020).

Recent approaches to KG→text employ encoder-decoder architectures: the encoder first computes vector representations of the graph's nodes, the decoder then uses them to predict the text sequence. Typical encoder choices are graph neural networks based on message passing between direct neighbors in the graph (Kipf and Welling, 2017; Veličković et al., 2018) or variants of Transformer (Vaswani et al., 2017) that apply self-attention on all nodes together, including those that are not directly connected. To avoid losing information, the latter approaches use edge or node *labels* from the shortest path when computing the attention between two nodes (Zhu et al., 2019; Cai and Lam, 2020). Assuming the existence of a path between any two nodes is particularly problematic for KGs: a set of KG facts often does not form a connected graph.

We propose a flexible alternative that neither needs such an assumption nor uses label information to model graph structure: a Transformer-based encoder that interprets the *lengths* of shortest paths in a graph as relative position information and thus, by means of multi-head attention, dynamically learns different structural views of the input graph with differently weighted connection patterns. We call this new architecture *Graformer*.

Following previous work, we evaluate Graformer on two benchmarks: (i) the AGENDA dataset (Koncel-Kedziorski et al., 2019), i.e., the generation of scientific abstracts from automatically extracted entities and relations specific to scientific text, and (ii) the WebNLG challenge dataset (Gardent et al., 2017), i.e., the task of generating text from DBPedia subgraphs. On both datasets, Graformer achieves more than 96% of the state-of-the-art performance while using only about half as many parameters.

In summary, our contributions are as follows: (1) We develop *Graformer*, a novel graph-to-text architecture that interprets shortest path lengths as relative position information in a graph self-attention network. (2) Graformer achieves competitive performance on two popular KG-to-text generation benchmarks, showing that our architecture can learn about graph structure without any guidance other than its text generation objective. (3) To further investigate what Graformer learns about graph structure, we visualize the differently connected

[1]Our code is publicly available: https://github.com/mnschmit/graformer



*Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 10–21
June 11, 2021. ©2021 Association for Computational Linguistics

graph views it has learned and indeed find different attention heads for more local and more global graph information. Interestingly, direct neighbors are considered particularly important even without any structural bias, such as introduced by a graph neural network. (4) Analyzing the performance w.r.t. different input graph properties, we find evidence that Graformer's more elaborate global view on the graph is an advantage when it is important to distinguish between distant but connected nodes and truly unreachable ones.

## 2 Related Work

Most recent approaches to graph-to-text generation employ a graph neural network (GNN) based on message passing through the input graph's topology as the encoder in their encoder-decoder architectures (Marcheggiani and Perez-Beltrachini, 2018; Koncel-Kedziorski et al., 2019; Ribeiro et al., 2019; Guo et al., 2019). As one layer of these encoders only considers immediate neighbors, a large number of stacked layers can be necessary to learn about distant nodes, which in turn also increases the risk of propagating noise (Li et al., 2018).

Other approaches (Zhu et al., 2019; Cai and Lam, 2020) base their encoder on the Transformer architecture (Vaswani et al., 2017) and thus, in each layer, compute self-attention on all nodes, not only direct neighbors, facilitating the information flow between distant nodes. Like Graformer, these approaches incorporate information about the graph topology with some variant of relative position embeddings (Shaw et al., 2018). They, however, assume that there is always a path between any pair of nodes, i.e., there are no unreachable nodes or disconnected subgraphs. Thus they use an LSTM (Hochreiter and Schmidhuber, 1997) to compute a relation embedding from the labels along this path. However, in contrast to the AMR[2] graphs used for their evaluation, KGs are frequently disconnected. Graformer is more flexible and makes no assumption about connectivity. Furthermore, its relative position embeddings only depend on the *lengths* of shortest paths i.e., purely structural information, not labels. It thus effectively learns *differently connected views* of its input graph.

Deficiencies in modeling long-range dependencies in GNNs have been considered a serious limitation before. Various solutions orthogonal to our approach have been proposed in recent work: By incorporating a connectivity score into their graph attention network, Zhang et al. (2020) manage to increase the attention span to k-hop neighborhoods but, finally, only experiment with $k = 2$. Our graph encoder efficiently handles dependencies between much more distant nodes. Pei et al. (2020) define an additional neighborhood based on Euclidean distance in a continuous node embedding space. Similar to our work, a node can thus receive information from distant nodes, given their embeddings are close enough. However, Pei et al. (2020) compute these embeddings only once before training whereas in our approach node similarity is based on the learned representation in each encoder layer. This allows Graformer to dynamically change node interaction patterns during training.

Recently, Ribeiro et al. (2020) use two GNN encoders – one using the original topology and one with a fully connected version of the graph – and combine their output in various ways for graph-to-text generation. This approach can only see two extreme versions of the graph: direct neighbors and full connection. Our approach is more flexible and dynamically learns a different structural view per attention head. It is also more parameter-efficient as our multi-view encoder does not need a separate set of parameters for each view.

## 3 The Graformer Model

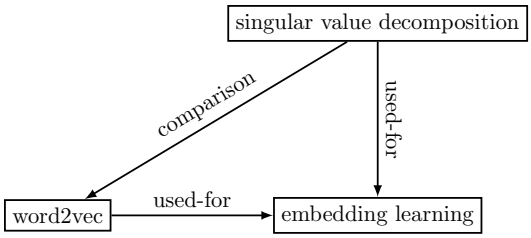
Graformer follows the general multi-layer encoder-decoder pattern known from the original Transformer (Vaswani et al., 2017). In the following, we first describe our formalization of the KG input and then how it is processed by Graformer.

### 3.1 Graph data structure

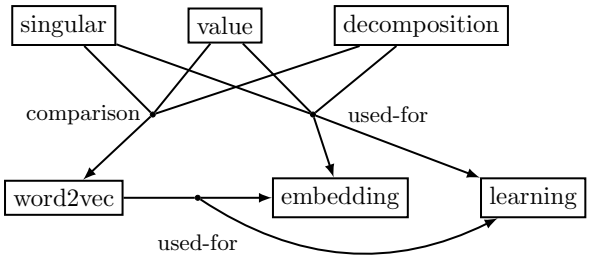
**Knowledge graph.** We formalize a knowledge graph (KG) as a directed, labeled multigraph $G_{KG} = (V, A, s, t, l_V, l_A, \mathcal{E}, \mathcal{R})$ with $V$ a set of vertices (the KG entities), $A$ a set of arcs (the KG facts), $s, t : A \to V$ functions assigning to each arc its source/target node (the subject/object of a KG fact), and $l_V : V \to \mathcal{E}, l_A : A \to \mathcal{R}$ providing labels for vertices and arcs, where $\mathcal{R}$ is a set of KG-specific relations and $\mathcal{E}$ a set of entity names.

**Token graph.** Entity names usually consist of more than one token or subword unit. Hence, a tokenizer $tok : \mathcal{E} \to \Sigma_T^*$ is needed that splits an entity's label into its components from the vocabulary $\Sigma_T$ of text tokens. Following recent work (Ribeiro et al., 2020), we mimic this composition-

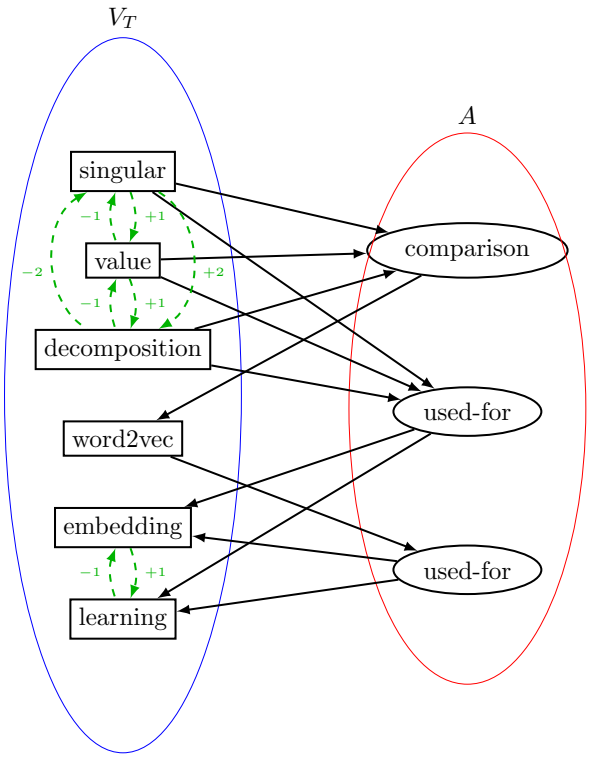
[2] abstract meaning representation




(a) Original knowledge graph




(b) Directed hypergraph (token graph)




(c) Incidence graph with $\text{SAME}_p$ edges (dashed green)

Figure 1: Different representations of the same KG (types are omitted for clarity).

ality of node labels in the graph structure by splitting each node into as many nodes as there are tokens in its label. We thus obtain a directed hypergraph $G_T = (V_T, A, s_T, t_T, l_T, l_A, \Sigma_T, \mathcal{R}, same)$, where $s_T, t_T : A \to \mathcal{P}(V_T)$ now assign a set of source (resp. target) nodes to each (hyper-) arc and all nodes are labeled with only one token, i.e., $l_T : V_T \to \Sigma_T$. Unlike Ribeiro et al. (2020), we additionally keep track of all token nodes' origins: $same : V_T \to \mathcal{P}(V_T \times \mathbb{Z})$ assigns to each node $n$ all other nodes $n'$ stemming from the same entity together with the relative position of $l_T(n)$ and $l_T(n')$ in the original tokenized entity name. Fig. 1b shows the token graph corresponding to the KG in Fig. 1a.

**Incidence graph.** For ease of implementation, our final data structure for the KG is the hypergraph's incidence graph, a bipartite graph where hyper-arcs are represented as nodes and edges are unlabeled: $G = (N, E, l, \Sigma, \{\, \text{SAME}_p \mid p \in \mathbb{Z} \,\})$ where $N = V_T \cup A$ is the set of nodes, $E = \{\, (n_1, n_2) \mid n_1 \in s_T(n_2) \vee n_2 \in t_T(n_1) \,\}$ the set of directed edges, $l : N \to \Sigma$ a label function, and $\Sigma = \Sigma_T \cup \mathcal{R}$ the vocabulary. We introduce $\text{SAME}_p$ edges to fully connect $same$ clusters: $\text{SAME}_p = \{\, (n_1, n_2) \mid (n_2, p) \in same(n_1) \,\}$ where $p$ differentiates between different relative positions in the original entity string, similar to (Shaw et al., 2018). See Fig. 1c for an example.

### 3.2 Graformer encoder

The initial graph representation $\boldsymbol{H}^{(0)} \in \mathbb{R}^{|N| \times d}$ is obtained by looking up embeddings for the node labels in the learned embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|\Sigma| \times d}$, i.e., $\boldsymbol{H}_i^{(0)} = \boldsymbol{e}^{l(n_i)} \boldsymbol{E}$ where $\boldsymbol{e}^{l(n_i)}$ is the one-hot-encoding of the $i$th node's label.

To compute the node representation $\boldsymbol{H}^{(L)}$ in the $L$th layer, we follow Vaswani et al. (2017), i.e., we first normalize the input from the previous layer $\boldsymbol{H}^{(L-1)}$ via layer normalization $LN$, followed by multi-head graph self-attention $SelfAtt_g$ (see § 3.3 for details), which – after dropout regularization $Dr$ and a residual connection – yields the intermediate representation $\boldsymbol{\mathcal{I}}$ (cf. Eq. (1)). A feedforward layer $FF$ with one hidden layer and GeLU (Hendrycks and Gimpel, 2016) activation computes the final layer output (cf. Eq. (2)). As recommended by Chen et al. (2018), we apply an additional layer normalization step to the output $\boldsymbol{H}^{(L_E)}$ of the last encoder layer $L_E$.

$$\boldsymbol{\mathcal{I}}^{(L)} = Dr(SelfAtt_g(LN(\boldsymbol{H}^{(L-1)}))) + \boldsymbol{H}^{(L-1)} \tag{1}$$

$$\boldsymbol{H}^{(L)} = Dr(FF(LN(\boldsymbol{\mathcal{I}}^{(L)}))) + \boldsymbol{\mathcal{I}}^{(L)} \tag{2}$$

$SelfAtt_g$ computes a weighted sum of $\boldsymbol{H}^{(L-1)}$:

$$SelfAtt_g(\boldsymbol{H})_i = \sum_{j=1}^{|N|} \alpha_{ij}^g (\boldsymbol{H}_j \boldsymbol{W}^{V_g}) \tag{3}$$

where $\boldsymbol{W}^{V_g} \in \mathbb{R}^{d \times d}$ is a learned parameter matrix.

In the next section, we derive the definition of the graph-structure-informed attention weights $\alpha_{ij}^g$.

### 3.3 Self-attention for text and graphs with relative position embeddings

In this section, we describe the computation of attention weights for multi-head self-attention. Note that the formulas describe the computations for one head. The output of multiple heads is combined as in the original Transformer (Vaswani et al., 2017).

**Text self-attention.** Shaw et al. (2018) introduced position-aware self-attention in the Transformer by (i) adding a relative position embedding $\mathbf{A}^K \in \mathbb{R}^{M\times M\times d}$ to $\boldsymbol{X}$'s key representation, when computing the softmax-normalized attention scores $\boldsymbol{\alpha}_i$ between $\boldsymbol{X}_i \in \mathbb{R}^d$ and the complete input embedding matrix $\boldsymbol{X} \in \mathbb{R}^{M\times d}$ (cf. Eq. (4)), and (ii) adding a second type of position embedding $\mathbf{A}^V \in \mathbb{R}^{M\times M\times d}$ to $\boldsymbol{X}$'s value representation when computing the weighted sum (cf. Eq. (5)):

$$\boldsymbol{\alpha}_i = \sigma\left(\frac{\boldsymbol{X}_i\boldsymbol{W}^Q(\boldsymbol{X}\boldsymbol{W}^K + \mathbf{A}_i^K)^\top}{\sqrt{d}}\right) \quad (4)$$

$$\boldsymbol{V}_i = \sum_{j=1}^{n}\alpha_{ij}(\boldsymbol{X}_j\boldsymbol{W}^V + \mathbf{A}_{ij}^V) \quad (5)$$

where $\sigma(\cdot)$ denotes the softmax function, i.e.,

$$\sigma(\boldsymbol{b})_i = \frac{\exp(b_i)}{\sum_{j=1}^J \exp(b_j)}, \quad \text{for } \boldsymbol{b} \in \mathbb{R}^J.$$

Recent work (Raffel et al., 2019) has adopted a simplified form where value-modifying embeddings $\mathbf{A}^V$ are omitted and key-modifying embeddings $\mathbf{A}^K$ are replaced with learned scalar embeddings $\boldsymbol{S} \in \mathbb{R}^{M\times M}$ that – based on relative position – directly in- or decrease attention scores before normalization, i.e., Eq. (4) becomes Eq. (6).

$$\boldsymbol{\alpha}_i = \sigma\left(\frac{\boldsymbol{X}_i\boldsymbol{W}^Q(\boldsymbol{X}\boldsymbol{W}^K)^\top}{\sqrt{d}} + \boldsymbol{S}_i\right) \quad (6)$$

Shaw et al. (2018) share their position embeddings across attention heads but learn separate embeddings for each layer as word representations from different layers can vary a lot. Raffel et al. (2019) learn separate $\boldsymbol{S}$ matrices for each attention head but share them across layers. We use Raffel et al. (2019)'s form of relative position encoding for text self-attention in our decoder (§ 3.4).

**Graph self-attention.** Analogously to self-attention on text, we define our structural graph self-attention as follows:

| | $V_T$ | | | | | | $A$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | s | v | d | w | e | l | c | u1 | u2 |
| s | 0 | 4 | 5 | 2 | 2 | 2 | 1 | 1 | 3 |
| v | -4 | 0 | 4 | 2 | 2 | 2 | 1 | 1 | 3 |
| d | -5 | -4 | 0 | 2 | 2 | 2 | 1 | 1 | 3 |
| w | -2 | -2 | -2 | 0 | 2 | 2 | -1 | ∞ | 1 |
| e | -2 | -2 | -2 | -2 | 0 | 4 | -3 | -1 | -1 |
| l | -2 | -2 | -2 | -2 | -4 | 0 | -3 | -1 | -1 |
| c | -1 | -1 | -1 | 1 | 3 | 3 | 0 | ∞ | 2 |
| u1 | -1 | -1 | -1 | ∞ | 1 | 1 | ∞ | 0 | ∞ |
| u2 | -3 | -3 | -3 | -1 | 1 | 1 | -2 | ∞ | 0 |

Figure 2: $\boldsymbol{R}$ matrix for the graph in Fig. 1c ($\delta_{max} = 3$).

$$\boldsymbol{\alpha}_i^g = \sigma\left(\frac{\boldsymbol{H}_i\boldsymbol{W}^{Q_g}(\boldsymbol{H}\boldsymbol{W}^{K_g})^\top}{\sqrt{d}} + \gamma(\boldsymbol{R})_i\right) \quad (7)$$

$\boldsymbol{W}^{K_g}, \boldsymbol{W}^{Q_g} \in \mathbb{R}^{d\times d}$ are learned matrices and $\gamma : \mathbb{Z}\cup\{\infty\} \rightarrow \mathbb{R}$ looks up *learned scalar embeddings* for the relative graph positions in $\boldsymbol{R} \in \mathbb{R}^{N\times N}$.

We define the relative graph position $R_{ij}$ between the nodes $n_i$ and $n_j$ with respect to two factors: (i) the text relative position $p$ in the original entity name if $n_i$ and $n_j$ stem from the same original entity, i.e., $(n_i, n_j) \in \text{SAME}_p$ for some $p$ and (ii) shortest path lengths otherwise:

$$R_{ij} = \begin{cases} \infty, & \text{if } \delta(n_i,n_j) = \infty \\ & \text{and } \delta(n_j,n_i) = \infty \\ encode(p), & \text{if } (n_i,n_j) \in \text{SAME}_p \\ \delta(n_i,n_j), & \text{if } \delta(n_i,n_j) \le \delta(n_j,n_i) \\ -\delta(n_j,n_i), & \text{if } \delta(n_i,n_j) > \delta(n_j,n_i) \end{cases} \quad (8)$$

where $\delta(n_i, n_j)$ is the length of the shortest path from $n_i$ to $n_j$, which we define to be $\infty$ if and only if there is no such path. *encode* maps a text relative position $p \in \mathbb{Z}\setminus\{0\}$ to an integer outside $\delta$'s range to avoid clashes. Concretely, we use $encode(p) := sgn(p)\cdot\delta_{max} + p$ where $\delta_{max}$ is the maximum graph diameter, i.e., the maximum value of $\delta$ over all graphs under consideration.

Thus, we model graph relative position as the length of the shortest path using either only forward edges ($R_{ij} > 0$) or only backward edges ($R_{ij} < 0$). Additionally, two special cases are considered: (i) Nodes without any purely forward or purely backward path between them ($R_{ij} = \infty$) and (ii) token nodes from the same entity. Here the relative position in the original entity string $p$ is encoded outside the range of path length encodings (which are always in the interval $[-\delta_{max}, \delta_{max}]$).

In practice, we use two thresholds, $n_\delta$ and $n_p$. All values of $\delta$ exceeding $n_\delta$ are set to $n_\delta$ and analogously for $p$. This limits the number of different positions a model can distinguish.

**Intuition.** Our definition of relative position in graphs combines several advantages: (i) Any node can attend to any other node – even unreachable ones – while learning a suitable attention bias for different distances. (ii) $\text{SAME}_p$ edges are treated differently in the attention mechanism. Thus, entity representations can be learned like in a regular transformer encoder, given that tokens from the same entity are fully connected with $\text{SAME}_p$ edges with $p$ providing relative position information. (iii) The lengths of shortest paths often have an intuitively useful interpretation in our incidence graphs and the sign of the entries in $\boldsymbol{R}$ also captures the important distinction between incoming and outgoing paths. In this way, Graformer can, e.g., capture the difference between the subject and object of a fact, which is expressed as a relative position of $-1$ vs. 1. The subject and object nodes, in turn, see each other as 2 and $-2$, respectively.

Fig. 2 shows the $\boldsymbol{R}$ matrix corresponding to the graph from Fig. 1c. Note how token nodes from the same entity, e.g., `s`, `v`, and `d`, form clusters as they have the same distances to other nodes, and how the relations inside such a cluster are encoded outside the interval $[-3, 3]$, i.e., the range of shortest path lengths. It is also insightful to compare node pairs with the same value in $\boldsymbol{R}$. E.g., both `s` and `w` see `e` at a distance of 2 because the entities *SVD* and *word2vec* are both the subject of a fact with *embedding learning* as the object. Likewise, `s` sees both `c` and `u1` at a distance of 1 because its entity *SVD* is subject to both corresponding facts.

### 3.4 Graformer decoder

Our decoder follows closely the standard Transformer decoder (Vaswani et al., 2017), except for the modifications suggested by Chen et al. (2018).

**Hidden decoder representation.** The initial decoder representation $\boldsymbol{Z}^{(0)} \in \mathbb{R}^{M \times d}$ embeds the (partially generated) target text $\boldsymbol{T} \in \mathbb{R}^{M \times |\Sigma|}$, i.e., $\boldsymbol{Z}^{(0)} = \boldsymbol{T}\boldsymbol{E}$. A decoder layer $L$ then obtains a contextualized representation via self-attention as in the encoder (§ 3.2):

$$\boldsymbol{C}^{(L)} = Dr(SelfAtt_t(LN(\boldsymbol{Z}^{(L-1)}))) + \boldsymbol{Z}^{(L-1)} \tag{9}$$

$SelfAtt_t$ differs from $SelfAtt_g$ by using different position embeddings in Eq. (7) and, obviously, $R_{ij}$ is defined in the usual way for text. $\boldsymbol{C}^{(L)}$ is then modified via multi-head attention $MHA$ on the output $\boldsymbol{H}^{(L_E)}$ of the last graph encoder layer $L_E$. As in § 3.2, we make use of residual connections, layer normalization $LN$, and dropout $Dr$:

$$\boldsymbol{U}^{(L)} = Dr(MHA(LN(\boldsymbol{C}^{(L)}), \boldsymbol{H}^{(L_E)})) + \boldsymbol{C}^{(L)} \tag{10}$$

$$\boldsymbol{Z}^{(L)} = Dr(FF(LN(\boldsymbol{U}^{(L)}))) + \boldsymbol{U}^{(L)} \tag{11}$$

where

$$MHA(\boldsymbol{C}, \boldsymbol{H})_i = \sum_{j=1}^{|N|} \alpha_{ij}(\boldsymbol{H}_j \boldsymbol{W}^{V_t}) \tag{12}$$

$$\boldsymbol{\alpha}_i = \sigma\left(\frac{\boldsymbol{C}_i \boldsymbol{W}^{Q_t} (\boldsymbol{H}\boldsymbol{W}^{K_t})^\top}{\sqrt{d}}\right) \tag{13}$$

**Generation probabilities.** The final representation $\boldsymbol{Z}^{(L_D)}$ of the last decoder layer $L_D$ is used to compute the probability distribution $\boldsymbol{P}_i \in [0, 1]^{|\Sigma|}$ over all words in the vocabulary $\Sigma$ at time step $i$:

$$\boldsymbol{P}_i = \sigma\left(\boldsymbol{Z}_i^{(L_D)} \boldsymbol{E}^\top\right) \tag{14}$$

Note that $\boldsymbol{E} \in \mathbb{R}^{|\Sigma| \times d}$ is the same matrix that is also used to embed node labels and text tokens.

### 3.5 Training

We train Graformer by minimizing the standard negative log-likelihood loss based on the likelihood estimations described in the previous section.

## 4 Experiments

### 4.1 Datasets

We evaluate our new architecture on two popular benchmarks for KG-to-text generation, AGENDA (Koncel-Kedziorski et al., 2019) and WebNLG (Gardent et al., 2017). While the latter contains crowd-sourced texts corresponding to subgraphs from various DBPedia categories, the former was automatically created by applying an information extraction tool (Luan et al., 2018) on a corpus of scientific abstracts (Ammar et al., 2018). As this process is noisy, we corrected 7 train instances where an entity name was erroneously split on a special character and, for the same reason, deleted 1 train instance entirely. Otherwise, we use the data as is, including the train/dev/test split.

We list the number of instances per data split, as well as general statistics about the graphs in Table 1. Note that the graphs in WebNLG are human-authored subgraphs of DBpedia while the graphs

| | AGENDA | WebNLG |
|---|---|---|
| #instances in train | 38,719 | 18,102 |
| #instances in val | 1,000 | 872 |
| #instances in test | 1,000 | 971 |
| #relation types | 7 | 373 |
| avg #entities in KG | 13.4 | 4.0 |
| % connected graphs | 0.3 | 99.9 |
| avg #graph components | 8.4 | 1.0 |
| avg component size | 1.6 | 3.9 |
| avg #token nodes in graph | 98.0 | 36.0 |
| avg #tokens in text | 157.9 | 31.5 |
| avg % text tokens in graph | 42.7 | 56.1 |
| avg % graph tokens in text | 48.6 | 49.0 |
| Vocabulary size $\|\Sigma\|$ | 24,100 | 2,100 |
| Character coverage in % | 99.99 | 100.0 |

Table 1: Statistics of AGENDA and the dataset from the WebNLG challenge as used in our experiments. Upper part: data splits and original KGs. Lower part: token graphs and BPE settings.

in AGENDA were automatically extracted. This leads to a higher number of disconnected graph components. Nearly all WebNLG graphs consist of a single component, i.e., are connected graphs, whereas for AGENDA this is practically never the case. We also report statistics that depend on the tokenization (cf. § 4.2) as factors like the length of target texts and the percentage of tokens shared verbatim between input graph and target text largely impact the task difficulty.

### 4.2 Data preprocessing

Following previous work on AGENDA (Ribeiro et al., 2020), we put the paper title into the graph as another entity. In contrast to Ribeiro et al. (2020), we also link every node from a real entity to every node from the title by TITLE2TXT and TXT2TITLE edges. The type information provided by AGENDA is, as usual for KGs, expressed with one dedicated node per type and HAS-TYPE arcs that link entities to their types. We keep the original pretokenized texts but lowercase the title as both node labels and target texts are also lowercased.

For WebNLG, we follow previous work (Gardent et al., 2017) by replacing underscores in entity names with whitespace and breaking apart camelcased relations. We furthermore follow the evaluation protocol of the original challenge by converting all characters to lowercased ASCII and separating all punctuation from alphanumeric characters during tokenization.

For both datasets, we train a BPE vocabulary using sentencepiece (Kudo and Richardson, 2018) on the train set, i.e., a concatenation of node labels and target texts. See Table 1 for vocabulary sizes. Note that for AGENDA, only 99.99% of the characters found in the train set are added to the vocabulary. This excludes exotic Unicode characters that occur in certain abstracts.

We prepend entity and relation labels with dedicated $\langle E \rangle$ and $\langle R \rangle$ tags.

### 4.3 Hyperparameters and training details

We train Graformer with the Adafactor optimizer (Shazeer and Stern, 2018) for 40 epochs on AGENDA and 200 epochs on WebNLG. We report test results for the model yielding the best validation performance measured in corpus-level BLEU (Papineni et al., 2002). For model selection, we decode greedily. The final results are generated by beam search. Following Ribeiro et al. (2020), we couple beam search with a length penalty (Wu et al., 2016) of 5.0. See Appendix A for more details and a full list of hyperparameters.

### 4.4 Epoch curriculum

We apply a data loading scheme inspired by the bucketing approach of Koncel-Kedziorski et al. (2019) and length-based curriculum learning (Platanios et al., 2019): We sort the train set by target text length and split it into four buckets of two times 40% and two times 10% of the data. After each training epoch, the buckets are shuffled internally but their global order stays the same from shorter target texts to longer ones. This reduces padding during batching as texts of similar lengths stay together and introduces a mini-curriculum from presumably easier examples (i.e., shorter targets) to more difficult ones for each epoch. This enables us to successfully train Graformer *even without a learning rate schedule*.

## 5 Results and Discussion

### 5.1 Overall performance

Table 2 shows the results of our evaluation on AGENDA in terms of BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and CHRF++ (Popović, 2017). Like the models we compare with, we report the average and standard deviation of 4 runs with different random seeds.

Our model outperforms previous Transformer-based models that only consider first-order neighborhoods per encoder layer (Koncel-Kedziorski et al., 2019; An et al., 2019). Compared to the very

| | BLEU | METEOR | CHRF++ | #P |
|---|---|---|---|---|
| Ours | 17.80 ±0.31 | 22.07 ±0.23 | 45.43 ±0.39 | 36.3 |
| GT | 14.30 ±1.01 | 18.80 ±0.28 | – | – |
| GT+RBS | 15.1 ±0.97 | 19.5 ±0.29 | – | – |
| CGE-LW | 18.01 ±0.14 | 22.34 ±0.07 | 46.69 ±0.17 | 69.8 |

Table 2: Experimental results on AGENDA. GT (Graph Transformer) from (Koncel-Kedziorski et al., 2019); GT+RBS from (An et al., 2019); CGE-LW from (Ribeiro et al., 2020). Number of parameters in millions.

| | BLEU | METEOR | CHRF++ | #P |
|---|---|---|---|---|
| Ours | 61.15 ±0.22 | 43.38 ±0.17 | 75.43 ±0.19 | 5.3 |
| UPF-FORGe | 40.88 | 40.00 | – | – |
| Melbourne | 54.52 | 41.00 | 70.72 | – |
| Adapt | 60.59 | 44.00 | 76.01 | – |
| Graph Conv. | 55.90 | 39.00 | – | 4.9 |
| GTR-LSTM | 58.60 | 40.60 | – | – |
| E2E GRU | 57.20 | 41.00 | – | – |
| CGE-LW-LG | 63.69 ±0.10 | 44.47 ±0.12 | 76.66 ±0.10 | 10.4 |

Table 3: Experimental results on the WebNLG test set with seen categories. CGE-LW-LG from (Ribeiro et al., 2020); Adapt, Melbourne and UPF-FORGe from (Gardent et al., 2017); Graph Conv. from (Marcheggiani and Perez-Beltrachini, 2018); GTR-LSTM from (Trisedya et al., 2018); E2E GRU from (Castro Ferreira et al., 2019). Number of parameters in millions.

recent models by Ribeiro et al. (2020), Graformer performs very similarly. Using both a local and a global graph encoder, Ribeiro et al. (2020) combine information from very distant nodes but at the same time need extra parameters for the second encoder. Graformer is more efficient and still matches their best model's BLEU and METEOR scores within a standard deviation.

The results on the test set of seen categories of WebNLG (Table 3) look similar. Graformer outperforms most original challenge participants and more recent work. While not performing on par with CGE-LW on WebNLG, Graformer still achieves more than 96% of its performance while using only about half as many parameters.

### 5.2 Performance on different types of graphs

We investigate whether Graformer is more suitable for disconnected graphs by comparing its performance on different splits of the AGENDA test set according to two graph properties: (i) the average number of nodes per connected component ($\mu_c$) and (ii) the largest diameter across all of a graph's components (d).

| $\mu_c$ | | BLEU | METEOR | CHRF++ |
|---|---|---|---|---|
| $< 1.25$ | Ours | **15.44** | 20.59 | 43.23 |
| (213) | CGE-LW | 15.34 | **20.64** | **43.56** |
| $< 1.5$ | Ours | **17.45** | 22.03 | 45.67 |
| (338) | CGE-LW | 17.29 | **22.32** | **45.88** |
| $< 2.0$ | Ours | 18.94 | 22.86 | 46.49 |
| (294) | CGE-LW | **19.46** | **23.76** | **47.78** |
| $\geq 2.0$ | Ours | **21.72** | 24.22 | 48.79 |
| (155) | CGE-LW | 20.97 | **24.98** | **49.83** |

(a) Average size $\mu_c$ of graph components.

| d | | BLEU | METEOR | CHRF++ |
|---|---|---|---|---|
| 1 | Ours | **16.48** | **21.16** | 43.94 |
| (368) | CGE-LW | 16.33 | **21.16** | **44.16** |
| 2 | Ours | **18.46** | 22.70 | 46.85 |
| (414) | CGE-LW | 18.20 | **23.14** | **47.28** |
| $\geq 3$ | Ours | 19.44 | 23.17 | 47.29 |
| (218) | CGE-LW | **20.32** | **24.42** | **49.25** |

(b) Largest diameter d across all of a graph's components.

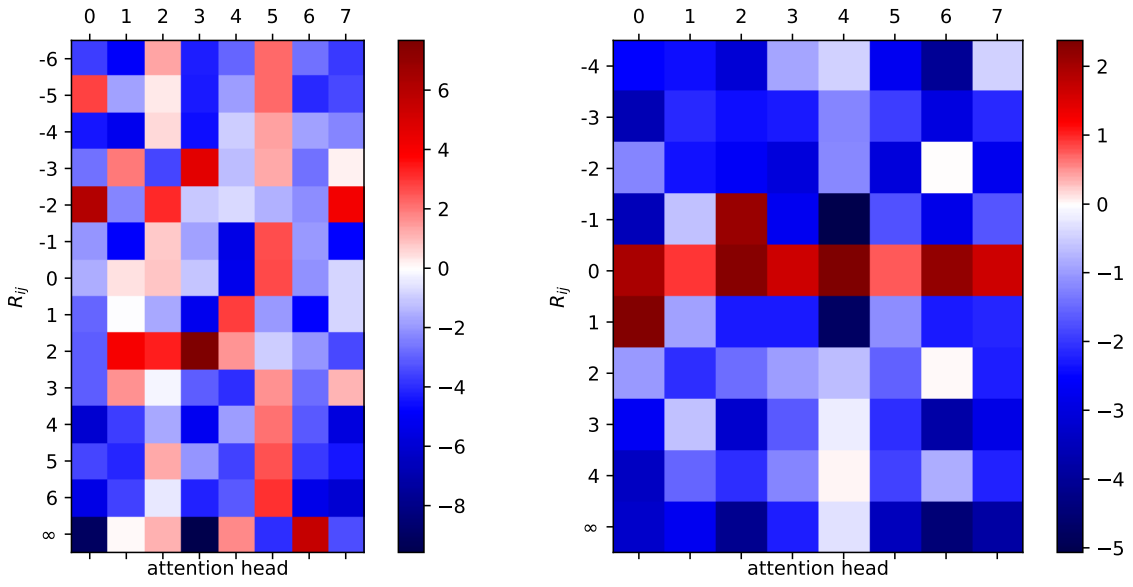
Table 4: Performance of a single run on the test split of AGENDA w.r.t. different input graph properties. The number of data points in each split is indicated in parentheses.

We can see in Table 4 that the performance of both Graformer and CGE-LW (Ribeiro et al., 2020) increases with more graph structure (larger $\mu_c$ and d), i.e., more information leads to more accurate texts. Besides, Graformer outperforms CGE-LW on BLEU for graphs with smaller components ($0 < \mu_c < 1.5$) and smaller diameters ($d < 3$). Although METEOR and CHRF++ scores always favor CGE-LW, the performance difference is also smaller for cases where BLEU favors Graformer.

We conjecture that Graformer benefits from its more elaborate global view, i.e., its ability to distinguish between distant but connected nodes and unreachable ones. CGE-LW's global encoder cannot make this distinction because it only sees a fully connected version of the graph.

Curiously, Graformer's BLEU is also better for larger components ($\mu_c \geq 2.0$). With multiple larger components, Graformer might also better distinguish nodes that are part of the same component from those that belong to a different one.

Only for $1.5 < \mu_c < 2.0$, CGE-LW clearly outperforms Graformer in all metrics. It seems that Graformer is most helpful for extreme cases, i.e., when either most components are isolated nodes or when isolated nodes are the exception.




(a) AGENDA (b) WebNLG

Figure 3: Attention bias $\gamma$ learned by Graformer on the two datasets. $\text{SAME}_p$ edges are omitted.

| Model | BLEU | METEOR | CHRF++ |
|---|---|---|---|
| Graformer | 18.09 | 22.29 | 45.77 |
| -length penalty | 17.99 | 22.19 | 45.63 |
| -beam search | 17.33 | 21.74 | 44.87 |
| -epoch curriculum | 13.55 | 18.91 | 39.22 |

Table 5: Ablation study for a single run on the test portion of AGENDA.

### 5.3 Ablation study

In a small ablation study, we examine the impact of beam search, length penalty, and our new epoch curriculum training. We find that beam search and length penalty do contribute to the overall performance but to a relatively small extent. Training with our new epoch curriculum, however, proves crucial for good performance. Platanios et al. (2019) argue that curriculum learning can replace a learning rate schedule, which is usually essential to train a Transformer model. Indeed we successfully optimize Graformer without any learning rate schedule, when applying the epoch curriculum.

## 6 Learned graph structure

We visualize the learned attention bias $\gamma$ for different relative graph positions $R_{ij}$ (cf. § 3.3; esp. Eq. (7)) after training on AGENDA and WebNLG in Fig. 3. The eight attention heads (x-axis) have learned different weights for each graph position $R_{ij}$ (y-axis). Note that AGENDA has more possible $R_{ij}$ values because $n_\delta = 6$ whereas we set $n_\delta = 4$ for WebNLG.

For both datasets, we notice that one attention head primarily focuses on global information (5 for AGENDA, 4 for WebNLG). AGENDA even dedicates head 6 entirely to unreachable nodes, showing the importance of such nodes for this dataset. In contrast, most WebNLG heads suppress information from unreachable nodes.

For both datasets, we also observe that nearer nodes generally receive a high weight (focus on local information): In Fig. 3b, e.g., head 2 concentrates solely on direct incoming edges and head 0 on direct outgoing ones. Graformer can learn empirically based on its task where direct neighbors are most important and where they are not, showing that the strong bias from graph neural networks is not necessary to learn about graph structure.

## 7 Conclusion

We presented Graformer, a novel encoder-decoder architecture for graph-to-text generation based on Transformer. The Graformer encoder uses a novel type of self-attention for graphs based on shortest path lengths between nodes, allowing it to detect global patterns by automatically learning appropriate weights for higher-order neighborhoods. In our experiments on two popular benchmarks for text generation from knowledge graphs, Graformer achieved competitive results while using many fewer parameters than alternative models.

## Acknowledgments

This work was supported by the BMBF (first author) as part of the project MLWin (01IS18050), by the German Research Foundation (second author) as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under the grant No. GRK 1994/1, and by the Bavarian research institute for digital transformation (bidt) through their fellowship program (third author). We also gratefully acknowledge a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes).

## References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91, New Orleans - Louisiana. Association for Computational Linguistics.

Bang An, Xuannan Dong, and Changyou Chen. 2019. Repulsive bayesian sampling for diversified attention modeling. *4th workshop on Bayesian Deep Learning (NeurIPS 2019)*.

Sören Auer, Chris Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2008. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc.

Rajarshi Bhowmik and Gerard de Melo. 2018. Generating fine-grained open vocabulary entity type descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 877–888, Melbourne, Australia. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. *AAAI Conference on Artificial Intelligence*.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *Computing Research Repository*, arXiv:1606.08415.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Qimai Li, Zhichao Han, and Xiao ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI Conference on Artificial Intelligence*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Computing Research Repository*, arXiv:1910.10683.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8(0):589–604.

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7117–7130, Online. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637, Melbourne, Australia. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, page 5998–6008. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. 2018. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Computing Research Repository*, arXiv:1609.08144.

Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. 2020. Adaptive structural fingerprints for graph attention networks. In *International Conference on Learning Representations (ICLR)*.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468, Hong Kong, China. Association for Computational Linguistics.

## A Hyperparameter details

For AGENDA and WebNLG, a minimum and maximum decoding length were set according to the shortest and longest target text in the train set. Table 6 lists the hyperparameters used to obtain final results on both datasets. Input dropout is applied on the word embeddings directly after lookup for node labels and target text tokens before they are fed into encoder or decoder. Attention dropout is applied to all attention weights computed during multi-head (self-)attention.

| Hyperparameter | WebNLG | AGENDA |
|---|---|---|
| model dimension $d$ | 256 | 400 |
| # heads | 8 | 8 |
| # encoder layers $L_E$ | 3 | 4 |
| # decoder layers $L_D$ | 3 | 5 |
| feedforward dimension | 512 | 2000 |
| attention dropout | 0.3 | 0.1 |
| dropout | 0.1 | 0.1 |
| input dropout | 0.0 | 0.1 |
| text self-attention range $n_t$ | 25 | 50 |
| graph self-attention range $n_\delta$ | 4 | 6 |
| SAME range $n_p$ | 10 | 10 |
| gradient accumulation | 3 | 2 |
| gradient clipping | 1.0 | 1.0 |
| label smoothing | 0.25 | 0.3 |
| $L_2$ regularizer | $3 \cdot 10^{-3}$ | $3 \cdot 10^{-4}$ |
| batch size | 4 | 8 |
| # beams | 2 | 2 |
| length penalty | 5.0 | 5.0 |

Table 6: Hyperparameters used to obtain final experimental results on WebNLG and AGENDA.

For hyperparameter optimization, we only train for the first 10 (AGENDA) or 50 (WebNLG) epochs to save time. We use a combination of manual tuning and a limited number of randomly sampled runs. For the latter we apply Optuna with default parameters (Akiba et al., 2019; Bergstra et al., 2011) and median pruning, i.e., after each epoch we check if the best performance so far is worse than the median performance of previous runs at the same epoch and if so, abort. For hyperparameter tuning, we decode greedily and measure performance in corpus-level BLEU (Papineni et al., 2002).

## B Qualitative examples

Table 7 shows three example generations from our Graformer model and the CGE-LW system by Ribeiro et al. (2020). Often CGE-LW generations have a high surface overlap with the reference text while Graformer texts fluently express the same content.

| | |
|---|---|
| Ref. | julia morgan has designed many significant buildings , including the los angeles herald examiner building . |
| CGE-LW | julia morgan has designed many significant buildings including the los angeles herald examiner building . |
| Ours | one of the significant buildings designed by julia morgan is the los angeles herald examiner building . |
| Ref. | asam pedas is a dish of fish cooked in a sour and hot sauce that comes from indonesia . |
| CGE-LW | the main ingredients of asam pedas are fish cooked in a sour and hot sauce and comes from indonesia . |
| Ours | the main ingredients of asam pedas are fish cooked in sour and hot sauce . the dish comes from indonesia . |
| Ref. | banana is an ingredient in binignit which is a dessert . a cookie is also a dessert . |
| CGE-LW | banana is an ingredient in binignit , a cookie is also a dessert . |
| Ours | a cookie is a dessert , as is binignit , which contains banana as one of its ingredients . |

Table 7: Example references and texts generated by CGE-LW (Ribeiro et al., 2020) and Graformer (marked Ours) for samples from the WebNLG test set. In case of multiple references, only one is shown for brevity.

# Chapter 7

# Investigating Pretrained Language Models for Graph-to-Text Generation

# Investigating Pretrained Language Models for Graph-to-Text Generation

**Leonardo F. R. Ribeiro**[†], **Martin Schmitt**[‡], **Hinrich Schütze**[‡] **and Iryna Gurevych**[†]

[†]Research Training Group AIPHES and UKP Lab, Technical University of Darmstadt
[‡]Center for Information and Language Processing (CIS), LMU Munich

www.ukp.tu-darmstadt.de

## Abstract

Graph-to-text generation aims to generate fluent texts from graph-based data. In this paper, we investigate two recent pretrained language models (PLMs) and analyze the impact of different task-adaptive pretraining strategies for PLMs in graph-to-text generation. We present a study across three graph domains: meaning representations, Wikipedia knowledge graphs (KGs) and scientific KGs. We show that approaches based on PLMs BART and T5 achieve new state-of-the-art results and that task-adaptive pretraining strategies improve their performance even further. We report new state-of-the-art BLEU scores of 49.72 on AMR-LDC2017T10, 59.70 on WebNLG, and 25.66 on AGENDA datasets - a relative improvement of 31.8%, 4.5%, and 42.4%, respectively, with our models generating significantly more fluent texts than human references. In an extensive analysis, we identify possible reasons for the PLMs' success on graph-to-text tasks. Our findings suggest that the PLMs benefit from similar facts seen during pretraining or fine-tuning, such that they perform well even when the input graph is reduced to a simple bag of node and edge labels.[1]

## 1 Introduction

Graphs are important data structures in NLP as they represent complex relations within a set of objects. For example, semantic and syntactic structures of sentences can be represented using different graph representations (e.g., AMRs, Banarescu et al., 2013; semantic-role labeling, Surdeanu et al., 2008; syntactic and semantic graphs, Belz et al., 2011) and knowledge graphs (KGs) are used to describe factual knowledge in the form of relations between entities (Gardent et al., 2017; Vougiouklis et al., 2018; Koncel-Kedziorski et al., 2019).

Graph-to-text generation, a subtask of data-to-text generation (Gatt and Krahmer, 2018), aims to create fluent natural language text to describe an input graph (see Figure 1). This task is important for NLP applications such as dialogue generation (Moon et al., 2019) and question answering (Duan et al., 2017). Recently, it has been shown that structured meaning representation, such as AMR or KG, can store the internal state of a dialog system, providing core semantic knowledge (Bonial et al., 2020; Bai et al., 2021) or can be the result of a database query for conversational QA (Yu et al., 2019). Moreover, dialog states can be represented as KGs to encode compositionality and can be shared across different domains, slot types and dialog participators (Cheng et al., 2020).

Transfer learning has become ubiquitous in NLP and pretrained Transformer-based architectures (Vaswani et al., 2017) have considerably outperformed prior state of the art in various downstream tasks (Devlin et al., 2019; Yang et al., 2019a; Liu et al., 2020; Radford et al., 2019).

In this paper, we analyze the applicability of two recent text-to-text pretrained language models (PLMs), BART (Lewis et al., 2020) and T5 (Raffel et al., 2019), for graph-to-text generation. We choose these models because of their *encoder-decoder* architecture, which makes them particularly suitable for conditional text generation. Our study comprises three graph domains (meaning representations, Wikipedia KGs, and scientific KGs). We further introduce *task-adaptive* graph-to-text pretraining approaches for PLMs and demonstrate that such strategies improve the state of the art by a substantial margin.

While recent works have shown the benefit of explicitly encoding the graph structure in graph-to-text generation (Song et al., 2018; Ribeiro et al., 2019, 2020; Schmitt et al., 2020; Zhao et al., 2020a, to name a few), our approaches based on PLMs consistently outperform these models, even though PLMs – as sequence models – do not exhibit any

[1]Our code is available at https://github.com/UKPLab/plms-graph2text.



*Proceedings of the Third Workshop on Natural Language Processing for Conversational AI*, pages 211–227
November 10, 2021. ©2021 Association for Computational Linguistics

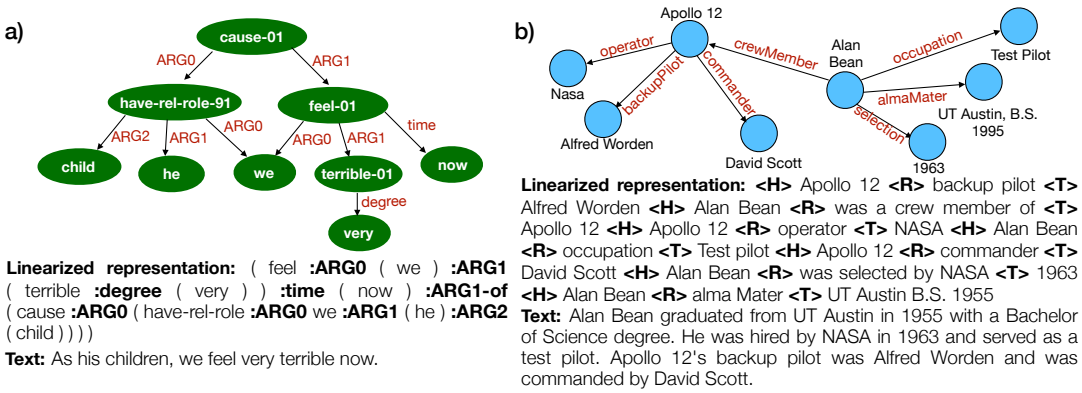



Figure 1: Examples of (a) AMR and (b) WebNLG graphs, the input for the models and the reference texts.

*graph-specific structural bias*.[2] Simply representing the graph as a linear traversal (see Figure 1) leads to remarkable generation performance in the presence of a strong language model. In our analysis we investigate to what extent fine-tuned PLMs make use of the graph structure represented in the graph linearization. We notably observe that PLMs achieve high performance on two popular KG-to-text benchmarks even when the KG is reduced to a mere bag of node and edge labels.

Our contributions are the following:

- We investigate and compare two PLMs, BART and T5, for graph-to-text generation, exploring *language model adaptation* (LMA) and *supervised task adaptation* (STA) pretraining, employing additional task-specific data.
- Our approaches consistently outperform the state of the art by significant margins, ranging from 2.6 to 12.0 BLEU points, on three established graph-to-text benchmarks from different domains, exceeding specialized graph architectures (e.g., Graph Neural Networks, GNNs, Kipf and Welling, 2017).
- In a crowdsourcing experiment, we demonstrate that our methods generate texts with significantly better fluency than existing works and the human references.
- We discover that PLMs perform well even when trained on a shuffled linearized graph representation without any information about connectivity (bag of node and edge labels), which is surprising since prior studies showed that explicitly encoding the graph structure improves models trained from scratch (e.g., Zhao et al., 2020a); and investigate the possible reasons for such a good performance.

[2]The model architecture does not explicitly encode the graph structure, i.e., which entities are connected to each other, but has to retrieve it from a sequence that tries to encode this information.

## 2 Related Work

**Graph-to-text Learning.** Various neural models have been proposed to generate sentences from graphs from different domains. Konstas et al. (2017) propose the first neural approach for AMR-to-text generation that uses a linearized input graph. Prior approaches for KG-to-text generation train text-to-text neural models using sequences of KG triples as input (Trisedya et al., 2018; Moryossef et al., 2019; Castro Ferreira et al., 2019; Ribeiro et al., 2021a).

Recent approaches (Marcheggiani and Perez Beltrachini, 2018; Song et al., 2018; Beck et al., 2018; Damonte and Cohen, 2019; Ribeiro et al., 2019; Zhao et al., 2020a; Schmitt et al., 2021; Ribeiro et al., 2021b) propose architectures based on GNNs to directly encode the graph structure, whereas other efforts (Ribeiro et al., 2020; Schmitt et al., 2020; Yao et al., 2020; Wang et al., 2020) inject the graph structure information into Transformer-based architectures. The success of those approaches suggests that imposing a strong relational inductive bias into the graph-to-text model can assist the generation.

**Pretrained Language Models.** Pretrained Transformer-based models, such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019b), or RoBERTa (Liu et al., 2020), have established a qualitatively new level of baseline performance for many widely used natural language understanding (NLU) benchmarks. Generative pretrained Transformer-based methods, such as GPT-2 (Radford et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2019), are employed in many

natural language generation (NLG) tasks.

Mager et al. (2020) were the first to employ GPT-2, a decoder-only PLM, for AMR-to-text generation and use cycle consistency to improve the adequacy. In contrast, we are the first to investigate BART and T5 models, which have both a Transformer-based encoder and decoder, in AMR-to-text generation. Recently, Harkous et al. (2020) and Kale (2020) demonstrate state-of-the-art results in different data-to-text datasets, employing GPT-2 and T5 models respectively. Radev et al. (2020) propose DART, a new data-to-text dataset, and train a BART model gradually augmenting the WebNLG training data with DART data.

Hoyle et al. (2021) explore scaffolding objectives in PLMs and show gains in low-resource graph-to-text settings. Different from the above works, we focus on a general transfer learning strategies for graph-to-text generation, investigating task-adaptive pretraining approaches, employing additional collected task-specific data for different PLMs (BART and T5) and benchmarks. In addition, we provide a detailed analysis aimed at explaining the good performance of PLMs on KG-to-text tasks.

Recently, Gururangan et al. (2020) explored task-adaptive pretraining strategies for text classification. While our LMA (see §3) is related to their DAPT as both use a self-supervised objective on a domain-specific corpus, they notably differ in that DAPT operates on the model input while LMA models the output. We are the first to show the benefits of additional task-specific pretraining in PLMs for graph-to-text tasks.

## 3 PLMs for Graph-to-Text Generation

### 3.1 Models in this Study

We investigate BART (Lewis et al., 2020) and T5 (Raffel et al., 2019), two PLMs based on the Transformer *encoder-decoder* architecture (Vaswani et al., 2017), for graph-to-text generation. They mainly differ in how they are pretrained and the input corpora used for pretraining. We experiment with different T5 (*small* - 60M parameters, *base* - 220M, and *large* - 770M) and BART (*base* - 140M and *large* - 400M) capacity models.

We fine-tune both PLMs for a few epochs on the supervised downstream graph-to-text datasets. For T5, in the supervised setup, we add a prefix "translate from Graph to Text:" before the graph input. We add this prefix to imitate the T5 setup, when translating between different languages.

### 3.2 Task-specific Adaptation

Inspired by previous work (Konstas et al., 2017; Gururangan et al., 2020), we investigate whether leveraging additional task-specific data can improve the PLMs' performance on graph-to-text generation. Task-specific data refers to a pretraining corpus that is more task-relevant and usually smaller than the text corpora used for task-independent pretraining. In order to leverage the task-specific data, we add an intermediate adaptive pretraining step between the original pretraining and fine-tuning phases for graph-to-text generation.

More precisely, we first continue pretraining BART and T5 using language model adaptation (LMA) or supervised task adaptation (STA) training. In the supervised approach, we use pairs of graphs and corresponding texts collected from the same or similar domain as the target task. In the LMA approach, we follow BART and T5 pretraining strategies for language modeling, using the reference texts that describe the graphs. Note that we do not use the graphs in the LMA pretraining, but only the target text of our task-specific data collections. The goal is to adapt the decoder to the domain of the final task (Gururangan et al., 2020). In particular, we randomly mask text spans, replacing 15% of the tokens.[3] Before evaluation, we finally fine-tune the models using the original training set as usual.

## 4 Datasets

We evaluate the text-to-text PLMs on three graph-to-text benchmarks: AMR (LDC2017T10), WebNLG (Gardent et al., 2017), and AGENDA (Koncel-Kedziorski et al., 2019). We chose those datasets because they comprise different domains and are widely used in prior work. Table 10 in Appendix shows statistics for each dataset.

**AMR.** Abstract meaning representation (AMR) is a semantic formalism that represents the meaning of a sentence as a rooted directed graph expressing "who is doing what to whom" (Banarescu et al., 2013). In an AMR graph, nodes represent concepts and edges represent semantic relations. An instance in LDC2017T10 consists of a sentence annotated with its corresponding AMR graph. Following Mager et al. (2020), we linearize the AMR graphs

[3] Please, refer to Lewis et al. (2020) and Raffel et al. (2019) for details about the self-supervised pretraining strategies.

using the PENMAN notation (see Figure 1a).[4]

**WebNLG.** Each instance of WebNLG contains a KG from DBPedia (Auer et al., 2007) and a target text with one or multiple sentences that describe the graph. The test set is divided into two partitions: *seen*, which contains only DBPedia categories present in the training set, and *unseen*, which covers categories never seen during training. Their union is called *all*. Following previous work (Harkous et al., 2020), we prepend $\langle H \rangle$, $\langle R \rangle$, and $\langle T \rangle$ tokens before the head entity, the relation and tail entity of a triple (see Figure 1b).

**AGENDA.** In this dataset, KGs are paired with scientific abstracts extracted from proceedings of AI conferences. Each sample contains the paper title, a KG, and the corresponding abstract. The KG contains entities corresponding to scientific terms and the edges represent relations between these entities. This dataset has loose alignments between the graph and the corresponding text as the graphs were automatically generated. The input for the models is a text containing the title, a sequence of all KG entities, and the triples. The target text is the paper abstract. We add special tokens into the triples in the same way as for WebNLG.

### 4.1 Additional Task-specific Data

In order to evaluate the proposed task-adaptive pretraining strategies for graph-to-text generation, we collect task-specific data for two graph domains: meaning representations (like AMR) and scientific data (like AGENDA). We did not attempt collecting additional data like WebNLG because the texts in this benchmark do not stem from a corpus but were specifically written by annotators.

**AMR Silver Data.** In order to generate additional data for AMR, we sample two sentence collections of size 200K and 2M from the Gigaword[5] corpus and use a state-of-the-art AMR parser (Cai and Lam, 2020a) to parse them into AMR graphs.[6] For supervised pretraining, we condition a model on the AMR silver graphs to generate the corresponding sentences before fine-tuning it on gold AMR graphs. For self-supervised pretraining, we only use the sentences.[7]

[4]Details of the preprocessing procedure of AMRs are provided in Appendix A.

[5]https://catalog.ldc.upenn.edu/LDC2003T05

[6]We filter out sentences that do not yield well-formed AMR graphs.

[7]Gigaword and AMR datasets share similar data sources.

| Model | BLEU | M | BT |
|---|---|---|---|
| Ribeiro et al. (2019) | 27.87 | 33.21 | - |
| Zhu et al. (2019) | 31.82 | 36.38 | - |
| Zhao et al. (2020b) | 32.46 | 36.78 | - |
| Wang et al. (2020) | 33.90 | 37.10 | - |
| Yao et al. (2020) | 34.10 | 38.10 | - |
| *based on PLMs* | | | |
| Mager et al. (2020) | 33.02 | 37.68 | - |
| Harkous et al. (2020) | 37.70 | 38.90 | - |
| $\text{BART}_{\text{base}}$ | 36.71 | 38.64 | 52.47 |
| $\text{BART}_{\text{large}}$ | 43.47 | 42.88 | 60.42 |
| $\text{T5}_{\text{small}}$ | 38.45 | 40.86 | 57.95 |
| $\text{T5}_{\text{base}}$ | 42.54 | 42.62 | 60.59 |
| $\text{T5}_{\text{large}}$ | **45.80** | **43.85** | **61.93** |
| *with task-adaptive pretraining* | | | |
| $\text{BART}_{\text{large}}$ + LMA | 43.94 | 42.36 | 58.54 |
| $\text{T5}_{\text{large}}$ + LMA | 46.06 | 44.05 | 62.59 |
| $\text{BART}_{\text{large}}$ + STA (200K) | 44.72 | 43.65 | 61.03 |
| $\text{BART}_{\text{large}}$ + STA (2M) | 47.51 | 44.70 | 62.27 |
| $\text{T5}_{\text{large}}$ + STA (200K) | 48.02 | 44.85 | 63.86 |
| $\text{T5}_{\text{large}}$ + STA (2M) | ***49.72*** | ***45.43*** | ***64.24*** |

Table 1: Results on AMR-to-text generation for the LDC2017T10 test set. M and BT stand for METEOR and BLEURT, respectively. **Bold** (***Italic***) indicates the best score without (with) task-adaptive pretraining.

**Semantic Scholar AI Data.** We collect titles and abstracts of around 190K scientific papers from the Semantic Scholar (Ammar et al., 2018) taken from the proceedings of 36 top Computer Science/AI conferences. We construct KGs from the paper abstracts employing DyGIE++ (Wadden et al., 2019), an information extraction system for scientific texts. Note that the AGENDA dataset was constructed using the older SciIE system (Luan et al., 2018), which also extracts KGs from AI scientific papers. A second difference is that in our new dataset, the domain is broader as we collected data from 36 conferences compared to 12 from AGENDA. Furthermore, to prevent data leakage, all AGENDA samples used for performance evaluation are removed from our dataset. We will call the new dataset KGAIA (KGs from AI Abstracts).[8] Table 11 in Appendix shows relevant dataset statistics.

## 5 Experiments

We modify the BART and T5 implementations released by Hugging Face (Wolf et al., 2019) in order to adapt them to graph-to-text generation. For the KG datasets, we add the $\langle H \rangle$, $\langle R \rangle$, and $\langle T \rangle$ tokens to the models' vocabulary. We add all edge labels seen in the training set to the vocabulary of the

[8]We will release the collected additional task-specific data.

| | BLEU | | | METEOR | | | chrF++ | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | **A** | **S** | **U** | **A** | **S** | **U** | **A** | **S** | **U** |
| Castro Ferreira et al. (2019) | 51.68 | 56.35 | 38.92 | 32.00 | 41.00 | 21.00 | - | - | - |
| Moryossef et al. (2019) | 47.24 | 53.30 | 34.41 | 39.00 | 44.00 | 37.00 | - | - | - |
| Schmitt et al. (2020) | - | 59.39 | - | - | 42.83 | - | - | 74.68 | - |
| Ribeiro et al. (2020) | - | 63.69 | - | - | 44.47 | - | - | 76.66 | - |
| Zhao et al. (2020a) | 52.78 | 64.42 | 38.23 | 41.00 | 46.00 | 37.00 | - | - | - |
| *based on PLMs* | | | | | | | | | |
| Harkous et al. (2020) | 52.90 | - | - | 42.40 | - | - | - | - | - |
| Kale (2020) | 57.10 | 63.90 | 52.80 | 44.00 | 46.00 | 41.00 | - | - | - |
| Radev et al. (2020) | 45.89 | 52.86 | 37.85 | 40.00 | 42.00 | 37.00 | - | - | - |
| $\text{BART}_{\text{base}}$ | 53.11 | 62.74 | 41.53 | 40.18 | 44.45 | 35.36 | 70.02 | 76.68 | 62.76 |
| $\text{BART}_{\text{large}}$ | 54.72 | 63.45 | 43.97 | 42.23 | 45.49 | 38.61 | 72.29 | 77.57 | 66.53 |
| $\text{T5}_{\text{small}}$ | 56.34 | **65.05** | 45.37 | 42.78 | 45.94 | 39.29 | 73.31 | **78.46** | 67.69 |
| $\text{T5}_{\text{base}}$ | 59.17 | 64.64 | 52.55 | 43.19 | **46.02** | 41.49 | 74.82 | 78.40 | 70.92 |
| $\text{T5}_{\text{large}}$ | **59.70** | 64.71 | **53.67** | **44.18** | 45.85 | **42.26** | **75.40** | 78.29 | **72.25** |

Table 2: Results on WebNLG. A, S and U stand for *all*, *seen*, and *unseen* partitions of the test set, respectively.

models for AMR. Following Wolf et al. (2019), we use the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of $3 \cdot 10^{-5}$. We employ a linearly decreasing learning rate schedule without warm-up. The batch and beam search sizes are chosen from {2,4,8} and {1,3,5}, respectively, based on the respective development set. Dev BLEU is used for model selection.

Following previous works, we evaluate the results with BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), and chrF++ (Popović, 2015) metrics. We also use MoverScore (Zhao et al., 2019), BERTScore (Zhang et al., 2020), and BLEURT (Sellam et al., 2020) metrics, as they employ contextual and semantic knowledge and thus depend less on the surface symbols. Additionally, we perform a human evaluation (cf. §5.4) quantifying the fluency, semantic adequacy and meaning similarity of the generated texts.

### 5.1 Results on AMR-to-Text

Table 1 shows our results for the setting without additional pretraining, with additional self-supervised task-adaptive pretraining solely using the collected Gigaword sentences (LMA), and with additional supervised task adaptation (STA), before fine-tuning. We also report several recent results on the AMR test set. Mager et al. (2020) and Harkous et al. (2020) employ GPT-2 in their approaches. Note that GPT-2 only consists of a Transformer-based decoder.

Only considering approaches without task adaptation, $\text{BART}_{\text{large}}$ already achieves a considerable improvement of 5.77 BLEU and 3.98 METEOR scores over the previous state of the art. With a BLEU score of 45.80, $\text{T5}_{\text{large}}$ performs best. The other metrics follow similar trends. See Table 13 in Appendix for evaluation with more metrics. The strong performance of both BART and T5 in the AMR dataset suggests that PLMs can infer the AMR structure by a simple linear sequence of the graph, in contrast to GNN-based models that explicitly consider the graph structure using *message-passing* between adjacent nodes (Beck et al., 2018).

**Task-specific Adaptation.** LMA already brings some gains with T5 benefitting more than BART in most metrics. It still helps less than STA even though we only have automatically generated annotations. This suggests that the performance increases with STA do not only come from additional exposure to task-specific target texts and that the models learn how to handle graphs and the graph-text correspondence even with automatically generated AMRs. After STA, T5 achieves 49.72 BLEU points, the new state of the art for AMR-to-text generation. Interestingly, gains from STA with 2M over 200K are larger in BART than in T5, suggesting that large amounts of silver data may not be required for a good performance with T5.

In general, models pretrained on the STA setup converge faster than without task-specific adaptation. For example, $\text{T5}_{\text{large}}$ without additional pretraining converges after 5 epochs of fine-tuning whereas $\text{T5}_{\text{large}}$ with STA already converges after 2 epochs.

### 5.2 Results on WebNLG

Table 2 shows the results for the WebNLG test set. Neural pipeline models (Moryossef et al., 2019; Castro Ferreira et al., 2019) achieve strong performance in the *unseen* dataset. On the other

| Model | BLEU | M | BT |
|---|---|---|---|
| Koncel et al. 2019 | 14.30 | 18.80 | - |
| An (2019) | 15.10 | 19.50 | - |
| Schmitt et al. (2020) | 17.33 | 21.43 | - |
| Ribeiro et al. (2020) | 18.01 | 22.23 | - |
| $\text{BART}_{\text{base}}$ | 22.01 | 23.54 | -13.02 |
| $\text{BART}_{\text{large}}$ | **23.65** | **25.19** | **-10.93** |
| $\text{T5}_{\text{small}}$ | 20.22 | 21.62 | -24.10 |
| $\text{T5}_{\text{base}}$ | 20.73 | 21.88 | -21.03 |
| $\text{T5}_{\text{large}}$ | 22.15 | 23.73 | -13.96 |
| *with task-adaptive pretraining* | | | |
| $\text{BART}_{\text{large}}$ + LMA | 25.30 | 25.54 | -08.79 |
| $\text{T5}_{\text{large}}$ + LMA | 22.92 | 24.40 | -10.39 |
| $\text{BART}_{\text{large}}$ + STA | ***25.66*** | ***25.74*** | ***-08.97*** |
| $\text{T5}_{\text{large}}$ + STA | 23.69 | 24.92 | -08.94 |

Table 3: Results on AGENDA test set. **Bold** (***Italic***) indicates best scores without (with) task-adaptive pretraining.

| Model | AMR | |
|---|---|---|
| | **F** | **MS** |
| Mager et al. (2020) | $5.69^{A}$ | $5.08^{A}$ |
| Harkous et al. (2020) | $5.78^{A}$ | $5.47^{AB}$ |
| $\text{T5}_{\text{large}}$ | $6.55^{B}$ | $6.44^{C}$ |
| $\text{BART}_{\text{large}}$ | $6.70^{B}$ | $5.72^{BC}$ |
| Reference | $5.91^{A}$ | - |
| **Model** | **WebNLG** | |
| | **F** | **SA** |
| Castro Ferreira et al. (2019) | $5.52^{A}$ | $4.77^{A}$ |
| Harkous et al. (2020) | $5.74^{AB}$ | $6.21^{B}$ |
| $\text{T5}_{\text{large}}$ | $6.71^{C}$ | $6.63^{B}$ |
| $\text{BART}_{\text{large}}$ | $6.53^{C}$ | $6.50^{B}$ |
| Reference | $5.89^{B}$ | $6.47^{B}$ |

Table 4: Fluency (F), Meaning Similarity (MS) and Semantic Adequacy (SA) obtained in the human evaluation. Differences between models which have a letter in common are not statistically significant and were determined by pairwise Mann-Whitney tests with $p < 0.05$.

hand, fully end-to-end models (Ribeiro et al., 2020; Schmitt et al., 2020) have strong performance on the *seen* dataset and usually perform poorly in *unseen* data. Models that *explicitly encode the graph structure* (Ribeiro et al., 2020; Zhao et al., 2020a) achieve the best performance among approaches that do not employ PLMs. Note that T5 is also used in Kale (2020). Differences in our T5 setup include a modified model vocabulary, the use of beam search, the learning rate schedule and the prefix before the input graph. Our T5 approach achieves 59.70, 65.05 and 54.69 BLEU points on *all*, *seen* and *unseen* sets, the new state of the art.

We conjecture that the performance gap between *seen* and *unseen* sets stems from the advantage obtained by a model seeing examples of relation-text pairs during fine-tuning. For example, the relation *party* (political party) was never seen during training and the model is required to generate a text that verbalizes the tuple: ⟨*Abdul Taib Mahmud, party, Parti Bumiputera Sarawak*⟩. Interestingly, BART performs much worse than T5 on this benchmark, especially in the *unseen* partition with 9.7 BLEU points lower compared to T5.

For lack of a suitable data source (cf. §4), we did not explore our LMA or STA approaches for WebNLG. However, we additionally discuss cross-domain STA in Appendix B.

### 5.3 Results on AGENDA

Table 3 lists the results for the AGENDA test set. The models also show strong performance on this dataset. We believe that their capacity to generate fluent text helps when generating paper abstracts, even though they were not pretrained in the scientific domain. $\text{BART}_{\text{large}}$ shows an impressive performance with a BLEU score of 23.65, which is 5.6 points higher than the previous state of the art.

**Task-specific Adaptation.** On AGENDA, BART benefits more from our task-adaptive pretraining, achieving the new state of the art of 25.66 BLEU points, a further gain of 2 BLEU points compared to its performance without task adaptation. The improvements from task-adaptive pretraining are not as large as for AMR. We hypothesize that this is due to the fact that the graphs do not completely cover the target text (Koncel-Kedziorski et al., 2019), making this dataset more challenging. See Table 12 in Appendix for more automatic metrics.

### 5.4 Human Evaluation

To further assess the quality of the generated text, we conduct a human evaluation on AMR and WebNLG via crowd sourcing on Amazon Mechanical Turk.[9] Following previous works (Gardent et al., 2017; Castro Ferreira et al., 2019), we assess three quality criteria: (i) *Fluency* (i.e., does the text flow in a natural, easy-to-read manner?), for AMR and WebNLG; (ii) *Meaning Similarity* (i.e., how

[9]We exclude AGENDA because its texts are scientific in nature and annotators are not necessarily AI experts.

**Original Input**

• Arrabbiata sauce • country • Italy • Italy • demonym • Italians • Italy • capital • Rome • Italy • language • Italian language • Italy • leader Name • Sergio Mattarella

Shuffle

**Corrupted Input**

• Rome • Italy • Italy • language • capital • Italy • Italians • Italy • Italy • Sergio Mattarella • Arrabbiata sauce • leader Name • country • demonym • Italian language

T5$^{order}$

Arrabbiata sauce can be found in Italy where Sergio Mattarella is the leader and the capital city is Rome. Italians are the people who live there and the language spoken is Italian.

T5$^{shuf}$

Italians live in Italy where the capital is Rome and the language is Italian. Sergio Mattarella is the leader of the country and arrabbiata sauce can be found there.

**Reference:** Arrabbiata sauce is from Italy where the capital is Rome, Italian is the language spoken and Sergio Mattarella is a leader.

Figure 2: Example graph with 5 triples, from WebNLG dev linearized with the neutral separator tag, denoted •, (top left), its shuffled version (top right), texts generated with two fine-tuned versions of $T5_{small}$ and a gold reference (bottom). Note that T5 can produce a reasonable text even when the input triples are shuffled randomly.

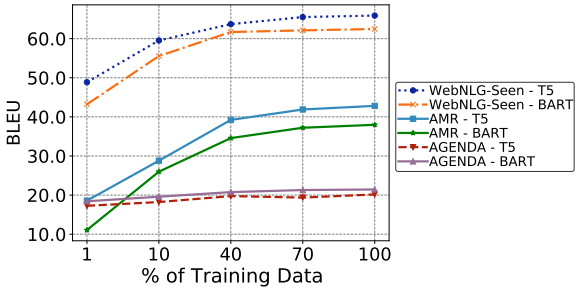



Figure 3: Performance of $BART_{base}$ and $T5_{base}$ in the dev set when experimenting with different amounts of training data.

close in meaning is the generated text to the reference sentence?) for AMR; (ii) *Semantic Adequacy* (i.e., does the text clearly express the data?) for WebNLG. We randomly select 100 generated texts of each model, which the annotators then rate on a 1-7 Likert scale. For each text, we collect scores from 3 annotators and average them.[10]

Table 4 shows the results. Our approaches improve the fluency, meaning similarity, and semantic adequacy on both datasets compared to other state-of-the-art approaches with statistically significant margins ($p<0.05$). Interestingly, the highest fluency improvement (+0.97) is on AMR, where our approach also has the largest BLEU improvement (+8.10) over Harkous et al. (2020). Finally, our models score higher than the references in fluency with statistically significant margins, highlighting their strong language generation abilities.[11]

### 5.5 Limiting the Training Data

In Figure 3, we investigate the PLMs' performance, measured with BLEU score, while varying (from 1% to 100%) the amount of training data used for fine-tuning. We find that, when fine-tuned with only 40% of the data, both BART and T5 already greatly improve the performance compared to using the entire training data in all three benchmarks. For example, BART fine-tuned on 40% of AMR training data achieves 91% of the BLEU score when fine-tuned on full data.

Note that in a low-resource scenario in AMR and WebNLG, T5 considerably outperforms BART. In particular, with only 1% of training examples, the difference between T5 and BART is 7.51 and 5.64 BLEU points for AMR and WebNLG, respectively. This suggests that T5 is more data efficient when adapting to the new task, likewise our findings in AMR-STA (cf. §5.1).

| Model | AMR | WebNLG | AGENDA |
|---|---|---|---|
| T5$^{order}$ | 36.83 | 63.41 | 19.86 |
| T5$^{shuf}$ | 15.56 | 61.54 | 19.08 |

Table 5: Impact (measured with BLEU) of using a bag of entities and relations (*shuf*) as input for $T5_{small}$.

## 6 Influence of the Graph Structure

We conduct further experiments to examine how much the PLMs consider the graph structure. To this end, we remove parentheses in AMRs and replace $\langle H \rangle$, $\langle R \rangle$, and $\langle T \rangle$ tokens with neutral separator tokens, denoted •, for KGs, such that the graph structure is only defined by the order of node and edge labels. If we shuffle such a sequence, the graph structure is thus completely obscured and the input effectively becomes a bag of node and edge labels. See Figure 2 for an example of both a correctly ordered and a shuffled triple sequence.

### 6.1 Quantitative Analysis

Table 5 shows the effect on T5's performance when its input contains correctly ordered triples (T5$^{order}$)

[10] Inter-annotator agreement for the three criteria ranged from 0.40 to 0.79, with an average Krippendorff's $\alpha$ of 0.56.

[11] Examples of fluent generations can be found in the Tables 15 and 16 in Appendix.

| | T/F | Input Fact | T5$^{order}$ | T5$^{shuf}$ |
|---|---|---|---|---|
| (1) | S | • German language • Antwerp • Antwerp • Antwerp International Airport • Belgium • Belgium • Charles Michel • city Served • leader Name • Belgium • language • country | Antwerp International Airport serves the city of Antwerp. German is the language spoken in Belgium where Charles Michel is the leader. | Antwerp International Airport serves the city of Antwerp in Belgium where the German language is spoken and Charles Michel is the leader. |
| (2) | T | • California • is Part Of • US • California • capital • Sacramento | California is part of the United States and its capital is Sacramento. | California is part of the United States and its capital is Sacramento. |
| (3) | F | • US • is Part Of • California • California • capital • Sacramento | California's capital is Sacramento and the United States is part of California. | California is part of the United States and its capital is Sacramento. |
| (4) | T | • Amarillo, Texas • is Part Of • United States | Amarillo, Texas is part of the United States. | Amarillo, Texas is part of the United States. |
| (5) | F | • United States • is Part Of • Amarillo, Texas | Amarillo, Texas is part of the United States. | Amarillo, Texas is part of the United States. |

Table 6: Example generations from shuffled (S), true (T), and corrupted (F) triple facts by $T5_{small}$, fine-tuned on correctly ordered triples (*order*) and randomly shuffled input (*shuf*).

vs. shuffled ones (T5$^{shuf}$) for both training and evaluation. We first observe that T5$^{order}$ only has marginally lower performance (around 2-4%) with the neutral separators than with the $\langle H\rangle/\langle R\rangle/\langle T\rangle$ tags or parentheses.[12] We see that as evidence that the graph structure is similarly well captured by T5$^{order}$. Without the graph structure (T5$^{shuf}$), AMR-to-text performance drops significantly. Possible explanations of this drop are: (i) the relative ordering of the AMR graph is known to correlate with the target sentence order (Konstas et al., 2017); (ii) in contrast to WebNLG that contains common knowledge, the AMR dataset contains very specific sentences with higher surprisal;[13] (iii) AMRs are much more complex graph structures than the KGs from WebNLG and AGENDA.[14]

On the other hand, KG-to-text performance is not much lower, indicating that most of the PLMs' success in this task stems from their language modeling rather than their graph encoding capabilities. We hypothesize that a PLM can match the entities in a shuffled input with sentences mentioning these entities from the pretraining or fine-tuning phase. It has recently been argued that large PLMs can recall certain common knowledge facts from pretraining (Petroni et al., 2019; Bosselut et al., 2019).

### 6.2 Qualitative Analysis

The example in Figure 2 confirms our impression. T5$^{shuf}$ produces a text with the same content as T5$^{order}$ but does not need the correct triple structure to do so. Example (1) in Table 6 shows the output of both models with shuffled input. Interestingly, even T5$^{order}$ produces a reasonable and truthful text. This suggests that previously seen facts serve as a strong guide during text generation, even for models that were fine-tuned with a clearly marked graph structure, suggesting that T5$^{order}$ also relies more on language modeling than the graph structure. It does have more difficulties covering the whole input graph though. The fact that *Antwerp* is located in *Belgium* is missing from its output.

To further test our hypothesis that PLMs make use of previously seen facts during KG-to-text generation, we generate example true facts, corrupt them in a controlled setting, and feed them to both T5$^{order}$ and T5$^{shuf}$ to observe their output (examples (2)–(5) in Table 6). The model trained on correctly ordered input has learned a bit more to rely on the input graph structure. The false fact in example (3) with two triples is reliably transferred to the text by T5$^{order}$ but not by T5$^{shuf}$, which silently corrects it. Also note that, in example (5), both models refuse to generate an incorrect fact. More examples can be found in Table 14 in the Appendix.

Our qualitative analysis illustrates that state-of-the-art PLMs, despite their fluency capacities (cf. §5.4), bear the risk of parroting back training sentences while ignoring the input structure. This issue can limit the practical usage of those models as, in many cases, it is important for a generation model to stay true to its input (Wiseman et al., 2017; Falke et al., 2019).

[12] See a more fine-grained comparison in Appendix C.

[13] Perplexities estimated on the dev sets of AMR and WebNLG datasets, with GPT-2 fine-tuned on the corresponding training set, are 20.9 and 7.8, respectively.

[14] In Appendix D, we present the graph properties of the datasets and discuss the differences.

## 7 Conclusion

We investigated two pretrained language models (PLMs) for graph-to-text generation and show that the pretraining strategies, language model adaptation (LMA) and supervised task adaptation (STA), can lead to notable improvements. Our approaches outperform the state of the art by a substantial margin on three graph-to-text benchmarks. Moreover, in a human evaluation our generated texts are perceived significantly more fluent than human references. Examining the influence of the graph structure on the text generation process, we find that PLMs may not always follow the graph structure and instead use memorized facts to guide the generation. A promising direction for future work is to explore ways of injecting a stronger graph-structural bias into PLMs, thus possibly leveraging their strong language modeling capabilities and keeping the output faithful to the input graph.

## Acknowledgments

We thank our anonymous reviewers for their thoughtful feedback. Leonardo F. R. Ribeiro is supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1) and as part of the DFG funded project UKP-SQuARE with the number GU 798/29-1. Martin Schmitt is supported by the BMBF as part of the project MLWin (01IS18050) and by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes).

## References

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, Rodney Kinney, Sebastian Kohlmeier, Kyle Lo, Tyler Murray, Hsu-Han Ooi, Matthew Peters, Joanna Power, Sam Skjonsberg, Lucy Wang, Chris Wilhelm, Zheng Yuan, Madeleine van Zuylen, and Oren Etzioni. 2018. Construction of the literature graph in semantic scholar. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 84–91, New Orleans - Louisiana. Association for Computational Linguistics.

Bang An. 2019. Repulsive bayesian sampling for diversified attention modeling. In *4th workshop on Bayesian Deep Learning (NeurIPS 2019)*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg. Springer-Verlag.

Xuefeng Bai, Yulong Chen, Linfeng Song, and Yue Zhang. 2021. Semantic representation for dialogue modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4430–4445, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020a. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020b. Graph transformer for graph-to-sequence learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7464–7471. AAAI Press.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghdha, and Anders Johannsen. 2020. Conversational semantic parsing for dialog state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117, Online. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61(1):65–170.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv e-prints*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR 2017.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized bert pretraining approach. *arXiv e-prints*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

Diego Marcheggiani and Laura Perez Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 1–9, Tilburg University, The Netherlands. Association for Computational Linguistics.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, Florence, Italy. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Nazneen Fatema Rajani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, and Richard Socher. 2020. Dart: Open-domain structured data record to text generation. *arXiv e-prints*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *arXiv e-prints*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Jonas Pfeiffer, Yue Zhang, and Iryna Gurevych. 2021a. Smelting gold and silver for improved multilingual amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

Leonardo F. R. Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. *Transactions of the Association for Computational Linguistics*, 8:589–604.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021b. Structural adapters in pretrained language models for amr-to-text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Punta Cana, November 7-11, 2021*.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2021. Modeling graph structure via relative position for text generation from knowledge graphs. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 10–21, Mexico City, Mexico. Association for Computational Linguistics.

Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2020. Modeling graph structure via relative position for better text generation from knowledge graphs. *arXiv e-prints*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Melbourne, Australia. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637, Melbourne, Australia. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52-53:1 – 15.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019a. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

*Linguistics*, pages 7145–7154, Online. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020a. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online. Association for Computational Linguistics.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020b. Line graph enhanced AMR-to-text generation with mix-order graph attention networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 732–741, Online. Association for Computational Linguistics.

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019. Modeling graph structure in transformer for better AMR-to-text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5459–5468, Hong Kong, China. Association for Computational Linguistics.

## Appendices

In this supplementary material, we provide: (i) additional information about the data used in the experiments, and (ii) results that we could not fit into the main body of the paper.

## A AMR Input Representation

We test three variants for the representation of the input AMR graph. Following previous work (Konstas et al., 2017; Mager et al., 2020), we evaluate (i) only node representation, where the edge information is removed from the linearization; (ii) depth-first search (DFS) through the graph and the (iii) PENMAN representation. An example for each representation is illustrated below:

only nodes
```
value interrogative commodity
true
```

DFS
```
value :mode interrogative
:ARG1 commodity :ARG1-of
true
```

PENMAN
```
( value :mode interrogative
:ARG1 ( commodity ) :ARG1-of
( true ) )
```

In this experiment we employ $T5_{small}$. Table 7 shows the results on the AMR development set. The PENMAN representation leads to best results. Therefore, this representation is used in the rest of the experiments.

| Input | BLEU |
|---|---|
| only nodes | 28.22 |
| DFS | 34.94 |
| PENMAN | 38.27 |

Table 7: Results on the AMR dev set using $T5_{small}$ for different AMR linearizations.

## B Cross-domain Adaptation

For a given task, it is not always possible to collect closely related data – as we saw, e.g., for WebNLG. We therefore report STA in a cross-domain setting for the different KG-to-text benchmarks. Table 8 shows the results using $BART_{base}$ and $T5_{base}$. While the texts in KGAIA and AGENDA share the domain of scientific abstracts, texts in WebNLG are more general. Also note that WebNLG graphs do not share any relations with the other KGs. For $BART_{base}$, STA increases the performance in the cross-domain setting in most of the cases. For $T5_{base}$, STA in KGAIA improves the performance on WebNLG.

In general, we find that exploring additional adaptive pretraining for graph-to-text generation can improve the performance even if the data do not come from the same domain.

| STA on | Fine-tuned & Evaluated on | |
|---|---|---|
| | WebNLG-*Seen* | AGENDA |
| | $BART_{base}$ | |
| None | 58.71 | 22.01 |
| KGAIA | 63.20 | 23.48 |
| WebNLG | - | 21.98 |
| AGENDA | 61.25 | - |
| | $T5_{base}$ | |
| None | 62.93 | 20.73 |
| KGAIA | 63.19 | 22.44 |
| WebNLG | - | 20.27 |
| AGENDA | 62.75 | - |

Table 8: Effect (measured with BLEU score) of cross-domain STA.

## C Input Graph Size

Figure 4 visualizes $T5_{small}$'s performance with respect to the number of input graph triples in WebNLG dataset. We observe that $T5^{order}$ and $T5^{shuf}$ perform similarly for inputs with only one triple but that the gap between the models increases with larger graphs. While it is obviously more difficult to reconstruct a larger graph than a smaller one, this also suggests that the graph structure is more taken into account for graphs with more than 2 triples. For the *unseen* setting, the performance gap for these graphs is even larger, suggesting that the PLM can make more use of the graph structure when it has to.

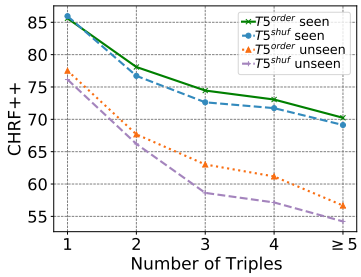



Figure 4: chrF++ scores with respect to the number of triples for WebNLG *seen* and *unseen* test sets.

## D Graph Statistics

In Table 9, we present the graph properties of the three datasets. All statistics are calculated using

| | AMR | | | WebNLG | | | AGENDA | | |
|---|---|---|---|---|---|---|---|---|---|
| min, avg and max number of nodes | 2 | 28.6 | 335 | 2 | 6.8 | 15 | 2 | 10.5 | 80 |
| min, avg and max node degrees | 1 | 2.2 | 21 | 1 | 1.7 | 7 | 1 | 1.67 | 15 |
| min, avg and max number of edges | 1 | 32.3 | 554 | 1 | 5.9 | 14 | 1 | 8.8 | 124 |
| min, avg and max graph diameter | 1 | 12.2 | 40 | 1 | 4.1 | 10 | 1 | 3.1 | 20 |
| min, avg and max shortest path length | 0 | 7.49 | 40 | 0 | 2.4 | 10 | 0 | 2.3 | 20 |

Table 9: Graph statistics of AMR, WebNLG and AGENDA datasets. The values are calculated using the training data. Note that AMR graphs contain a more complex structure than WebNLG and AGENDA graphs.

the Levi transformation (Beck et al., 2018) of the undirected version of the graphs, where edges are also considered nodes in the graph. WebNLG and AGENDA datasets contain disconnected graphs, and we use the largest subgraph to calculate the diameter. Note that AMR graphs have a much more complex structure: (i) they have more nodes and edges than WebNLG and AGENDA graphs; (ii) the average graph diameter and the average shortest path between nodes in AMRs are at least three times larger than in WebNLG and AGENDA graphs; (iii) nodes in AMRs have larger degrees than nodes in WebNLG and AGENDA graphs.

| | AMR17 | WebNLG | AGENDA |
|---|---|---|---|
| #Train | 36,521 | 18,102 | 38,720 |
| #Dev | 1,368 | 872 | 1,000 |
| #Test | 1,371 | 1,862 | 1,000 |
| #Relations | 155 | 373 | 7 |
| Avg #Tokens | 16.1 | 31.5 | 157.9 |

Table 10: Statistics for the graph-to-text benchmarks.

| | Title | Abstract | KG |
|---|---|---|---|
| Vocab | 48K | 173K | 113K |
| Tokens | 2.1M | 31.7M | 9.6M |
| Entities | - | - | 3.7M |
| Avg Length | 11.1 | 167.1 | - |
| Avg #Nodes | - | - | 19.9 |
| Avg #Edges | - | - | 9.4 |

Table 11: Statistics for the KGAIA dataset.

| Model | chrF++ | BS (F1) | MS |
|---|---|---|---|
| Schmitt et al. (2020) | 44.53 | - | - |
| Ribeiro et al. (2020) | 46.37 | - | - |
| $BART_{base}$ | 48.02 | 89.36 | 34.33 |
| $BART_{large}$ | **50.44** | 88.74 | 32.24 |
| $T5_{small}$ | 44.91 | 88.56 | 30.25 |
| $T5_{base}$ | 48.14 | 88.81 | 31.33 |
| $T5_{large}$ | 48.14 | **89.60** | **35.23** |
| *with task-adaptive pretraining* | | | |
| $BART_{large}$ + LMA | 51.33 | 89.12 | 33.42 |
| $T5_{large}$ + LMA | 49.37 | 89.75 | 36.13 |
| $BART_{large}$ + STA | ***51.63*** | 89.27 | 34.28 |
| $T5_{large}$ + STA | 50.27 | ***89.93*** | ***36.86*** |

Table 12: Results of the chrF++, BertScore (BS) and MoverScore (MS) scores for AGENDA test set. **Bold** (***Italic***) indicates best scores without (with) task-adaptive pretraining.

| Model | chrF++ | BS (F1) | MS |
|---|---|---|---|
| Guo et al. (2019) | 57.30 | - | - |
| Zhu et al. (2019) | 64.05 | - | - |
| Cai and Lam (2020b) | 59.40 | - | - |
| Wang et al. (2020) | 65.80 | - | - |
| Yao et al. (2020) | 65.60 | - | - |
| *based on PLMs* | | | |
| Mager et al. (2020) | 63.89 | - | - |
| $BART_{base}$ | 66.65 | 95.22 | 60.78 |
| $BART_{large}$ | 71.06 | 96.08 | 65.74 |
| $T5_{small}$ | 68.78 | 95.62 | 63.70 |
| $T5_{base}$ | 70.81 | 95.99 | 65.63 |
| $T5_{large}$ | **72.57** | **96.27** | **67.37** |
| *with task-adaptive pretraining* | | | |
| $BART_{large}$ + LMA | 71.14 | 95.94 | 64.75 |
| $T5_{large}$ + LMA | 72.83 | 96.32 | 67.44 |
| $BART_{large}$ + STA (200K) | 72.26 | 96.21 | 66.75 |
| $BART_{large}$ + STA (2M) | 73.58 | 96.43 | 68.14 |
| $T5_{large}$ + STA (200K) | 74.09 | 96.51 | 68.86 |
| $T5_{large}$ + STA (2M) | ***74.79*** | ***96.59*** | ***69.53*** |

Table 13: Results of the chrF++, BertScore (BS) and MoverScore (MS) scores for the LDC2017T10 test set. **Bold** (***Italic***) indicates the best score without (with) task-adaptive pretraining.

| | T/F | Input Facts | T5$^{order}$ | T5$^{shuf}$ |
|---|---|---|---|---|
| (1) | S | • capital • leader Name • London • Pound sterling • United Kingdom • leader Name • United Kingdom • Elizabeth II • United Kingdom • Boris Johnson • London • currency | The capital city is London, the currency is the Pound sterling and the leader is Elizabeth II. Boris Johnson is also a leader in the UK. | The capital of the United Kingdom is London, the currency is the Pound sterling and the country is lead by Elizabeth II and Boris Johnson. |
| (2) | T | • Germany • capital • Berlin | Berlin is the capital of Germany. | Berlin is the capital of Germany. |
| (3) | F | • Berlin • capital • Germany | Berlin's capital is Germany. | Berlin is the capital of Germany. |
| (4) | F | • Leinster • is Part Of • Dublin | Leinster is part of Dublin. | Leinster is part of Dublin. |
| (5) | F | • Rome • capital • Italy | Rome's capital is Italy. | Rome is the capital of Italy. |
| (6) | T | • Italy • capital • Rome | Italy's capital is Rome. | Rome is the capital of Italy. |
| (7) | T | • Texas • capital • Austin • Andrews County Airport • location • Texas | Austin is the capital of Texas where Andrews County Airport is located. | Austin is the capital of Texas where Andrews County Airport is located. |
| (8) | F | • Austin • capital • Texas • Andrews County Airport • location • Texas | The capital of Austin is Texas and Andrews County Airport is located in Texas. | Andrews County Airport is located in Texas where Austin is the capital. |

Table 14: Example generations from shuffled (S), true (T), and corrupted (F) triple facts by $T5_{small}$, fine-tuned on correctly ordered triples (*order*) and randomly shuffled input (*shuf*).

| D | Model | Examples |
|---|---|---|
| AMR | Reference | I had to deal with verbal abuse from my dad for a long 8 years before I came to uni and honestly, the only reason why I'm here is because it was the only way out. |
| | T5 | I had to deal with 8 years of verbal abuse from my dad before coming to university and honestly the only reason I'm here is because it's the only way out. |
| | BART | I had to deal with my dad's verbal abuse for 8 years long before coming to uni and honestly the only reason I'm here is because it's the only way out. |
| | Mager et al. (2020) | i've had to deal with verbal abuse from my dad for 8 years (before i came to uni i was honestly the only reason i was here) and it's only because of the way it is. |
| WebNLG | Reference | Aaron Turner is an electric guitar player who has played with the black metal band Twilight and with Old Man Gloom. Death metal is a musical fusion of black metal. |
| | T5 | Aaron Turner plays the electric guitar and is associated with the band Twilight. He is also a member of the Old Man Gloom band. Black metal and death metal are both genres of music. |
| | BART | The black metal genre is a variation of the death metal genre. It is also where the band, Twilight, and the alternative rock band, Old Man Gloom, are from. One of the members of the band is Aaron Turner, who plays the electric guitar. |
| | Harkous et al. (2020) | Aaron Turner, a.k.a. Black Metal, is a member of the Twilight (band) and Old Man Gloom bands. He also plays electric guitar and has a strong connection with the Death Metal genre. |

Table 15: Examples of text generated by the different models. D refers to the dataset.

| Model | Examples |
|---|---|
| Reference | in this paper, a new array signal processing technique by using particle swarm optimization is proposed to identify multipath channel parameters. the proposed array signal processing technique provides estimates to the channel parameters by finding a global minimum of an optimization problem. since the optimization problem is formulated in the cross-ambiguity function (caf) domain of the transmitted signal and the received array outputs, the proposed array signal processing technique is called as particle swarm optimization. the performance of the particle swarm optimization is compared with the space alternating generalized expectation maximization technique and with another recently proposed pso based technique for various snr values. simulation results indicate the superior performance of the pso based technique over mentioned techniques for all snr values. |
| T5 | this paper, we propose an array signal processing technique based on particle swarm optimization to estimate the multipath channel parameters from the received and the transmitted signal in a cross-ambiguity domain. the proposed array signal processing technique is formulated as an optimization problem and particle swarm optimization is compared with the generalized expectation maximization technique. simulation results show that the proposed particle swarm optimization outperforms the pso based technique for a wide range of snr values. in addition, the proposed particle swarm optimization is computationally more efficient than the gem based technique for small snr values. |
| BART | in this paper, a new array signal processing technique based on particle swarm optimization is proposed. the proposed array signal processing technique is used to estimate the multipath channel parameters from the transmitted signal. the proposed array signal processing technique is formulated as an optimization problem in the cross-ambiguity domain. the particle swarm optimization is compared with the generalized expectation maximization technique. simulation results show that the proposed particle swarm optimization outperforms the pso based technique for all snr values. furthermore, the proposed particle swarm optimization is able to estimate the channel parameters more accurately than the generalized expectation maximization technique. |
| Ribeiro et al. (2020) | in this paper, a novel array signal processing technique based on particle swarm optimization is proposed to estimate the multipath channel parameters from the transmitted signal. the proposed array signal processing technique uses particle swarm optimization to estimate the multipath channel parameters. the proposed array signal processing technique is formulated as an optimization problem. simulation results show that the proposed array signal processing technique outperforms the conventional generalized expectation maximization technique and the pso based technique is robust to the snr values. |

Table 16: Examples of text generated by the different models trained on the AGENDA dataset.

# Chapter 8

# Improving Visual Reasoning by Exploiting The Knowledge in Texts

The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)

# Improving Scene Graph Classification by Exploiting Knowledge from Texts

**Sahand Sharifzadeh**[1*], **Sina Moayed Baharlou**[1*†], **Martin Schmitt**[2], **Hinrich Schütze**[2], **Volker Tresp** [1,3]

[1] Department of Informatics, LMU Munich, Germany
[2] Center for Information and Language Processing (CIS), LMU Munich, Germany
[3] Siemens AG, Munich, Germany
sahand.sharifzadeh@gmail.com, sina.baharlou@gmail.com

**Abstract**

Training scene graph classification models requires a large amount of annotated image data. Meanwhile, scene graphs represent relational knowledge that can be modeled with symbolic data from texts or knowledge graphs. While image annotation demands extensive labor, collecting textual descriptions of natural scenes requires less effort. In this work, we investigate whether textual scene descriptions can substitute for annotated image data. To this end, we employ a scene graph classification framework that is trained not only from annotated images but also from symbolic data. In our architecture, the symbolic entities are first mapped to their correspondent image-grounded representations and then fed into the relational reasoning pipeline. Even though a structured form of knowledge, such as the form in knowledge graphs, is not always available, we can generate it from unstructured texts using a transformer-based language model. We show that by fine-tuning the classification pipeline with the extracted knowledge from texts, we can achieve ∼8x more accurate results in scene graph classification, ∼3x in object classification, and ∼1.5x in predicate classification, compared to the supervised baselines with only 1% of the annotated images.

## Introduction

Relational reasoning is one of the essential components of intelligence; humans explore their environment by grasping the entire context of a scene rather than studying each item in isolation from the others. Furthermore, we expand our understanding of the world by educating ourselves about novel facts through reading or listening. For example, we might have never seen a "cow wearing a dress" but might have read about Hindu traditions of decorating cows. While we already have a robust visual system that can extract basic visual features such as edges and curves from a scene, the description of a "cow wearing a dress" refines our visual understanding of relations on an object level and enables us to recognize a dressed cow when seeing it.

Relational reasoning is gaining growing popularity in the Computer Vision community and especially in the form of scene graph (SG) classification. The goal of SG classification is to classify objects and their relations in an image. One of the challenges in SG classification is collecting annotated image data. Most approaches in this domain rely on thousands of manually labeled and curated images. In this paper, we investigate whether the SG classification models can be fine-tuned from textual scene descriptions (similar to the "dressed cow" example above).

We consider a classification pipeline with two major parts: a feature extraction *backbone*, and a *relational reasoning* component (Figure 1). The backbone is typically a convolutional neural network (CNN) that detects objects and extracts an image-based representation for each. On the other hand, the relational reasoning component can be a variant of a recurrent neural network [Xu et al. 2017, Zellers et al. 2018] or graph convolutional networks [Yang et al. 2018, Sharifzadeh, Baharlou, and Tresp 2021]. This component operates on an object level by taking the latent representations of all the objects in the image and propagating them in the graph.

Note that, unlike the feature extraction backbone that requires images as input, the relational reasoning component operates on graphs with the nodes representing objects and the edges representing relations. The distinction between the input to the backbone (images) and the relational reasoning component (graphs) is often overlooked. Instead, the scene graph classification pipeline is treated as a network that takes only images as inputs. However, one can also train or fine-tune the relational reasoning component directly by injecting it with relational knowledge. For example, Knowledge Graphs (KGs) contain curated facts that indicate the relations between a `head` object and a `tail` object in the form of `(head, predicate, tail)` e.g., `(Person, Rides, Horse)`. The facts in KGs are represented by symbols whereas the inputs to the relational reasoning component are image-based embeddings. In this work, we map the triples to image-grounded embeddings as if they are coming from an image. We then use these embeddings to fine-tune the relational reasoning component through a denoising graph autoencoder scheme.

Note that the factual knowledge is not always available in a well-structured form, specially in domains where the knowledge is not stored in the machine-accessible form of KGs. In fact, most of the collective human knowledge is only

[*] These authors contributed equally.
[†] S. M. Baharlou contributed to this project while he was a visiting researcher at the Ludwig Maximilian University of Munich.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

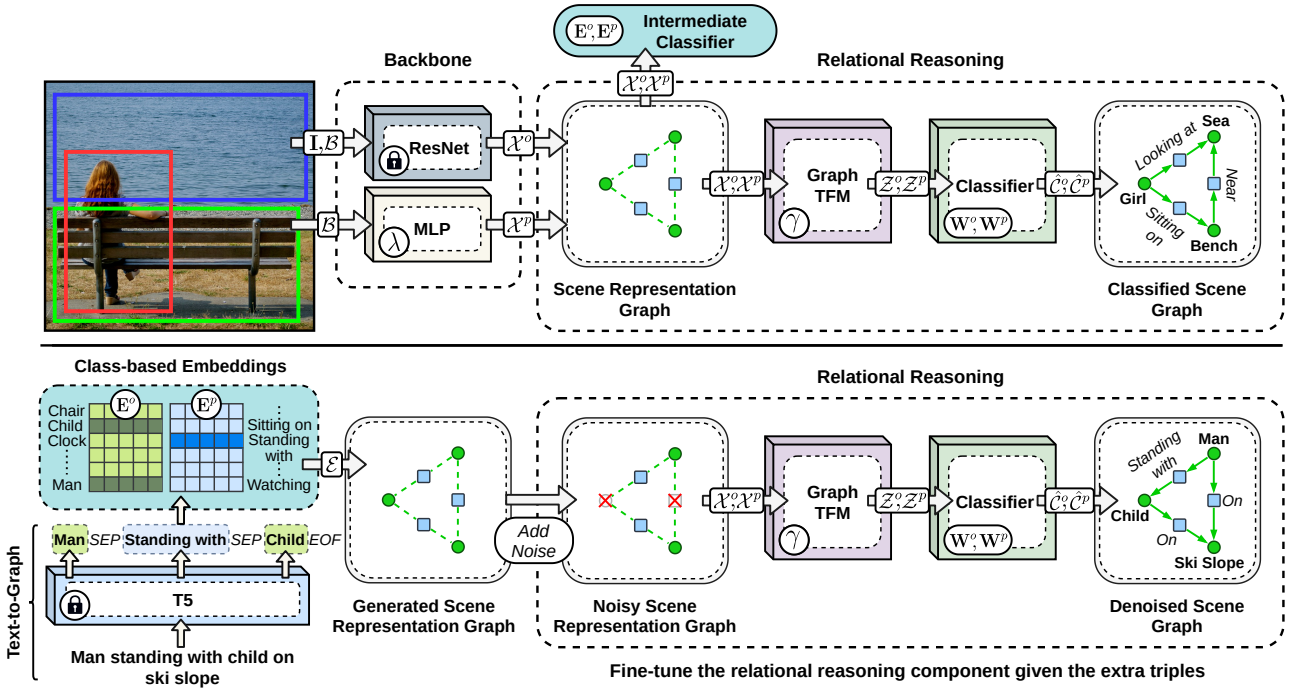



Figure 1: Top: we initially train a scene graph classification pipeline from images and their corresponding SGs. Bottom: we then use a text-to-graph module to extract structured knowledge from unstructured texts. The extracted graph is embedded by image-grounded vectors, masked, and then fed to the relational reasoning module to predict the missing relations and thus, encourage the network to learn the new relations from texts. The *lock* sign indicates pre-trained and frozen parts of the network.

available in the unstructured form of texts and documents. Exploiting this form of knowledge, in addition to structured knowledge, can be significantly beneficial. To this end, we employ a transformer-based model to generate structured graphs from textual input and utilize them to improve the relational reasoning module.

In summary, we propose *Texema*, a scene graph classification pipeline that can be trained from the large corpora of unstructured knowledge. We evaluate our approach on the Visual Genome dataset. In particular, we show that we can fine-tune the reasoning component using textual scene descriptions instead of thousands of images. As a result, when using as little as ∼500 images (1% of the VG training data), we can achieve ∼3x more accurate results in object classification, ∼8x in scene graph classification and ∼1.5x in predicate classification compared to the supervised baselines. Additionally, in our ablation studies, we evaluate the performance of using different rule-based, LSTM-based, and transformed-based text-to-graph models.

## Related Works

**Scene Graph Classification:** There is an extensive body of work on visual reasoning in general that includes different forms of reasoning [Wu, Lenz, and Saxena 2014, Deng et al. 2014, Hu et al. 2016, 2017, Santoro et al. 2017, Zellers et al. 2019]. Here, we mainly review the works that are focused on scene graph classification. Visual Relation Detection (VRD) [Lu et al. 2016] and the Visual Genome [Krishna et al. 2017] are the main datasets for this task. While the original papers on VRD and VG provide the baselines for scene graph classification by treating objects independently, several follow-up works contextualize the entities before classification. Iterative Message Passing (IMP) [Xu et al. 2017], Neural Motifs [Zellers et al. 2018] (NM), Graph R-CNN [Yang et al. 2018], and Schemata [Sharifzadeh, Baharlou, and Tresp 2021] proposed to propagate the image context using basic RNNs, LSTMs, graph convolutions, and graph transformers respectively. On the other hand, authors of VTransE [Zhang et al. 2017] proposed to capture relations by applying TransE [Bordes et al. 2013], a knowledge graph embedding model, on the visual embeddings, Tang et al. [2019] exploited dynamic tree structures to place the object in an image into a visual context. Chen et al. [2019a] proposed a multi-agent policy gradient method that frames objects into cooperative agents and then directly maximizes a graph-level metric as the reward. In tangent to those works, Sharifzadeh et al. [2021] proposed to enrich the input domain in scene graph classification by employing the predicted pseudo depth maps of VG images that were released as an extension called *VG-Depth*.

**Commonsense in Scene Understanding:** Several recent works have proposed to employ external or internal sources of knowledge to improve visual understanding [Wang, Ye,

| Input | man standing with child on ski slope |
|---|---|
| Reference Graph (RG) | **(child, on, ski slope) (man, on, ski slope) (man, standing with, child)** |
| $R_{\text{text}\rightarrow\text{graph}}$ | *(man, standing, child)* |
| SSGP | *(standing, with, child) (standing, on, slope)* |
| CopyNet (1%) | **(man, standing with, child)** |
| T5 (1%) | **(man, standing with, child)** |
| CopyNet (10%) | **(man, standing with, child)** *(child, on, slope)* |
| T5 (10%) | **(man, standing with, child) (child, on, ski slope)** |

Table 1: An example of extracted triples from a given text input in VG, using different methods. Bold: correct ($\in$ RG). Italic: incorrect ($\notin$ RG). The results are computed using the respective official code bases of the related works.

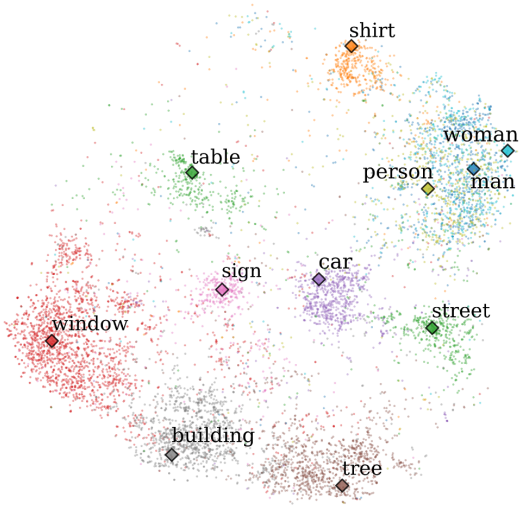



Figure 2: The t-SNE representation of the $\mathbf{e}_i$s (diamonds) and image-based representations $\mathcal{X}$s (dots) where each color represents the ground-truth class of the dot.

and Gupta 2018, Jiang et al. 2018, Singh et al. 2018, Kato, Li, and Gupta 2018]. In the scene graph classification domain, some of the works have proposed to correct the SG prediction errors by merely comparing them to the co-occurrence statistics of internal triples as a form of commonsense knowledge [Chen et al. 2019c,b, Zellers et al. 2018]. Earlier, Baier, Ma, and Tresp [2017, 2018] proposed the first scene graph classification model that employed prior knowledge in the form of Knowledge Graph Embeddings (KGEs) that generalize beyond the given co-occurrence statistics. Zareian, Karaman, and Chang [2020], Zareian et al. [2020] followed this approach by extending it to models that are based on graph convolutional networks. More recently, Sharifzadeh, Baharlou, and Tresp [2021] proposed Schemata as a generalized form of a KGE model that is learned directly from the images rather than triples. In general, scene graph classification methods are closely related to the KGE models. Therefore, we refer the interested readers to [Nickel et al. 2016, Ali et al. 2020a,b] for a review and large-scale study on the KG models, and to [Tresp, Sharifzadeh, and Konopatzki 2019, Tresp et al. 2020] for an extensive investigation of the connection between perception, KG models, and cognition.

Nevertheless, to the best of our knowledge, the described methods have employed curated knowledge in the form of triples, and none of them have directly exploited the textual knowledge. In this direction, the closest work to ours is by Yu et al. [2017], proposing to distill the external language knowledge using a teacher-student model. However, this work does not include a relational reasoning component and only refines the final predictions. Also, as shown in the experiments, our knowledge extraction module performs two times better than the SG Parser used in that work.

**Knowledge Extraction from Text:** Knowledge extraction from text has been studied for a long time [Chinchor 1991]. Previous work ranges from pattern-based approaches [Hearst 1992] to supervised neural approaches with specialized architectures [Gupta et al. 2019, Yaghoobzadeh, Adel, and Schütze 2017]. Recently, Schmitt et al. [2020] successfully applied a general sequence-to-sequence architecture to graph↔text conversion. With the recent rise of transfer learning in NLP, an increasing number of approaches are based on large language models, pre-trained in a self-supervised manner on massive amounts of texts [Devlin et al. 2019]. Inspired from previous work that explores transfer learning for graph-to-text conversion [Ribeiro et al. 2020], we base our text-to-graph model on a pre-trained T5 model [Raffel et al. 2019].

## Methods

In this section, we first describe the backbone and relational reasoning components. We then describe our approach for fine-tuning the network from texts. We have three possible forms of data: Images (**IM**), Scene Graphs (**SG**) and Textual Scene Descriptions (**TXT**). We consider having two sets of data: one is the *parallel* set, which is the set of IM with their corresponding SG and TXT, and another is the *text* set which is a set of additional TXT that come without any images or scene graphs. These two sets have no elements in common.

We initially train our backbone and relational reasoning component from IM and SG, and our text-to-graph model from the TXT and SG in the parallel set. We then show that we can fine-tune the pipeline using the text set and without using any additional images.

### Backbone (Algorithm 1.1)

The feature-extraction backbone is a convolutional neural network (ResNet-50) that has been pre-trained in a self-supervised manner [Grill et al. 2020] from unlabeled images of ImageNet [Deng et al. 2009] and Visual Genome [Krishna et al. 2017]. Given an image $\mathbf{I}$ with several objects in bounding boxes $\mathcal{B} = \{\mathbf{b}_i\}_{i=1}^n$, $\mathbf{b}_i = [b_i^x, b_i^y, b_i^w, b_i^h]$, we apply the ResNet-50 to extract pooled object features $\mathcal{X}^o = \{\mathbf{x}_i^o\}_{i=1}^n$, $\mathbf{x}_i^o \in \mathbb{R}^d$. Here $[b_i^x, b_i^y]$ are the coordinates of $\mathbf{b}_i$ and $[b_i^w, b_i^h]$ are its width and height, and $d$

Algorithm 1: Classify objects/predicates from images

1. **Extract image features (Backbone):**

   **Input:** Images and object bounding boxes ($\mathbf{I}, \mathcal{B} : \{\mathbf{b_i}\}_{i=1}^{n}$).

   **Output:** Object embeddings $\mathcal{X}^o : \{\mathbf{x}_i^o\}_{i=1}^{n}$ and predicate embeddings $\mathcal{X}^p : \{\mathbf{x}_i^p\}_{i=1}^{m}$.

   **Trainable params:** $\lambda$.

   $\mathcal{X}^o = ResNet50(\mathbf{I}, \mathcal{B})$

   $\mathcal{X}^p = \{MLP_\lambda(t(\mathbf{b}_i, \mathbf{b}_j)) \mid \forall \mathbf{b}_i, \mathbf{b}_j \in \mathcal{B}\}$

2. **Contextualize and Classify (Relational Reasoning):**

   **Input:** Object embeddings $\mathcal{X}^o : \{\mathbf{x}_i^o\}_{i=1}^{n}$, Predicate embeddings $\mathcal{X}^p : \{\mathbf{x}_i^p\}_{i=1}^{m}$ and ground truth classes $\mathcal{C}^o$ and $\mathcal{C}^p$.

   **Output:** Predicted object class distribution $\hat{\mathcal{C}}^o : \{\hat{\mathbf{c}}_i^o\}_{i=1}^{n}$ and predicted predicate class distribution $\hat{\mathcal{C}}^p : \{\hat{\mathbf{c}}_i^p\}_{i=1}^{m}$.

   **Trainable params:** $\gamma$, $\mathbf{W}^o, \mathbf{W}^p$.

   $\mathcal{Z}^o, \mathcal{Z}^p = GraphTransformer_\gamma(\mathcal{X}^o, \mathcal{X}^p)$

   $\hat{\mathcal{C}}^o = \{\text{softmax}(\mathbf{W}^o \cdot \mathbf{z}^o) \mid \forall \mathbf{z}^o \in \mathcal{Z}^o\}$

   $\hat{\mathcal{C}}^p = \{\text{softmax}(\mathbf{W}^p \cdot \mathbf{z}^p) \mid \forall \mathbf{z}^p \in \mathcal{Z}^p\}$

3. **Apply Loss (Cross-Entropy):**

   $l_o = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{\|\mathbf{c}_i^o\|} \mathbf{c}_{i,j}^o . log(\hat{\mathbf{c}}_{i,j}^o)$

   $l_p = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{\|\mathbf{c}_i^p\|} \mathbf{c}_{i,j}^p . log(\hat{\mathbf{c}}_{i,j}^p)$

are the vector dimensions. Following [Zellers et al. 2018], we define $\mathcal{X}^p = \{\mathbf{x}_i^p\}_{i=1}^{m}$, $\mathbf{x}_i^p \in \mathbb{R}^d$ as the relational features between each pair of objects. Each $\mathbf{x}_i^p$ is initialized by applying a two layered fully connected network on the relational position vector $\mathbf{t}$ between a head $i$ and a tail $j$ where $\mathbf{t} = [t_x, t_y, t_w, t_h]$, $t_x = (b_i^x - b_j^x)/b_i^w{}_j, t_y = (b_i^y - b_j^y)/b_j^h, t_w = \log(b_i^w/b_j^w), t_h = \log(b_i^h/b_j^h)$. The implementation and pre-training details of the layers are provided in the Evaluation. $\mathcal{X}^o$ and $\mathcal{X}^p$ form a structured presentation of the objects and predicates in the image also known as **Scene Representation Graph** (SRG) [Sharifzadeh, Baharlou, and Tresp 2021]. SRG is a fully connected graph with each node representing either an object or a predicate, where each object node is a direct neighbor to predicate nodes and each predicate node is a direct neighbor with its head and tail object nodes.

## Relational Reasoning (Algorithm 1.2)

The relational reasoning component updates the initial SRG representations through Graph Transformer layers [Koncel-Kedziorski et al. 2019]. The outputs of these layers are $\mathcal{Z}^o = \{\mathbf{z}_i^o\}_{i=1}^{n}$, $\mathbf{z}_i^o \in \mathbb{R}^d$ and $\mathcal{Z}^p = \{\mathbf{z}_i^p\}_{i=1}^{m}$, $\mathbf{z}_i^p \in \mathbb{R}^d$ with equal dimensions as $\mathcal{X}$s. From here on, we drop the superscripts of $o$ and $p$ for brevity. We apply a linear classification layer $\mathbf{W}$ to classify the contextualized representations $\mathcal{Z}$ such that $\hat{\mathbf{c}} = \text{softmax}(\mathbf{W} \cdot \mathbf{z}_i)$, with cross-entropy as the loss function.

## Fine-tuning from Texts (Algorithm 2)

Let us assume that we have already trained the backbone and relational reasoning components from IM and SG in the *parallel* set. Now, we want to fine-tune the weights in the

Algorithm 2: Fine-tune the relational reasoning component from textual triples using a denoising auto-encoder paradigm

1. **Learn image-grounded representations E for each symbol through classification (without Graph Transformer):**

   **Input:** Object embeddings $\mathcal{X}^o : \{\mathbf{x}_i^o\}_{i=1}^{n}$, predicate embeddings $\mathcal{X}^p : \{\mathbf{x}_i^p\}_{i=1}^{m}$ and their corresponding ground truth classes $\mathcal{C}^o$ and $\mathcal{C}^p$.

   **Output:** Predicted object class distribution $\hat{\mathcal{C}}^o : \{\hat{\mathbf{c}}_i^o\}_{i=1}^{n}$ and predicted predicate class distribution $\hat{\mathcal{C}}^p : \{\hat{\mathbf{c}}_i^p\}_{i=1}^{m}$.

   **Trainable params:** $\mathbf{E}^o, \mathbf{E}^p$.

   $\hat{\mathcal{C}}^o = \{\text{softmax}(\mathbf{E}^o \cdot \mathbf{x}^o) \mid \forall \mathbf{x}^o \in \mathcal{X}^o\}$

   $\hat{\mathcal{C}}^p = \{\text{softmax}(\mathbf{E}^p \cdot \mathbf{x}^p) \mid \forall \mathbf{x}^p \in \mathcal{X}^p\}$

2. **Apply Loss (Cross Entropy):**

   $l_o = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{\|\mathbf{c}_i^o\|} \mathbf{c}_{i,j}^o . log(\hat{\mathbf{c}}_{i,j}^o)$

   $l_p = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{\|\mathbf{c}_i^p\|} \mathbf{c}_{i,j}^p . log(\hat{\mathbf{c}}_{i,j}^p)$

3. **Fine-tune the relational reasoning component given the extra triples (Denoising Graph Autoencoder):**

   **Input:** Symbolic triples $\mathcal{S} : \{(h_i, p_i, t_i)\}_{i=1}^{k}$ and canonical object/predicate representations $\mathbf{E}^o/\mathbf{E}^p$.

   **Output:** Embedded representations $\mathcal{E} : \{(\mathbf{e}_i^h, \mathbf{e}_i^p, \mathbf{e}_i^t)\}_{i=1}^{k}$.

   **Trainable params:** $\gamma, \mathbf{W}^o, \mathbf{W}^p$.

   - Build $\mathcal{E} : \{(\mathbf{e}_i^h, \mathbf{e}_i^p, \mathbf{e}_i^t)\}_{i=1}^{k}$ where for each $(h_i, p_i, t_i)$:
     $\mathbf{e}_i^h = onehot(h_i) \cdot \mathbf{E}^o$
     $\mathbf{e}_i^p = onehot(p_i) \cdot \mathbf{E}^p$
     $\mathbf{e}_i^t = onehot(t_i) \cdot \mathbf{E}^o$
   - Randomly set 20% of the nodes and edges in $\mathcal{E}$ to zero.
   - Set $\mathcal{X}^o = \mathcal{E}^h \cup \mathcal{E}^t$ and $\mathcal{X}^p = \mathcal{E}^p$ and run Algorithm 1.2 to fine-tune $\gamma, \mathbf{W}^o, \mathbf{W}^p$, with $\mathcal{E}^h$, $\mathcal{E}^t$ and $\mathcal{E}^p$ as the set of all heads, tails, and predicates in $\mathcal{E}$.

relational reasoning component given the additional *text* set. The relational reasoning component takes graphs as input, therefore, we first need to convert TXT to SG:

**Text-to-graph:** This model is trained from the SG and TXT in the parallel set, and then used to generate SG from the text set. Let us consider an unstructured text such as "man standing with child on ski slope" (Table 1 - Input). A structured form of this sentence is a graph with unique nodes and edges for each entity or predicate. For example, the reference graph for this sentence contains the triples `(child, on, ski slope)`, `(man, standing with, child)` and `(man, on, ski slope)` (Table 1 - RG).

In order to learn this mapping, we employ a transformer-based [Vaswani et al. 2017] sequence-to-sequence $T5_{small}$ model [Raffel et al. 2019] and adapt it for the task of extracting graphs from texts. T5 consists of an encoder with several layers of self-attention (like BERT, Devlin et al. 2019) and a decoder with autoregressive self-attention (like GPT-3, Brown et al. 2020). In order to use a T5 model with graphs, we need to represent the graphs as a sequence. To this end, we serialize the graphs by writing out their facts separated

| Method | Precision | | Recall | | F1 | |
|---|---|---|---|---|---|---|
| | 1% | 10% | 1% | 10% | 1% | 10% |
| $R_{text \rightarrow graph}$ | $1.92 \pm 0.00$ | $1.86 \pm 0.01$ | $1.87 \pm 0.00$ | $1.81 \pm 0.01$ | $1.89 \pm 0.00$ | $1.84 \pm 0.01$ |
| SSGP | $14.86 \pm 0.01$ | $14.52 \pm 0.02$ | $18.47 \pm 0.01$ | $18.05 \pm 0.02$ | $16.47 \pm 0.01$ | $16.09 \pm 0.02$ |
| CopyNet | $29.20 \pm 0.13$ | $30.77 \pm 0.49$ | $27.19 \pm 0.28$ | $29.79 \pm 0.29$ | $28.16 \pm 0.21$ | $30.27 \pm 0.34$ |
| **T5** | $\mathbf{33.37 \pm 0.11}$ | $\mathbf{33.81 \pm 0.08}$ | $\mathbf{31.06 \pm 0.18}$ | $\mathbf{32.45 \pm 0.33}$ | $\mathbf{32.17 \pm 0.13}$ | $\mathbf{33.12 \pm 0.16}$ |

Table 2: The mean and standard deviation of Precision, Recall, and F1 scores of the predicted facts from the texts on four random splits. The results are computed using the respective official code bases of the related works and evaluated on VG.

by end-of-fact symbols (EOF), and separate the elements of each fact with SEP symbols [Schmitt et al. 2020], e.g. *"child SEP on SEP ski slope EOF"* (Fig. 1). To adapt the multi-task setting from T5's pretraining, we use the task prefix "make graph: " to mark our text-to-graph task. Table 1 shows an example text and the extracted graphs using T5 and other previous methods (see Evaluation for details).

**Map to embeddings:** Note that the predicted graphs are a sequence of symbols for heads, predicates, and tails where each symbol represents a class $c \in \mathcal{C}$. However, the inputs to the relational reasoning component are image-based vectors $\mathcal{X}$. Thus, before feeding the symbols to the relational reasoning component, we need to map them to a corresponding embedding from the space of $\mathcal{X}$ as if we are feeding it with image-based embeddings. In order to do that, we train a mapping from symbols to $\mathcal{X}$s using the IM and SG of the parallel set. This is simply done by training a linear classification layer $\mathbf{E}$ given $\mathcal{X}$s from the parallel set (Algorithm 2.1). Unlike the classification layer in Algorithm 1, here we classify $\mathcal{X}$s instead of $\mathcal{Z}$s and the goal is *not* to use the classification output but to train image-grounded, canonical representations for each class: each row $\mathbf{e}_i$ in the classification layer becomes a cluster center for $\mathcal{X}$s from class $i$ (Figure 2). Therefore, instead of the extracted symbolic $c_i$ from the text set, we can feed its canonical image-grounded representation $\mathbf{e}_i$ to the graph transformer (Algorithm 2.3).

**Denoising Graph Autoencoder:** To fine-tune the relational reasoning given this data, we treat the relational reasoning component as a denoising autoencoder where the input is an incomplete (noisy) graph that comes from the text and the output is the denoised graph. If we do not apply a denoising autoencoder paradigm, the function will collapse to an identity map. We create the noisy graph by randomly setting some of the input nodes and edges to zero during the training (Algorithm 2.3). The goal is to encourage the graph transformer to predict the missing links and therefore, learn the relational structure.

## Evaluation

We first compare the performance of different rule-based and embedding-based text-to-graphs models on our data. We then evaluate the performance of our entire pipeline in classifying objects and relations in images. In particular, we show that the extracted knowledge from the texts can largely substitute annotated images as well as ground-truth graphs.

**Dataset:** We use the sanitized version [Xu et al. 2017] of Visual Genome (VG) dataset [Krishna et al. 2017] including images and their annotations, i.e., bounding boxes, scene graphs, and scene descriptions. Our goal is to design an experiment that evaluates whether we can substitute annotated images with textual scene descriptions. Therefore, instead of using external textual datasets with unbounded information, we use Visual Genome itself by dividing it into different splits of *parallel* (with IM, SG and TXT) and *text* data (with only TXT). To this end, we assume only a random proportion (1% or 10%) of training images are annotated (parallel set containing IM with corresponding SG and TXT). We consider the remaining data (99% or 90%) as our text set and discard their IM and SG. We aim to see whether employing TXT from the text set, can substitute the discarded IM and SG from this set. We use four different random splits [Sharifzadeh, Baharlou, and Tresp 2021] to avoid a sampling bias. For more detail on the datasets refer to the supplementary materials.

Note that the scene graphs and the scene descriptions from the VG are collected separately and by crowd-sourcing. Therefore, even though the graphs and the scene descriptions refer to the same image region, they are disjoint and contain complementary knowledge.

### Graphs from Texts

The goal of this experiment is to study the effectiveness of the text-to-graph model. We fine-tune the pre-trained T5 model on parallel TXT and SG, and apply it on the text set to predict their corresponding SG. We also implement the following rule-based and embedding-based baselines to compare their performance using our splits: (1) $R_{text \rightarrow graph}$ is a simple rule-based system introduced by Schmitt et al. [2020] for general knowledge graph generation from text. (2) The Stanford Scene Graph Parser (SSGP) [Schuster et al. 2015] is another rule-based approach that is more adapted to the scene graph domain. Even though this approach was not specifically designed to match the scene graphs from the Visual Genome dataset, it was still engineered to cover typical idiosyncrasies of textual image descriptions and corresponding scene graphs. (3) CopyNet [Gu et al. 2016] is an LSTM sequence-to-sequence model with a dedicated copy mechanism, which allows copying text elements directly into the graph output sequence. It was used for unsupervised text-to-graph generation by Schmitt et al. [2020]. However, we train it on the supervised data of our parallel sets. We use a vocabulary of around 70k tokens extracted from the VG-graph-text

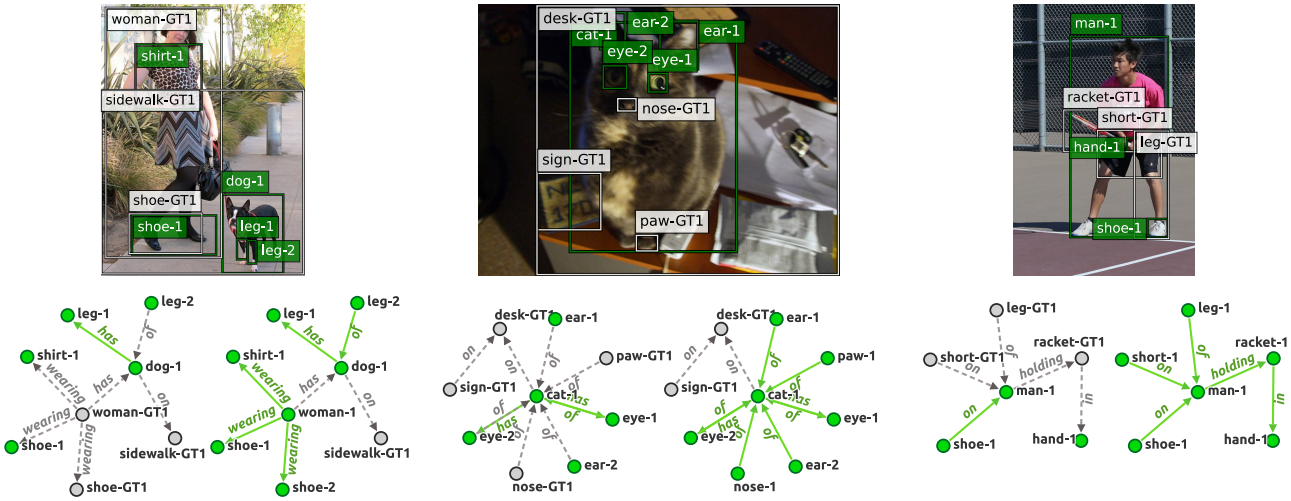



Figure 3: Qualitative examples of improved classification results (Recall@100) before and after (from left to right) fine-tuning the model using the knowledge in texts. Green and gray colors indicate true positives and false negatives concluded by the model.

| | Method | R@50 | | R@100 | |
|---|---|---|---|---|---|
| | | 1% | 10% | 1% | 10% |
| SGCls | $R_{text \to graph}$ | $10.90 \pm 0.12$ | $24.96 \pm 0.15$ | $11.80 \pm 0.11$ | $26.09 \pm 0.15$ |
| | SSGP | $14.35 \pm 0.15$ | $26.11 \pm 0.19$ | $15.14 \pm 0.17$ | $27.12 \pm 0.22$ |
| | CopyNet | $14.46 \pm 0.31$ | $26.05 \pm 0.29$ | $15.19 \pm 0.24$ | $27.08 \pm 0.26$ |
| | **TXM - T5** | $\mathbf{14.53 \pm 0.34}$ | $\mathbf{26.16 \pm 0.32}$ | $\mathbf{15.28 \pm 0.38}$ | $\mathbf{27.22 \pm 0.28}$ |
| | GT | $14.72 \pm 0.38$ | $26.33 \pm 0.45$ | $15.36 \pm 0.38$ | $27.37 \pm 0.47$ |
| PredCls | $R_{text \to graph}$ | $23.34 \pm 0.10$ | $49.99 \pm 0.12$ | $26.83 \pm 0.15$ | $54.40 \pm 0.12$ |
| | SSGP | $54.65 \pm 0.14$ | $55.65 \pm 0.15$ | $59.33 \pm 0.18$ | $59.67 \pm 0.20$ |
| | CopyNet | $56.24 \pm 0.31$ | $59.27 \pm 0.28$ | $60.35 \pm 0.20$ | $63.28 \pm 0.25$ |
| | **TXM - T5** | $\mathbf{58.64 \pm 0.34}$ | $\mathbf{59.31 \pm 0.30}$ | $\mathbf{63.07 \pm 0.37}$ | $\mathbf{63.32 \pm 0.24}$ |
| | GT | $62.02 \pm 0.10$ | $61.71 \pm 0.19$ | $65.68 \pm 0.12$ | $65.42 \pm 0.19$ |

Table 3: SGCls and PredCls results using different text-to-graph modules. We have substituted the missing 99% and 90% of annotated images with the textual knowledge extracted from their scene descriptions.

benchmark and, otherwise, also adopt the hyperparameters from [Schmitt et al. 2020]. Table 1 shows sample predictions from these models. Table 2 compares precision, recall, and F1 measures. and T5 outperforms others by a large margin.

## Graphs from Images

The goal of this experiment is to evaluate scene graph classification after fine-tuning the pipeline using textual knowledge. We evaluate our models for object classification, predicate classification (PredCls - predicting predicate labels given a ground truth set of object boxes and object labels) and scene graph classification (SGCls - predicting object and predicate labels, given the set of object boxes) on the test sets. Our ablation study concerns the following configurations:

- **SPB**: In this setting, both the backbone and the relational reasoning component are trained by *supervised learning* on the IM and SGs (1% or 10%) from the parallel set.
- **SCH**: Here, the backbone is trained by *self-supervised learning* on all VG images (without labels), and the relational reasoning component is trained on the IM and SGs (1% or 10%) from the parallel set.
- **TXM**: Here, the backbone is trained by *self-supervised learning* on all VG images (without labels), and the relational reasoning component is trained on the IM and SGs (1% or 10%) from the parallel set and fine-tuned from the SGs predicted from the text set (99% or 90%) using the text-to-graph module.
- **GT**: Here, the backbone is trained by *self-supervised learning* on all VG images (without labels), and the relational reasoning component is trained on the IM and SGs (1% or 10%) from the parallel set, and fine-tuned from the *ground truth graphs* (99% or 90%), instead of the text-to-graph predictions.

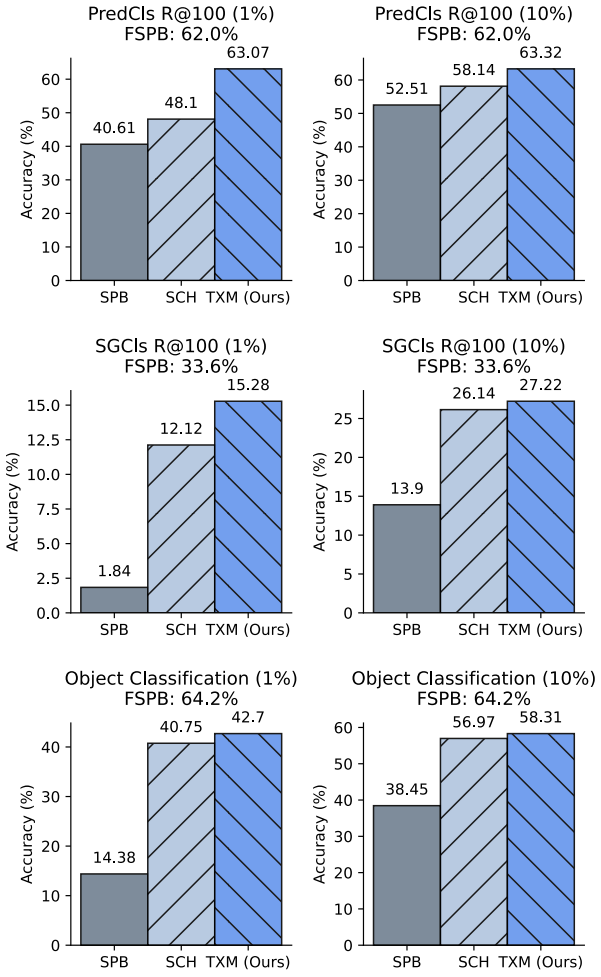



Figure 4: Fine-tuning with the textual knowledge (TXM) significantly improves the results in all settings of PredCls (top), SGCls (middle), and object classification (bottom).

- **FSPB**: Here, both the backbone and the relational reasoning component are trained by *supervised learning* on $100\%$ of the VG annotated images. Meaning that we have redefined the parallel set to include $100\%$ of the VG training data and we do not need to substitute the images with the text set anymore. The goal of this setting is to compute the maximum accuracy that our model achieves, when we have all the annotated images with ground truth SGs, instead of using their textual scene descriptions. The results of this settings are written above each table so that the other bars maintain a meaningful scale.

Figure 4 presents the results of the ablation study. We use the Recall@K (**R@K**) as metric. which computes the mean prediction accuracy in each image given the top $K$ predictions. For more results (mR@K metric and unconstrained setups) refer to the supplementary materials. As shown, fine-tuning with textual scene descriptions (TXM) improves the classification results under all settings, substituting a large proportion of the omitted images. Furthermore, the results even outperform FSPB under PredCls (recall that the scene descriptions are sometimes complementary to image annotations and contain additional information).

| Method | SGCls | | PredCls | |
|---|---|---|---|---|
| | R@50 | R@100 | R@50 | R@100 |
| VRD [Lu et al. 2016] | 11.8 | 14.1 | 27.9 | 35.0 |
| IMP+ [Xu et al. 2017] | 34.6 | 35.4 | 59.3 | 61.3 |
| SMN [Zellers et al. 2018] | 35.8 | 36.5 | 65.2 | 67.1 |
| KERN [Chen et al. 2019c] | 36.7 | 37.4 | 65.8 | 67.6 |
| VCTree [Tang et al. 2019] | 38.1 | 38.8 | 66.4 | 68.1 |
| CMAT [Chen et al. 2019a] | 39.0 | 39.8 | 66.4 | 68.1 |
| SIG [Wang et al. 2020] | 36.6 | 37.3 | 66.3 | 68.1 |
| GB-Net [Zareian et al. 2020] | 38.0 | 38.8 | 66.6 | 68.2 |
| **TXM** | **39.0** | **39.9** | **66.7** | **68.3** |

Table 4: Comparing the general performance of the architecture to some other methods under the VG test set.

Table 3 presents additional results also using different text-to-graph baselines. We can see that fine-tuning with the predicted graphs using T5, is as effective as fine-tuning with the crowd-sourced ground truth graphs (GT), and in some settings even better (object classification with $1\%$). Notice that compared to the self-supervised baseline, we gained up to $\sim$5% *relative improvement* in object classification, more than $\sim$26% in scene graph classification, and $\sim$31% in predicate prediction accuracy. As expected, the choice of text-to-graph module has a larger effect on the PredCls compared to the SGCls and ObjCls, due to the fact that SGCls and ObjCls rely heavily on the image-based features, whereas PredCls has a strong dependency to relational knowledge. In supplementary materials we also provide additional results on the improvements per object class after fine-tuning the model with the textual knowledge (From SCH to TXM) and show that most improvements occur in under-represented classes. Figure 3 provides some qualitative examples of the predicted scene graphs before and after fine-tuning with the texts. Finally, to provide an intuition on our general performance, Table 4 present the results of our architecture using a VGG-16 [Simonyan and Zisserman 2014] backbone trained with 100% of the annotations, instead of the self-supervised BYOL.

## Conclusion

In this work, we proposed the first relational image-based classification pipeline that can be fine-tuned directly from the large corpora of unstructured knowledge available in texts. We generated structured graphs from textual input using different rule-based or embedding-based approaches. We then fine-tuned the relational reasoning component of our classification pipeline by employing the canonical representations of each entity in the generated graphs. We showed that we gain a significant improvement in all settings after employing the inferred knowledge within the classification pipeline. In most cases, the accuracy was similar to when using the ground truth graphs that are manually annotated by crowd-sourcing.

## Acknowledgments

We would like to thank Masoud Jalili Sabet for the fruitful discussions, and the anonymous reviewers for their helpful feedback on the manuscript. This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A.

## References

Ali, M.; Berrendorf, M.; Hoyt, C. T.; Vermue, L.; Galkin, M.; Sharifzadeh, S.; Fischer, A.; Tresp, V.; and Lehmann, J. 2020a. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *arXiv preprint arXiv:2006.13365*.

Ali, M.; Berrendorf, M.; Hoyt, C. T.; Vermue, L.; Sharifzadeh, S.; Tresp, V.; and Lehmann, J. 2020b. Pykeen 1.0: A python library for training and evaluating knowledge graph emebddings. *arXiv preprint arXiv:2007.14175*.

Baier, S.; Ma, Y.; and Tresp, V. 2017. Improving visual relationship detection using semantic modeling of scene descriptions. In *International Semantic Web Conference*, 53–68. Springer.

Baier, S.; Ma, Y.; and Tresp, V. 2018. Improving information extraction from images with learned semantic models. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 5214–5218. AAAI Press.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. *Computing Research Repository*, arXiv:2005.14165.

Chen, L.; Zhang, H.; Xiao, J.; He, X.; Pu, S.; and Chang, S.-F. 2019a. Counterfactual critic multi-agent training for scene graph generation. In *Proceedings of the IEEE International Conference on Computer Vision*, 4613–4623.

Chen, T.; Xu, M.; Hui, X.; Wu, H.; and Lin, L. 2019b. Learning semantic-specific graph representation for multi-label image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 522–531.

Chen, T.; Yu, W.; Chen, R.; and Lin, L. 2019c. Knowledge-embedded routing network for scene graph generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6163–6171.

Chinchor, N. 1991. MUC-3 Linguistic Phenomena Test Experiment. In *Third Message Uunderstanding Conference (MUC-3): Proceedings of a Conference Held in San Diego, California, May 21-23, 1991*.

Deng, J.; Ding, N.; Jia, Y.; Frome, A.; Murphy, K.; Bengio, S.; Li, Y.; Neven, H.; and Adam, H. 2014. Large-scale object classification using label relation graphs. In *European conference on computer vision*, 48–64. Springer.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. A.; Guo, Z. D.; Azar, M. G.; et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1631–1640. Berlin, Germany: Association for Computational Linguistics.

Gupta, P.; Rajaram, S.; Schütze, H.; and Runkler, T. A. 2019. Neural Relation Extraction within and across Sentence Boundaries. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 6513–6520.

Hearst, M. A. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.

Hu, H.; Deng, Z.; Zhou, G.-T.; Sha, F.; and Mori, G. 2017. Labelbank: Revisiting global perspectives for semantic segmentation. *arXiv preprint arXiv:1703.09891*.

Hu, H.; Zhou, G.-T.; Deng, Z.; Liao, Z.; and Mori, G. 2016. Learning structured inference neural networks with label relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2960–2968.

Jiang, C.; Xu, H.; Liang, X.; and Lin, L. 2018. Hybrid knowledge routed modules for large-scale object detection. *arXiv preprint arXiv:1810.12681*.

Kato, K.; Li, Y.; and Gupta, A. 2018. Compositional learning for human object interaction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 234–251.

Koncel-Kedziorski, R.; Bekal, D.; Luan, Y.; Lapata, M.; and Hajishirzi, H. 2019. Text generation from knowledge graphs with graph transformers. *arXiv preprint arXiv:1904.02342*.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.-J.; Shamma, D. A.; et al.

2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1): 32–73.

Lu, C.; Krishna, R.; Bernstein, M.; and Fei-Fei, L. 2016. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 852–869. Springer.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1): 11–33.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Ribeiro, L. F. R.; Schmitt, M.; Schütze, H.; and Gurevych, I. 2020. Investigating Pretrained Language Models for Graph-to-Text Generation. *Computing Research Repository*, arXiv:2007.08426.

Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, 4967–4976.

Schmitt, M.; Sharifzadeh, S.; Tresp, V.; and Schütze, H. 2020. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7117–7130.

Schuster, S.; Krishna, R.; Chang, A.; Fei-Fei, L.; and Manning, C. D. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, 70–80.

Sharifzadeh, S.; Baharlou, S. M.; Berrendorf, M.; Koner, R.; and Tresp, V. 2021. Improving Visual Relation Detection using Depth Maps. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 3597–3604.

Sharifzadeh, S.; Baharlou, S. M.; and Tresp, V. 2021. Classification by Attention: Scene Graph Classification with Prior Knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 5025–5033.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, K. K.; Divvala, S.; Farhadi, A.; and Lee, Y. J. 2018. Dock: Detecting objects by transferring common-sense knowledge. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 492–508.

Tang, K.; Zhang, H.; Wu, B.; Luo, W.; and Liu, W. 2019. Learning to compose dynamic tree structures for visual contexts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6619–6628.

Tresp, V.; Sharifzadeh, S.; and Konopatzki, D. 2019. A Model for Perception and Memory. *Conference on Cognitive Computational Neuroscience*.

Tresp, V.; Sharifzadeh, S.; Konopatzki, D.; and Ma, Y. 2020. The Tensor Brain: Semantic Decoding for Perception and Memory. *arXiv preprint arXiv:2001.11027*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, W.; Wang, R.; Shan, S.; and Chen, X. 2020. Sketching image gist: Human-mimetic hierarchical scene graph generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, 222–239. Springer.

Wang, X.; Ye, Y.; and Gupta, A. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6857–6866.

Wu, C.; Lenz, I.; and Saxena, A. 2014. Hierarchical Semantic Labeling for Task-Relevant RGB-D Perception. In *Robotics: Science and systems*.

Xu, D.; Zhu, Y.; Choy, C. B.; and Fei-Fei, L. 2017. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5410–5419.

Yaghoobzadeh, Y.; Adel, H.; and Schütze, H. 2017. Noise Mitigation for Neural Entity Typing and Relation Extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 1183–1194. Valencia, Spain: Association for Computational Linguistics.

Yang, J.; Lu, J.; Lee, S.; Batra, D.; and Parikh, D. 2018. Graph r-cnn for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 670–685.

Yu, R.; Li, A.; Morariu, V. I.; and Davis, L. S. 2017. Visual relationship detection with internal and external linguistic knowledge distillation. In *IEEE International Conference on Computer Vision (ICCV)*.

Zareian, A.; Karaman, S.; and Chang, S.-F. 2020. Bridging knowledge graphs to generate scene graphs. In *European Conference on Computer Vision*, 606–623. Springer.

Zareian, A.; You, H.; Wang, Z.; and Chang, S.-F. 2020. Learning Visual Commonsense for Robust Scene Graph Generation. *arXiv preprint arXiv:2006.09623*.

Zellers, R.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6720–6731.

Zellers, R.; Yatskar, M.; Thomson, S.; and Choi, Y. 2018. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5831–5840.

Zhang, H.; Kyaw, Z.; Chang, S.; and Chua, T. 2017. Visual Translation Embedding Network for Visual Relation Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 3107–3115. IEEE Computer Society. ISBN 978-1-5386-0457-1.

# Bibliography

Grigoris Antoniou and Frank van Harmelen. 2004. Web ontology language: Owl. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 67–92. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. 2007. *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd edition. Cambridge University Press.

Franz Baader, Ian Horrocks, and Ulrike Sattler. 2005. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan, editors, *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, pages 228–248. Springer Berlin Heidelberg, Berlin, Heidelberg.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Presented at the 3rd International Conference on Learning Representations (ICLR)*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings*

*of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jonathan Berant. 2012. *Global Learning of Textual Entailment Graphs*. Ph.D. thesis, The Blavatnik School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University.

Jonathan Berant, Ido Dagan, Meni Adler, and Jacob Goldberger. 2012. Efficient tree-based approximation for entailment graph learning. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 117–125, Jeju Island, Korea. Association for Computational Linguistics.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2010. Global learning of focused entailment graphs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1220–1229, Uppsala, Sweden. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA.

Grady Booch, James Rumbaugh, and Ivar Jacobson. 2005. *The Unified Modeling Language User Guide*. Addison-Wesley object technology series. Addison-Wesley.

Alex Borgida. 1996. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1):353–367.

Jos de Bruijn and Stijn Heymans. 2010. Logical foundations of rdf(s) with datatypes. *J. Artif. Int. Res.*, 38(1):535–568.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Augustin-Louis Cauchy. 1847. Méthode générale pour la résolution de systèmes d'équations simultanées. In *Compte rendu des séances de l'académie des sciences*, pages 536–538.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on*

*Empirical Methods in Natural Language Processing*, pages 33–40, Barcelona, Spain. Association for Computational Linguistics.

Namyoun Choi, Il-Yeol Song, and Hyoil Han. 2006. A survey on ontology mapping. *ACM SIGMOD Record*, 35(3):34–41.

Edgar F. Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Bernard Comrie. 1989. *Language Universals and Linguistic Typology: Syntax and Morphology*, 2nd edition. University of Chicago Press.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Claudio Delli Bovi, Luis Espinosa-Anke, and Roberto Navigli. 2015. Knowledge base unification via sense embeddings and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 726–736, Lisbon, Portugal. Association for Computational Linguistics.

Caglar Demir, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2021. A shallow neural model for relation prediction. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 179–182.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Philipp Dufter. 2021. *Distributed representations for multilingual language processing*. Ph.D. thesis, Ludwig-Maximilians-Universität München.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. 2020. Increasing learning efficiency of self-attention networks through direct position interactions, learnable temperature, and convoluted attention. In *Proceedings of the 28th International*

*Conference on Computational Linguistics*, pages 3630–3636, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. 2021. Position information in transformers: An overview. *Computing Research Repository*, arXiv:2102.11090.

Philipp Dufter, Mengjie Zhao, Martin Schmitt, Alexander Fraser, and Hinrich Schütze. 2018. Embedding learning through multilingual concept induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1520–1530, Melbourne, Australia. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

John R. Firth. 1957. *A synopsis of linguistic theory 1930–1955*, pages 1–32. The Philological Society, Oxford. Reprinted in: F. R. Palmer (ed.) (1968). Selected Papers of J. R. Firth 1952–1959, pages 168–205. London: Longman.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114, Ann Arbor, Michigan. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the*

*54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. CycleGT: Unsupervised graph-to-text and text-to-graph generation via cycle training. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 77–88, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Graeme S. Halford, William H. Wilson, and Steven Phillips. 2010. Relational knowledge: the foundation of higher cognition. *Trends in Cognitive Sciences*, 14(11):497–505.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Pat Hayes and Chris Welty. 2006. Defining n-ary relations on the semantic web. *Online:* `https://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/`. Accessed on 16th August 2021.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge graphs. *Computing Research Repository*, arXiv:2003.02320.

Ian Horrocks, Oliver Kutz, and Ulrike Sattler. 2006. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press.

Mohammad Javad Hosseini, Nathanael Chambers, Siva Reddy, Xavier R. Holt, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2018. Learning typed entailment graphs with global soft constraints. *Transactions of the Association for Computational Linguistics*, 6:703–717.

Mohammad Javad Hosseini, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2019. Duality of link prediction and entailment graph induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4736–4746, Florence, Italy. Association for Computational Linguistics.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Richard Hudson. 1990. *English Word Grammar*. Oxford: Blackwell.

Tomasz Imielinski and Witold Lipski. 1982. A systematic approach to relational database theory. In *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data*, SIGMOD '82, page 8–14, New York, NY, USA. Association for Computing Machinery.

Dan Jurafsky and James H. Martin. 2020. *Speech and Language Processing*, 3rd edition. Online. December 30, 2020 draft.

Manoj Prabhakar Kannan Ravi, Kuldeep Singh, Isaiah Onando Mulang', Saeedeh Shekarpour, Johannes Hoffart, and Jens Lehmann. 2021. CHOLAN: A modular approach for neural entity linking on Wikipedia and Wikidata. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 504–514, Online. Association for Computational Linguistics.

Pei Ke, Haozhe Ji, Yu Ran, Xin Cui, Liwei Wang, Linfeng Song, Xiaoyan Zhu, and Minlie Huang. 2021. JointGT: Graph-text joint representation learning for text generation from knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2526–2538, Online. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine

translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR.

Omer Levy and Ido Dagan. 2016. Annotating relation inference in context via question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 249–255, Berlin, Germany. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, volume 27, pages 2177–2185. Curran Associates, Inc.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

John Lewis and William Loftus. 2009. *Java Software Solutions: Foundations of Program Design*. Pearson/Addison-Wesley.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of Word2Vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *Computing Research Repository*, arXiv:2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Computing Research Repository*, arXiv:1907.11692.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. 2017. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020a. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online. Association for Computational Linguistics.

Nitika Mathur, Johnny Wei, Markus Freitag, Qingsong Ma, and Ondřej Bojar. 2020b. Results of the WMT20 metrics shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 688–725, Online. Association for Computational Linguistics.

Yehudit Meged, Avi Caciularu, Vered Shwartz, and Ido Dagan. 2020. Paraphrasing vs coreferring: Two sides of the same coin. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4897–4907, Online. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computing Research Repository*, arXiv:1301.3781.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Abhishek Nadgeri, Anson Bastos, Kuldeep Singh, Isaiah Onando Mulang', Johannes Hoffart, Saeedeh Shekarpour, and Vijay Saraswat. 2021. KGPool: Dynamic knowledge graph context selection for relation extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 535–548, Online. Association for Computational Linguistics.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145, Jeju Island, Korea. Association for Computational Linguistics.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8528–8535.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3866–3878, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI research blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Computing Research Repository*, arXiv:1910.10683.

Leonardo F.R̃. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for amr-to-text generation. *Computing Research Repository*, arXiv:2103.09120.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–363, Melbourne, Australia. Association for Computational Linguistics.

Frank Rosenblatt. 1962. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Washington: Spartan Books.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, College of Engineering and Informatics, School of Engineering and Informatics, National University of Ireland, Galway.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098, Cambridge, MA. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Vered Shwartz, Gabriel Stanovsky, and Ido Dagan. 2017. Acquiring predicate paraphrases from news tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 155–160, Vancouver, Canada. Association for Computational Linguistics.

Aaron Smith, Christian Hardmeier, and Joerg Tiedemann. 2016. Climbing mont BLEU: The strange world of reachable high-BLEU translations. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 269–281.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Lucien Tesnière. 1959. *Eléments de la syntaxe structurale*. Paris: Klincksieck.

Lloyd N. Trefethen and David Bau, III. 1997. *Numerical Linear Algebra*, volume 50 of *Other Titles in Applied Mathematics*. Society for Industrial and Applied Mathematics.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1317–1327, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, page 5998–6008. Curran Associates, Inc.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55(1):953–994.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, Berlin Chen, Benjamin Van Durme, Edouard Grave, Ellie Pavlick, and

Samuel R. Bowman. 2019. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4465–4476, Florence, Italy. Association for Computational Linguistics.

Zhichun Wang, Jinjian Yang, and Xiaoju Ye. 2020. Knowledge graph alignment with entity-pair embedding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1672–1680, Online. Association for Computational Linguistics.

Warren Weaver. 1955. Translation. In *Machine Translation of Languages*, pages 15–23. Cambridge, Massachusetts: MIT Press.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1015–1021, Geneva, Switzerland. COLING.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Tien-Hsuan Wu, Zhiyong Wu, Ben Kao, and Pengcheng Yin. 2018. Towards practical open knowledge base canonicalization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 883–892, New York, NY, USA. Association for Computing Machinery.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations (ICLR)*.

Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online. Association for Computational Linguistics.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

# Acronyms

**ALM** autoregressive language model. 28, 30, 31

**DIH** distributional inclusion hypothesis. 33, 34

**DL** deep learning. 18, 23, 24

**KB** knowledge base. 16, 20, 23, 37, 38

**KBC** knowledge base completion. 17, 32

**KG** knowledge graph. 4, 16–23, 27, 35, 37, 38

**MLM** masked language model. 28, 30, 31

**MT** machine translation. 35

**NLI** natural language inference. 17

**NLP** natural language processing. 18, 24, 30, 31

**NN** neural network. 18, 23–25, 27, 29, 30

**OIE** open information extraction. 38

**PLM** pretrained language model. 4, 18, 30, 31, 34, 39, 40

**RIC** relation inference in context. 4, 18, 31–35, 38, 39

# Acknowledgments

This work would not have been possible without the help and support of many people. First, I would like to thank my supervisor Hinrich Schütze. You believed in my potential and thanks to you I have had many opportunities to widen my horizon and deepen my knowledge. I am particularly grateful that you encouraged me to work autonomously and find the research directions I feel most passionate about. Thank you also particularly for all your advice about scientific writing and the English language in general.

Many thanks to Axel-Cyrille Ngonga Ngomo and Benjamin Roth for taking the time to review this thesis and to the whole examination board for creating a positive atmosphere and sparking a great discussion during my defense.

I would like to thank all my co-workers at CIS for many fruitful discussions and collaborations, but also for enjoyable excursions and convivial dinners. Specifically, I want to mention Wenpeng Yin, Philipp Dufter, and Nora Kassner, with whom I shared an office. It was great working with and beside you! A special thank you to Simon Steinheber, Konrad Schreiber, Marina Speranskaya, Marina Sedinkina, Max Mozes, Sahand Sharifzadeh, Volker Tresp, Leonardo Ribeiro, and Iryna Gurevych for working together on our projects. Thank you, Heike Adel, Dario Stojanovski, Alex Fraser, Helmut Schmid, Annemarie Friedrich, and Alexandra Chronopoulou, for taking the time to listen to me and share your advice. Thank you, Axel Wisiorek, for organizing the lectures and exercises "Syntax natürlicher Sprachen" together. Teaching was a highlight of my time at CIS and working together with you certainly contributed to that fact. Thank you, Peggy Hobmaier, for your help with all administrative challenges and thank you, Thomas Schäfer, for being the amazing system administrator you are!

It was a great experience organizing the ESSLLI student session. I would like to thank all my co-organizers, especially Jen Sikos and Chantal van Son, for a great time working and laughing together. I am grateful I could learn so much from you.

I thank the Studienstiftung des deutschen Volkes for their long-lasting support already during my Bachelor's and Master's studies and now during my PhD. I very much enjoyed the vivid discussions during their events.

I also gratefully acknowledge support from the BMBF as part of the project

MLWin. I enjoyed the interdisciplinary exchange between the MLWin team members and I notably thank Mehdi Ali, Afshin Sadeghi, and Vladimir Golkov for insightful discussions.

Last but not least, I would not be where I am today without the support of my family. Thank you for enabling me to focus on my work when the deadlines drew closer and thank you for being there when I needed you most.