Deep Learning Methods for Knowledge Base Population

Heike Adel

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

vorgelegt von
Heike Adel

München, den 25. Januar 2018

Erstgutachter: Prof. Dr. Hinrich Schütze
Zweitgutachter: Reader Ph.D. Sebastian Riedel
Tag der mündlichen Prüfung: 26.06.2018

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 25.09.2018

Heike Adel

Acknowledgments

My PhD time was accompanied and supported by a lot of amazing people. I would like to thank my advisor Hinrich Schütze for our many fruitful discussions and brainstorming meetings. I learned a lot from you and deeply admire your ability to quickly immerse yourself in new problems and possible solutions. You always supported us, your PhD students, and gave us the opportunity to focus as much as possible on our research. Moreover, I am grateful to you for getting the opportunity of doing two internships during my PhD time.

In these internships, I learned a lot about user-oriented research and met a lot of interesting people. In my first internship at FXPAL, I worked with Francine Chen and Yan-Ying Chen and would like to thank them for all their passion and time they put into my project and our collaboration, even after my internship had ended. My second internship at Google was full of interesting brainstorming sessions with a variety of researchers. I would like to especially thank Anton Bryl and Aliaksei Severyn for supporting my research and setting me in contact with so many amazing co-workers. In the context of my Google Doctoral Fellowship, I further met Enrique Alfonseca who became my fellowship mentor and talked with me regularly. I always enjoyed these chats and am very grateful for them, especially since I know about his busy calendar.

At CIS, I was happy to meet a very heterogenous group of colleagues and friends. I would like to especially mention some of them I spent most of my time with, both in and outside of our academic life: Katharina Kann, Yadollah Yaghoobzadeh, Sebastian Ebert, Sascha Rothe, Wenpeng Yin, Irina Sergienya, Lucia Krisnawati, Desislava Zhekova and Matthias Huck. I greatly enjoyed all of our meetings as well as our daily lunch and coffee breaks. We could always count on each other when one of us had a problem, either in research or in our daily life, and I am happy that many of us are still in contact although we are now distributed all over the world. Further, I am grateful to Marco Cornolti, who visited CIS for a few months and helped me a lot with integrating their entity linking system into my slot filling system.

Finally, my PhD time would have been much harder and less enjoyable without the support and patience of my parents Edeltraud and Jürgen and my husband Thang. *Es ist eine unbeschreibliche Erleichterung, dass ich immer auf euch zählen kann.*

Contents

List of Abbreviations

CNN convolutional neural network.

CRF conditional random field.

EC entity classification.

EL entity linking.

GRU gated recurrent unit.

KB knowledge base.

KBP knowledge base population.

LDC Linguistic Data Consortium.

LSTM long short-term memory.

NER named entity recognition.

NLP natural language processing.

NYT New York Times.

OOV out-of-vocabulary.

PR precision-recall.

RE relation extraction.

RNN recurrent neural network.

SVM support vector machine.

TAC Text Analysis Conference.

List of Figures

List of Tables

Abstract

Knowledge bases store structured information about entities or concepts of the world and can be used in various applications, such as information retrieval or question answering. A major drawback of existing knowledge bases is their incompleteness. In this thesis, we explore deep learning methods for automatically populating them from text, addressing the following tasks: slot filling, uncertainty detection and type-aware relation extraction.

Slot filling aims at extracting information about entities from a large text corpus. The Text Analysis Conference yearly provides new evaluation data in the context of an international shared task. We develop a *modular system* to address this challenge. It was one of the top-ranked systems in the shared task evaluations in 2015. For its slot filler classification module, we propose `contextCNN`, a *convolutional neural network based on context splitting*. It improves the performance of the slot filling system by 5.0% micro and 2.9% macro F1. To train our binary and multiclass classification models, we create a dataset using distant supervision and reduce the number of noisy labels with a *self-training strategy*. For model optimization and evaluation, we automatically extract a *labeled benchmark* for slot filler classification from the manual shared task assessments from 2012–2014. We show that results on this benchmark are correlated with slot filling pipeline results with a Pearson's correlation coefficient of 0.89 (0.82) on data from 2013 (2014). The combination of patterns, support vector machines and `contextCNN` achieves the best results on the benchmark with a micro (macro) F1 of 51% (53%) on test. Finally, we analyze the results of the slot filling pipeline and the impact of its components.

For knowledge base population, it is essential to assess the factuality of the statements extracted from text. From the sentence "Obama was rumored to be born in Kenya", a system should not conclude that Kenya is the place of birth of Obama. Therefore, we address uncertainty detection in the second part of this thesis. We investigate attention-based models and make a first attempt to systematize the attention design space. Moreover, we propose novel attention variants: *External attention*, which incorporates an external knowledge source, *k-max average attention*, which only considers the vectors with the k maximum attention weights, and *sequence-preserving attention*, which allows to maintain order information. Our convolutional neural network with external k-max average attention sets the new state of the art on a Wikipedia benchmark dataset with an F1 score of 68%. To the best of our knowledge, we are the first to integrate an *uncertainty detection component* into a slot filling pipeline. It improves precision by 1.4% and micro F1 by 0.4%.

In the last part of the thesis, we investigate type-aware relation extraction with neural

networks. We compare different models for joint entity and relation classification: pipeline models, jointly trained models and globally normalized models based on structured prediction. First, we show that using *entity class prediction scores* instead of binary decisions helps relation classification. Second, *joint training* clearly outperforms pipeline models on a large-scale distantly supervised dataset with fine-grained entity classes. It improves the area under the precision-recall curve from 0.53 to 0.66. Third, we propose a model with a *structured prediction output layer*, which globally normalizes the score of a triple consisting of the classes of two entities and the relation between them. It improves relation extraction results by 4.4% F1 on a manually labeled benchmark dataset. Our analysis shows that the model learns correct correlations between entity and relation classes. Finally, we are the first to use neural networks for joint entity and relation classification in a slot filling pipeline. The jointly trained model achieves the best micro F1 score with a score of 22% while the neural structured prediction model performs best in terms of macro F1 with a score of 25%.

Zusammenfassung

Wissensdatenbanken speichern strukturierte Informationen über Entitäten und Konzepte der Welt und können in verschiedenen Anwendungen eingesetzt werden, wie zum Beispiel zur Informationssuche oder zum automatischen Beantworten von Fragen. Eine große Schwachstelle existierender Wissensdatenbanken ist ihre Unvollständigkeit. In dieser Arbeit erforschen wir "deep learning" Methoden, um sie automatisch mit Hilfe von Textdaten zu erweitern. Konkret befassen wir uns mit den folgenden Aufgaben: "Slot Filling", Erkennung von Ungewissheit und Relationsextraktion mit Hilfe von Entitätentypen.

Slot Filling zielt auf die Extraktion von Informationen über Entitäten aus einem großen Textkorpus ab. Die Text Analysis Conference stellt jährlich neue Evaluationsdaten im Rahmen eines internationalen Wettbewerbs zur Verfügung. Wir entwickeln ein *modulares System* für diese Aufgabe. Es war eines der führenden Systeme in den Wettbewerbsbewertungen von 2015. Für sein Relationsklassifikationsmodul schlagen wir `contextCNN` vor, ein auf Faltung und Kontextteilung basierendes neuronales Netz. Es verbessert die Leistung des Slot-Filling-Systems um 5.0% Micro- und 2.9% Macro-F1. Um die binären und mehrklassigen Relationsklassifikationsmodelle zu trainieren, erstellen wir einen Datensatz mit Hilfe von "distant supervision" (entfernter Überwachung) und reduzieren die Anzahl verrauschter Annotationen mit einer *Selbsttrainingsstrategie*. Zur Modelloptimierung und -evaluierung extrahieren wir automatisch *annotierte Benchmarkdaten* zur Slot-Filling-Relationsklassifikation aus den manuellen Bewertungen des Wettbewerbs der Jahre 2012–2014. Wir zeigen, dass Ergebnisse auf diesem Benchmark mit den Ergebnissen der Slot-Filling-Pipeline auf den Daten von 2013 (2014) mit einem Pearson-Korrelationskoeffizienten von 0.89 (0.82) korreliert sind. Eine Kombination aus Mustererkennung, Support-Vektor-Maschinen und `contextCNN` erreicht auf dem Benchmark die besten Ergebnisse mit einem Micro-F1-Wert von 51% (beziehungsweise einem Macro-F1-Wert von 53%) auf den Testdaten. Schließlich analysieren wir die Ergebnisse der Slot-Filling-Pipeline und den Einfluss ihrer Komponenten.

Für die Erweiterung von Wissensdatenbanken ist die Bewertung der Faktizität einer Aussage entscheidend. Ein System sollte aus einem Satz wie "Es hieß, dass Obama in Kenia geboren sei" nicht schließen, dass der Geburtsort von Obama Kenia ist. Daher befassen wir uns im zweiten Teil dieser Arbeit mit der Erkennung von Ungewissheit. Wir untersuchen Methoden basierend auf "Attention" (Aufmerksamkeit) und unternehmen einen ersten Versuch, den Designraum von Attention zu systematisieren. Außerdem schlagen wir neue Varianten von Attention vor: *externe Attention*, die eine externe Wissensquelle einbezieht,

k-max Durchschnittsattention, die nur die Vektoren mit den k höchsten Attentiongewichten betrachtet, und *sequenzerhaltende Attention*, die es ermöglicht, Informationen über die Eingabereihenfolge zu bewahren. Unser faltendes neuronales Netzwerk mit externer k-max Durchschnittsattention setzt mit einem F1-Wert von 68% den Stand der Technik auf einem Wikipedia Benchmarkdatensatz neu. Nach unserem besten Wissen sind wir die ersten, die eine *Ungewissheitserkennungskomponente* in die Slot-Filling-Pipeline integrieren. Sie erhöht die Präzision des Systems um 1.4% und den Micro-F1-Wert um 0.4%.

Im letzten Teil der Arbeit untersuchen wir Relationsextraktion mit Hilfe von Entitätentypen. Wir vergleichen unterschiedliche neuronale Modelle zur gemeinsamen Entitäten- und Relationsklassifikation: Pipeline-Modelle, gemeinsam trainierte Modelle und global normalisierte Modelle basierend auf strukturierten Vorhersagen. Zum Einen zeigen wir, dass es der Relationsextraktion hilft, wenn *Wahrscheinlichkeitswerte* der Entitätenklassen verwendet werden anstatt binärer Entscheidungen. Zum Anderen übertreffen *gemeinsam trainierte* Modelle nacheinander trainierte Pipeline-Modelle deutlich auf einem großen, entfernt überwachten Datensatz mit feinkörnigen Entitätenklassen. Sie verbessern die Fläche unter der Precision-Recall Kurve (die Präzision und Trefferquote des Systems zueinander ins Verhältnis setzt) von 0.53 auf 0.66. Zum Dritten schlagen wir ein Modell vor, das die Bewertung eines Tripels bestehend aus den Klassen zweier Entitäten und der Relation zwischen ihnen global normalisiert, indem es eine *strukturierte Ausgabeschicht* verwendet. Es verbessert die F1-Ergebnisse der Relationsextraktion um 4.4% auf einem manuell annotierten Benchmarkdatensatz. Unsere Analyse zeigt, dass das Modell korrekte Korrelationen zwischen Entitäten- und Relationsklassen lernt. Schließlich sind wir die ersten, die gemeinsam trainierte und global normalisierte Modelle für Slot Filling verwenden. Das gemeinsam trainierte Modell erreicht den besten Micro-F1-Wert mit 22%, während das Modell basierend auf strukturierten Vorhersagen mit 25% am besten bezüglich des Macro-F1-Wertes abschneidet.

Prepublications

Contents and Co-Authorship

Some chapters of this thesis contain material that has been published at peer-reviewed international conferences or as shared task system descriptions in the context of this thesis. Table 1 illustrates which parts of the chapters are based on which publications.

Slot Filling

Chapter 3 covers content of the following publications:

- Heike Adel, Benjamin Roth, and Hinrich Schütze. Comparing convolutional neural networks to traditional models for slot filling. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 828–838. Association for Computational Linguistics, 2016

- Heike Adel and Hinrich Schütze. CIS at TAC cold start 2015: Neural networks and coreference resolution for slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015

- Heike Adel and Hinrich Schütze. TAC KBP 2014 slot filling shared task: Baseline system for investigating coreference. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014

Moreover, some of the coreference resolution investigations have been made available on a pre-print server:

- Heike Adel and Hinrich Schütze. Impact of coreference resolution on slot filling. *Computing Research Repository (arXiv.org), arXiv:1710.09753*, 2017a

The research described in this chapter was carried out entirely by myself. The other authors of the publications acted as advisors.

I regularly discussed this work with my advisors, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial drafts of the articles and did most of the subsequent corrections. My advisors assisted me in improving the drafts.

Uncertainty Detection

Chapter 4 covers content of the following publication:

- Heike Adel and Hinrich Schütze. Exploring different dimensions of attention for uncertainty detection. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 22–34. Association for Computational Linguistics, 2017c

The research described in this chapter was carried out entirely by myself. The other author of the publication acted as advisor.

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Type-Aware Relation Extraction

Chapter 5 covers content of the following publications:

- Heike Adel and Hinrich Schütze. Global normalization of convolutional neural networks for joint entity and relation classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 1724–1730. Association for Computational Linguistics, 2017b

- Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. Noise mitigation for neural entity typing and relation extraction. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 1183–1194. Association for Computational Linguistics, 2017

The research described in this chapter was carried out entirely by myself.

The second author of the first article acted as advisor. I regularly discussed the work of the first article with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the first article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

The second of the two articles is the result of a collaboration. Yadollah Yaghoobzadeh and I contributed equally to the published article. Yadollah Yaghoobzadeh contributed those parts that are concerned with entity typing while I contributed those parts that are concerned with relation extraction. The third author of the second article acted as advisor. I regularly discussed the relation extraction part with my co-authors. Apart from these explicitly declared exceptions, I conceived of the original research contributions of the relation extraction part and performed the corresponding implementation and evaluation. I wrote the initial draft of the relation extraction part and did most of the subsequent corrections. My co-authors assisted me in improving the draft. The parts of the article which Yadollah Yaghoobzadeh contributed are not included in this thesis. Thus, the research described in this chapter was carried out entirely by myself.

	EMN17	JOINT17	EACL17	COR17	NAACL16	TAC15	TAC14
Section 3.1.3					x		
Section 3.2						x	x
Section 3.3					x	x	
Section 3.4.2					x	x	
Section 3.4.3					x	x	
Section 3.4.5				x		x	
Section 3.5.2					x		
Section 3.5.4						x	
Section 3.6.3					x		
Section 3.6.4						x	
Section 4.1 – Section 4.5			x				
Section 5.2.1 – Section 5.2.4		x					
Section 5.3.1 – Section 5.3.5	x						

Table 1: Illustration which subsections are based on which of the following publications: EMN17: (Adel and Schütze, 2017b), JOINT17: (Yaghoobzadeh et al., 2017), EACL17: (Adel and Schütze, 2017c), COR2017: (Adel and Schütze, 2017a), NAACL16: (Adel et al., 2016), TAC15: (Adel and Schütze, 2015), TAC14: (Adel and Schütze, 2014).

Chapter 1

Introduction

1.1 Motivation

With the growing amount of unstructured text data, especially on the Internet, the need for structured representation of knowledge arises. Structured knowledge representations can be used in various applications, such as information retrieval, question answering or automatic assistant systems. Given a database containing facts about entities of the world (a so-called knowledge base), answering a question like "Who founded Apple" requires only a simple lookup. Similarly, an automatic assistant system can guide a user looking for popular sights nearby by extracting points of interests and information about them from that database.

Currently, several large-scale knowledge bases exist. However, they are highly incomplete, limiting their applicability in down-stream tasks. For many entities, even fundamental facts like the place of birth of a person are missing. On the other hand, many of those missing facts are mentioned in the large amount of unstructured text available through news or on the Internet. Therefore, this thesis tackles the challenge of automatically extracting information from text for populating a knowledge base.

When automatically extracting structured information from text data, several natural language processing (NLP) challenges arise. The following example is an excerpt of the Wikipedia article about Steve Jobs:[1]

> Steven Paul "Steve" Jobs (February 24, 1955 – October 5, 2011) was an American entrepreneur, business magnate, inventor, and industrial designer. He was the chairman, chief executive officer (CEO), and co-founder of Apple Inc; ...

These two sentences contain a variety of information, which can be used to populate a knowledge base: An alternative name for Steve Jobs ("Steven Paul Jobs"), his date of birth (February 24, 1955) and date of death (October 5, 2011), his origin ("American"), several job titles ("entrepreneur", "business magnate", "inventor", "industrial designer", "chairman", "chief executive officer") as well as the name of a company which he co-founded

[1] `https://en.wikipedia.org/wiki/Steve_Jobs` (November 14, 2017).

(“Apple Inc“). However, this piece of text poses several challenges to information extraction systems, such as the recognition that “Steven Paul ‘Steve’ Jobs” and “Steve Jobs” refer to the same person, the knowledge that “American” is a nationality, the conclusion that the multi-token expression “chief executive officer” forms a title rather than only the single token “officer” as it could occur it other contexts, the recognition that “he” refers to “Steve Jobs”, the recognition that “Apple Inc” is a company, etc.

In this thesis, we focus on different sub-tasks of information extraction for knowledge base population: slot filling, uncertainty detection and type-aware relation extraction.

Slot filling aims at extracting information about named entities from text by filling predefined slots, such as the place of birth of a person, or the founders of a company. It can be compared to the task of populating Wikipedia info boxes for a given entity (Wikipedia page) using textual information. It is annually organized as a shared task by the Text Analysis Conference (TAC). Even the top-ranked systems only achieve F1 scores around 30%. This indicates the difficulty of the task. In the context of this thesis, we participated in the official slot filling evaluations in 2014 and 2015.

In the second part of the thesis, we present our work on uncertainty detection. From a sentence like “He may have died in the accident”, it is not desirable to extract a cause-of-death fact for a knowledge base since its veracity is not proven. Instead, a knowledge base population system should recognize the uncertainty and handle it accordingly. It could, for instance, ignore the extracted values or assign a special marker to them.

In the last part of the thesis, we investigate type-aware relation extraction models which can jointly predict entity and relation classes. The knowledge of the types of the relation arguments can guide relation extraction while the knowledge of the relations an entity participates in can, in turn, improve entity classification results. For example, the information that the two arguments of a relation are a person and an organization can reduce the search space of relation classes by excluding particular relations, such as place of birth, which would require a person and a location as relation arguments. Similarly, the knowledge that an entity participates in a place-of-birth relation as well as in a founder-of relation can help conclude that it is most likely a person. Thus, the mutual dependency between entity and relation classes can help a model correctly classify information from text.

1.2 Main Contributions

In this thesis, we contribute to the state of the art of knowledge base population research as described below. More details are given at the end of each chapter.

Slot Filling. We develop a state-of-the-art slot filling system which tackles a variety of natural language processing challenges. It was ranked third in the official shared task evaluations 2015. With this system, we are one of the first to successfully apply neural networks in the relation classification component of a slot filling system. In particular, we propose `contextCNN`, a convolutional neural network especially designed for relation

classification, which is based on context splitting and makes use of a flag indicating the order of the relation arguments. It improves the results of the slot filling pipeline and outperforms a state-of-the-art convolutional neural network for relation extraction on slot filler classification. It can also be applied generally to relation extraction tasks. In order to facilitate the development and comparison of slot filling systems, we automatically create a benchmark dataset for slot filler classification from the manual slot filling system assessments.

Uncertainty Detection. For uncertainty detection, we experiment with attention-based models. We make a first attempt to systematize the design space of attention and propose several new attention mechanisms: external attention, k-max average attention and sequence-preserving attention. Although we investigate them in the context of uncertainty detection, they are generally applicable in attention-based models independent of the task. External attention provides the possibility to include external information, such as a lexicon of uncertainty cues for uncertainty detection, in order to guide the model during training. K-max average attention is an extension of traditional average attention which only averages the weighted vectors with the k largest attention weights. This can be beneficial when the attention weight distribution is not sharp and standard attention would introduce noise. Sequence-preserving attention tackles another drawback from standard attention with average: By taking the average, any sequence information from the input is lost. With sequence-preserving attention, we propose a way to maintain this information, which might be relevant for many natural language processing tasks. We are the first to apply convolutional and recurrent neural networks to uncertainty detection and analyze their different behavior. Our convolutional neural network with external k-max average attention performs best and sets the new state of the art on a benchmark dataset. When using it as a component in the slot filling pipeline, it improves precision. To the best of our knowledge, we are the first to employ an uncertainty detection component for slot filling.

Type-aware Relation Extraction. We investigate different type-aware neural networks for relation extraction. On a distantly supervised dataset, we show that jointly training entity and relation classification models improves results over traditional pipeline approaches. For pipeline models, computing features from the probabilities of the entity classifier outperforms features derived from binary decisions. Moreover, we propose a novel way of modeling the joint task of entity and relation classification as a sequence of predictions. This enables the application of a structured prediction output layer. On a manually labeled dataset, we compare jointly trained convolutional neural networks with the ones based on structured prediction and show that structured prediction improves the results. Finally, we are the first to integrate joint information extraction models into the slot filling pipeline and show that they improve the final results of the system.

1.3 Structure

The remainder of this thesis is structured as follows:

Chapter 2 provides background information which is useful for the remaining chapters of this thesis. Section 2.1 reports on knowledge bases and knowledge base population in general and the slot filling task in particular. Section 2.2 gives an overview of different neural network layers and how to train them.

Chapter 3 describes our work around the slot filling task. In particular, Section 3.1 introduces the task and Section 3.2 describes the modular system we have developed for participating in the shared task. Section 3.3 focuses on the slot filler classification models. In Section 3.4, we report on the datasets used for the different experiments. Section 3.5 and Section 3.6 show our results and analysis, respectively. Finally, we give an overview of related work in Section 3.7 and describe our contributions in more detail in Section 3.8.

Chapter 4 presents our experiments in the area of uncertainty detection. After motivating and introducing the task in Section 4.1, we describe the models we have developed in Section 4.2. Section 4.3, Section 4.4 and Section 4.5 present the dataset, our experimental results and our analysis, respectively. Finally, we show in Section 4.6 how the models can be applied in a slot filling system and report on related work in Section 4.7. Section 4.8 summarizes our contributions.

In Chapter 5, we describe experiments on relation extraction with type-aware neural networks. After motivating the task in Section 5.1, we explore models which are jointly trained on fine-grained entity typing and relation classification in Section 5.2. Afterwards, we propose a model with a structured prediction output layer which globally normalizes a sequence of predictions of entity and relation classes in Section 5.3. In both sections, we present our models, the dataset, our experimental results and our analysis. In Section 5.4, we show how we integrate the models into our slot filling pipeline. Finally, we report on related work in the field of type-aware relation extraction in Section 5.5 and summarize our contributions in Section 5.6.

Chapter 6 concludes the thesis and provides an overview of future work for each topic.

Chapter 2

Background

This chapter provides an overview of background relevant to this thesis. The first section describes knowledge bases, knowledge base population and distant supervision. The second section introduces neural networks, different types of layers and training with backpropagation, minibatches and regularization.

2.1 Knowledge Bases

Knowledge bases (KBs) store structured information about (real-world) entities, such as people, places or more abstract concepts like songs or artistic movements. Popular examples for large-scale knowledge bases are Freebase (Bollacker et al., 2008), Wikidata (Pellissier Tanon et al., 2016), YAGO (Suchanek et al., 2007; Hoffart et al., 2013; Mahdisoltani et al., 2015), DBpedia (Auer et al., 2007; Mendes et al., 2012; Lehmann et al., 2015) or the Google Knowledge Graph (Singhal, 2012).

Formally, a knowledge base can be defined as a collection of triples (e_1, r, e_2), also called facts or statements, with r being the binary relation between the two entities e_1 and e_2 (Gardner, 2015). These triples can be represented as a graph with entities being nodes and relations being directed edges between them. To express n-ary relations with triples, Freebase, for instance, uses mediator instances (abstract entities) which connect the n arguments. For example, to encode that Barack Obama was US president from 2009 to 2017, Freebase uses a mediator instance, called component value type (CVT) and six different triples as depicted in Figure 2.1 (Pellissier Tanon et al., 2016). Wikidata, in contrast, is able to express this n-ary relation as a single statement.

While YAGO and DBpedia automatically extract their facts from Wikipedia, Freebase and Wikidata are based on a mostly manual, collaborative effort. In contrast to Freebase, Wikidata statements encode claims rather than true facts from different sources, which may also contradict each other (Pellissier Tanon et al., 2016). The Google Knowledge Graph has been built based on the information stored in Freebase, Wikipedia and the CIA World Factbook, and has been augmented at large scale (Singhal, 2012). Singhal (2012) reports that it contained 500M entities and 3.5G facts about and relationships between

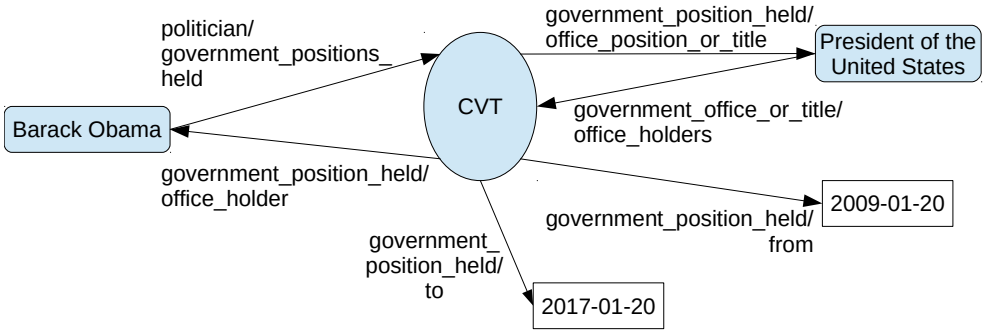


Figure 2.1: Example for component value type in Freebase (Pellissier Tanon et al., 2016).

them in 2012. However, it only provides a search API for accessing its information but no data dump. Other projects, such as NELL (The Never-Ending Language Learner) (Carlson et al., 2010) provide automatic methods with lower precision to automatically read the web and populate a knowledge base with the extracted information.

Table 2.1 provides statistics about the information stored in different knowledge bases. As noted by Pellissier Tanon et al. (2016), the numbers of entities (topics, items, instances), relation instances (facts, statements) or labels (properties) are not directly comparable since the knowledge bases have, for instance, different notability criteria for which entities they store and a different handling of inverse relations. Pellissier Tanon et al. (2016) further report that Freebase contains almost 3 billion facts out of which only 442 million facts are useful for integrating them into Wikidata. The other facts include triples about labels, identifiers, types and descriptions. On the other hand, when representing Wikidata statements as Freebase facts (with reverse facts and compound value types for n-ary relations), the number increases from 66 million statements to 110 million facts.

	Freebase	Wikidata	YAGO2	DBpedia (en)
# entities	48M	14.5M	9.8M	4.6M
# facts	2997M	66M	447.5M	152.9M
# labels	68M	82M	365.5k	61.8k

Table 2.1: Statistics of different knowledge bases. Sources: Freebase and Wikidata: (Pellissier Tanon et al., 2016), YAGO2: (Hoffart et al., 2013), DBpedia (sum of facts from different extractors for English): (DBpedia, 2015).

For the experiments in this thesis, we use Freebase for the following reasons: (i) It contains only true facts with a high precision due to the manual effort; (ii) it covers a high number of entity and relation instances; (iii) it provided downloadable data dumps making experiments stable over time and reproducible; (iv) it is one of the most widely used knowledge bases in NLP research with many datasets depending on it.

2.1.1 Knowledge Base Population

Despite the large number of entities and relations stored in knowledge bases (see Table 2.1), they are still incomplete. Min et al. (2013) report that 93.8% of persons from Freebase have no place of birth, 96.6% no places of living, 78.5% no nationality and 98.8% no parents. According to West et al. (2014), 99% of persons have no ethnicity in Freebase. Completing a knowledge base manually is expensive and slow, especially considering the large number of entities which would need to be updated for existing relations as well as for any newly introduced relation label. Therefore, research in natural language processing investigates automatic methods for creating new knowledge bases from scratch or filling missing information into an existing knowledge base. There are two main trends: Extending existing knowledge bases by reasoning over them and inferring missing links, and extracting new structured information from unstructured text data. The latter is often referred to as knowledge base population (KBP) (Glass and Gliozzo, 2018). The eponymous shared task organized by the Text Analysis Conference (TAC)[1] consists of the following tasks: entity discovery and linking, slot filling, event nugget detection and coreference, event argument extraction and linking, belief and sentiment (from entities towards entities). In this thesis, we focus on knowledge base population, in particular on the slot filling task.

Slot Filling

In the context of this thesis, we have participated in the slot filling task of the KBP track of TAC. The goal of this task is the extraction of information about an entity (person, organization or geo-political entity) from unstructured text data, such as the place of birth of a person or the founder of a company. A detailed task and system description as well as our results in the official evaluation in 2015 are provided in Chapter 3.

The participants of the slot filling task are provided with a large set of text documents from different genres as well as with queries containing the entities and slots their systems should produce outputs for. Considering a knowledge base with triples (e_1, r, e_2) as defined in Section 2.1, the systems are given e_1 (the query entity) and r (the relation provided by the query, also referred to as "slot") and are supposed to output e_2 (the slot filler) along with a proof sentence that validates that triple. Thus, core components of a slot filling system are a slot filler candidate extraction and a slot filler classification component, which need to identify a set of slot filler candidates C and classify whether the triple (e_1, r, c), $c \in C$, is supported by the text or not.

2.1.2 Distant Supervision

One challenge of slot filler classification, or relation extraction in general, is the limited amount of labeled training examples. This makes supervised learning challenging. Mintz et al. (2009) propose an alternative approach called "distant supervision" or "weak supervision" which is similar to the concept of weakly labeled examples introduced by Craven and

[1] `https://tac.nist.gov/2017/KBP`.

Kumlien (1999) for the biomedical domain. Given existing entity pairs from a knowledge base (such as Freebase), they extract sentences from a large unlabeled corpus containing those entity pairs and label them with the relation stored in the knowledge base. The underlying assumption is:

> "if two entities participate in a relation, any sentence that contain those two entities might express that relation" (Mintz et al. (2009), pp. 1006).

Handling Noisy Positive Labels

Obviously, this assumption leads to noisy labels (Mintz et al., 2009; Surdeanu et al., 2012). Given, for example, the entity pair (`Obama`, `Hawaii`) which is connected by the relation `born_in`, all the following sentences would be considered to express that relation:

1. "Obama was born in Hawaii." → correct label
2. "Obama gave a speech in Hawaii." → wrong label
3. "Former President Obama was seen in Hawaii." → wrong label

Riedel et al. (2010) analyze examples for three relations extracted from the New York Times (NYT) corpus with distant supervision using Freebase and find that 20–38% of the extracted examples mention the entity pair but do not express the relation between them. Training machine learning models on data with noisy labels may lead to wrong decisions during test time. Therefore, different approaches exist to mitigate the noise from the distant supervision assumption. In a post-processing step, the noisy labels can be cleaned based on patterns or rules, e.g., (Wang et al., 2011; Min et al., 2012; Takamatsu et al., 2012).

Alternative methods relax the distant supervision assumption. Multi-instance learning collects all instances (sentences) mentioning a certain entity pair in a bag and assigns the relation label to the bag under the assumption that at least one of the instances actually expresses the relation (Bunescu and Mooney, 2007; Riedel et al., 2010):

> "If two entities participate in a relation, at least one sentence that mentions these two entities might express that relation" (Riedel et al. (2010), pp. 149).

Originally, multi-instance learning has been proposed in the context of ambiguously labeled data for predicting drug activity (Dietterich et al., 1997). Bunescu and Mooney (2007) and Riedel et al. (2010) connect it to weak supervision and apply it for relation extraction.

Following this line of thoughts, Hoffmann et al. (2011) (`MultiR`) and Surdeanu et al. (2012) ("Multi-instance multi-label" (`MIML`)) develop models which allow entity pairs to participate in multiple relations. Pershina et al. (2014) propose an approach called "guided distant supervision" which extends the `MIML` model to make use of a few manually labeled examples. Grave (2014) learns a classifier based on entity-relation triples in a knowledge base to assign labels to text mentioning the entity pairs, rather than using the triples

directly. More recently, Zeng et al. (2015) integrate multi-instance learning into the loss function of a neural network and Jiang et al. (2016) create a representation for a bag of instances by cross-sentence max pooling. Lin et al. (2016) propose an attention-based approach in which a neural network learns to weight multiple instances of a bag in order to pay more attention to correctly labeled instances than to wrongly labeled ones.

Handling Noisy Negative Labels

While those approaches address the problem of false positive labels, distant supervision can also lead to false negative labels (Xu et al., 2013b). In distant supervision, all sentences mentioning an entity pair that does not have a relation according to a knowledge base will be labeled with an artificial negative relation. However, this leads to false negative labels due to the incompleteness of knowledge bases (see Section 2.1.1): From the absence of an entity pair in a knowledge base, it cannot be inferred that no relation between the two entities exists. Xu et al. (2013b) find through manual analysis that from 1834 sentences with two entities, sampled from the NYT 2006 corpus, 133 (7.3%) express a Freebase relation but only 32 (1.7%) of these relation triples are included in Freebase, leading to 101 (5.5%) false negative labels. This number is even higher than the number of false positive labels introduced by distant supervision (2.7%). This observation emphasizes the need of knowledge base population. Xu et al. (2013b) propose a passage-retrieval approach based on pseudo relevance feedback to reduce false negative labels. Zhang et al. (2013) clean negative labels by using information of other relations the entities participate in. Min et al. (2013) leave potentially negative instances unlabeled and present an extension of `MIML` which can model unlabeled instances. Similarly, Ritter et al. (2013) propose a latent-variable approach to model missing data in both the knowledge base and the text.

Application in this Thesis

To create training data for our slot filler classification models, we also use distant supervision. We clean potentially false negative labels with patterns and apply a self-training strategy to refine the remaining labels (see Section 3.4.2). We do not apply multi-instance learning techniques to slot filling since we have many training samples with only one or a few instances per entity pair. As Takamatsu et al. (2012) mention, the at-least-one assumption of multi-instance learning fails for those cases.

In our experiments with type-aware relation extraction models, we create training data with distant supervision and apply multi-instance learning. For a direct comparison with a state-of-the-art approach, we follow Zeng et al. (2015) and use their loss function for multi-instance training of neural networks. Similar to them, we do not handle false negative labels in that setup.

2.2 Neural Networks

In this section, we describe the neural network layers and training techniques which are relevant to this thesis.

2.2.1 Notation

Throughout this thesis, we will use capital bold letters to refer to matrices, e.g., $\boldsymbol{W}$, and lowercase bold letters to refer to vectors, e.g., $\boldsymbol{x}$. Indices refer to subelements, e.g., x_i denotes the i-th element of vector $\boldsymbol{x}$, W_{ij} the element in the i-th row and j-th column of matrix $\boldsymbol{W}$, and $\boldsymbol{W}_i$ denotes the i-th row of $\boldsymbol{W}$.

The sign $\odot$ as in $\boldsymbol{a} \odot \boldsymbol{b}$ denotes the element-wise multiplication of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. For the concatenation of two vectors, we follow Goldberg and Hirst (2017) and use $[\,;\,]$ as shown in the following example: For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, their concatenation is $[\boldsymbol{a}; \boldsymbol{b}] \in \mathbb{R}^{2n}$.

2.2.2 Linear Perceptron

The simplest form of a neural network, the linear perceptron (Rosenblatt, 1958), consists of one layer and has the following output y (Bishop, 1995):

$$y = g(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{2.1}$$

with $\boldsymbol{x} \in \mathbb{R}^n$ being the input, $\boldsymbol{w} \in \mathbb{R}^n$ a weight vector, $b \in \mathbb{R}$ a bias term and g a threshold activation function:

$$g(a) = \begin{cases} -1 & a < 0 \\ +1 & a \geq 0 \end{cases} \tag{2.2}$$

The weight vector and bias term are learned during training. The dimensionality n depends on the features used to represent the input. The linear perceptron is a linear classifier and can, thus, only classify linearly-separable data correctly (Bishop, 1995).

2.2.3 Neural Network Layers

In contrast to the linear perceptron, the neural networks which are typically used in NLP nowadays are non-linear classifiers and consist of several layers: an input layer, one or more hidden layers with non-linear activation functions and an output layer. Therefore, they are also called "deep" and training them is called "deep learning" (Bengio and LeCun, 2007; Bengio, 2009). The idea is that higher layers can learn more complex or abstract representations based on the representations of lower layers (Bengio, 2009; Goodfellow et al., 2016). In the following subsections, different neural network layers are described. In particular, we present the standard input layer for NLP (lookup layer), four different hidden layers (feed-forward layer, convolutional and pooling layer, recurrent layer and attention layer) and two choices of output layers (softmax layer and conditional random field layer).

Input Layer: Word Embedding Lookup Layer

The input layer of a neural network represents the input as vectors or matrices. For image processing, for instance, the input can be a matrix of continuous pixel values. For text processing, the choice of an input representation is more challenging since text does not have a numeric representation. The most common input layer for text is a lookup layer, which maps each word to a vector, called word embedding. Those word embeddings can be randomly initialized and then updated during training. Alternatively, they can be pre-trained (Erhan et al., 2010). Previous work, e.g., Kim (2014) or Nguyen and Grishman (2015), has shown that pre-trained word embeddings lead to a better performance than randomly initialized embeddings for a variety of sentence classification tasks including relation extraction. In this thesis, we follow those observations and use pre-trained embeddings. The main advantage of pre-training word embeddings is the possibility to make use of large text corpora without labels. Since the training set size for many NLP tasks is limited, this allows exploiting additional resources.

In this thesis, we use the `skip-gram` model of `word2vec` (Mikolov et al., 2013) to train word embeddings on a May-2014 English Wikipedia corpus. The `skip-gram` model trains word embeddings based on the idea that similar words occur in similar contexts (Miller and Charles, 1991; Collobert and Weston, 2008; Erk, 2012; Baroni et al., 2014) and should therefore get similar embeddings (Mikolov et al., 2013). This goes back to the "distributional hypothesis" by Harris (1954) and the work by Firth (1957) who has said "You shall know a word by the company it keeps" (Firth (1957), pp. 179). In NLP, many methods are based on this intuition, such as Brown clusters (Brown et al., 1992). In `skip-gram`, the (randomly initialized) word embedding of each word forms the input to a log-linear classifier, which predicts the embeddings of the surrounding words, usually in a window of five words to the left and five words to the right. The embeddings of the input word and the context words are then updated based on the prediction error. The `skip-gram` model is depicted in Figure 2.2.

The length of the resulting word embeddings is a hyperparameter of the model. In the following equations, we will use n to denote the dimensions of the word embeddings.

Hidden Layer: Fully-Connected Feed-Forward Layer

A feed-forward layer with a hidden layer size of H modulates the input vector $\boldsymbol{x} \in \mathbb{R}^n$ by multiplying it with a weight matrix $\boldsymbol{W} \in \mathbb{R}^{H \times n}$ and adding a bias vector $\boldsymbol{b} \in \mathbb{R}^H$. Afterwards, a non-linear function f is applied (Bishop, 1995). This is specified in Equation 2.3 and depicted in Figure 2.3.

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{2.3}$$

This type of feed-forward layer is often also referred to as "fully-connected layer" since each input neuron is connected to each hidden neuron, as visualized in Figure 2.3.

Popular non-linear functions are sigmoid σ, hyperbolic tangent tanh or rectified linear units `ReLu` (Nair and Hinton, 2010). If not mentioned otherwise in our descriptions or

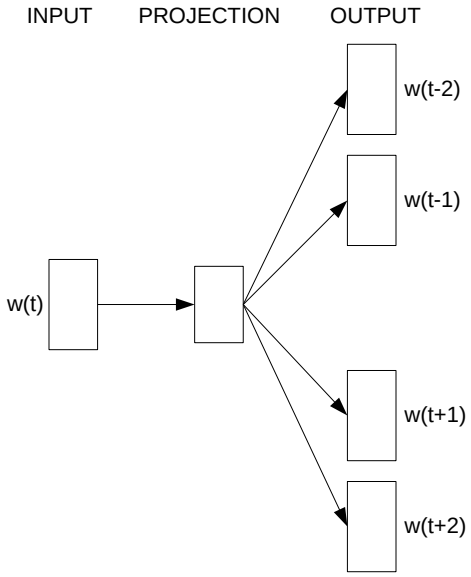


Figure 2.2: `Skip-gram` model for training word embeddings (Mikolov et al., 2013).

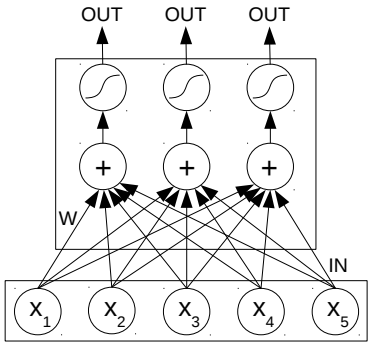


Figure 2.3: Schema of a feed-forward layer with five input and three hidden units.

equations, we apply the hyperbolic tangent as non-linearity, i.e., $f = \tanh$ as depicted in Figure 2.4 and given by Equation 2.4 (Bishop, 1995).

$$\tanh(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} \tag{2.4}$$

We apply the sigmoid function, which is given in Equation 2.5, for transforming the output scores of support vector machines (SVMs) (i.e, the distances to their hyperplanes) to probability-like values between 0 and 1.

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{2.5}$$

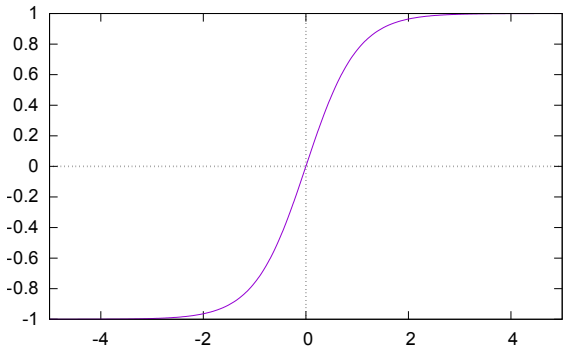


Figure 2.4: Hyperbolic tangent (tanh) function.

Hidden Layer: Convolutional and Pooling Layer

Convolutional neural networks (CNNs), i.e., neural networks with convolutional layers are inspired by the visual system (Bengio, 2009). They can be seen as feed-forward layers with weight sharing. Instead of connecting each input neuron to each hidden neuron, they use filters which are slid over the input and apply convolution to calculate the values of the hidden neurons. Convolutional layers are common in computer vision for creating representations for images. They are able to recognize patterns independent of their position in the input (Goodfellow et al., 2016). They have been applied to phoneme recognition by Waibel et al. (1989) and digit recognition by LeCun et al. (1989) and LeCun (1989). Collobert et al. (2011) have applied them to a range of different NLP tasks. Currently, they are used frequently in the NLP community. Input and filters of the convolutional layers are usually matrices. Thus, they can be used to create a representation for a phrase or even a whole sentence s of length $|s|$, represented as a matrix $\mathbf{X} \in \mathbb{R}^{n \times |s|}$ whose columns are the embeddings of the words of the sentence.

While for convolving images, the height of the filter matrix is usually much smaller than the height of the input image, it is common in NLP to use a filter matrix that spans all dimensions of the word embeddings. Thus, $\boldsymbol{F} \in \mathbb{R}^{n \times w}$ with n being the height and w being the width of the filter. The filter width usually spans 2–5 words, depending on whether bigrams, trigrams, 4-grams or 5-grams should be considered. Some work also applies filters of multiple lengths to combine the corresponding n-grams (Kim, 2014; Nguyen and Grishman, 2015; Vu et al., 2016).

Equation 2.6 shows the equation for convolution for the special case in which the first dimension of the filter matrix matches the first dimension of the input matrix, as explained above.

$$h_i = \sum_{j=1}^{n} \sum_{a=1}^{w} X_{j,i+a-1} \cdot F_{j,a} \tag{2.6}$$

Note that a CNN directly learns the flipped version of the filter, thus we do not need to flip $\boldsymbol{F}$ explicitely in Equation 2.6. For convolving the left and right edges of the input, $\boldsymbol{X}$

can be padded with zero vectors or specially trained padding embeddings (Collobert et al., 2011; Goodfellow et al., 2016). The convolution then results in a vector $\boldsymbol{h} \in \mathbb{R}^{|s|}$. Given the intuition that a single filter learns to recognize only a few specific n-grams which are relevant for the prediction of the network, there is not only one filter applied but often hundreds of filters. The number m of filters is another hyperparameter to the model. Bringing all convolution results together yields a matrix $\boldsymbol{H} \in \mathbb{R}^{m \times |s|}$.

The dimensions of this matrix depend on the length $|s|$ of the input sentence. In order to apply the same network to all sentences of a dataset, a representation independent of the sentence length is needed. Therefore, pooling is applied after convolution. It also makes the extracted features invariant to their position in the input (Goodfellow et al., 2016). Thus, the same n-gram can be recognized independent of where it occurs. This is an important difference to a feed-forward layer. There are different possible pooling functions, such as average or maximum. The maximum function has the advantage that it extracts only the maximum activation from each filter, i.e., only the most important n-gram. Since pooling is applied along the axis correponding to the sentence length (which corresponds to the time dimension when processing a speech signal), it is sometimes referred to as "max pooling over time" (Collobert et al., 2011; Kim, 2014).

Equation 2.7 shows max pooling.

$$P_i^{max} = \max_{t \in |s|} H_{i,t} \tag{2.7}$$

Kalchbrenner et al. (2014) argue that 1-max pooling, i.e., extracting only a single maximum value per filter, is too restrictive and show that k-max pooling performs better in their experiments. We follow them in this thesis and apply k-max pooling with $k = 3$ throughout our experiments. In particular, k-max pooling extracts the k maximum activations from each filter in the order of their occurrence in the filter vector, which corresponds to the order of occurrence of the n-grams in the sentence. We argue that k-max pooling is better suited to NLP tasks than max pooling since (i) it allows more than one extraction of important n-grams per filter, and (ii) it preserves a (limited) amount of sequence information. Applying k-max pooling after convolution yields a matrix $\boldsymbol{P} \in \mathbb{R}^{m \times k}$ as given in Equation 2.8.

$$\boldsymbol{P}_i^{k-max} = [H_{i,t} | \operatorname*{rank}_{t \in |s|}(H_{i,t}) \leq k] \tag{2.8}$$

where $\operatorname{rank}_{t \in |s|}(H_{i,t})$ is the rank of $H_{i,t}$ in the i-th row of $\boldsymbol{H}$ in descending order. Thus, the function extracts the subsequence of the k maximum values for each row in $\boldsymbol{H}$.

Afterwards, a bias matrix $\boldsymbol{B} \in \mathbb{R}^{m \times k}$ is added and a non-linear function f is applied. Again, we use tanh in this thesis.

Equation 2.9 shows the output $\boldsymbol{O} \in \mathbb{R}^{m \times k}$ of the convolutional and pooling layer. In the experiments of this thesis, we use $\boldsymbol{P} = \boldsymbol{P}^{k-max}$

$$\boldsymbol{O} = f(\boldsymbol{P} + \boldsymbol{B}) \tag{2.9}$$

In practice, the results are flattened to a vector $\boldsymbol{o} \in \mathbb{R}^{m \cdot k}$ and often fed into a fully-

connected feed-forward layer, which can, for instance, discover patterns across filters. Figure 2.5 depicts the schema of a convolutional and pooling layer.

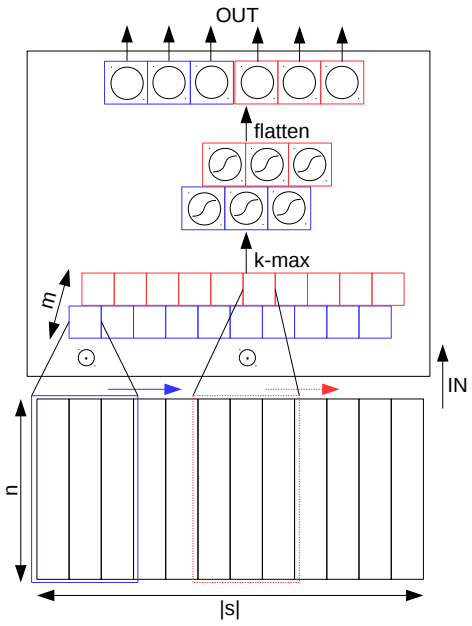


Figure 2.5: Schema of a convolutional layer with two filters and 3-max pooling.

Hidden Layer: Recurrent Layer

Another possibility of processing a sentence, i.e., a sequence of words, is feeding the word embeddings successively into a recurrent layer (recurrent neural network (RNN)). The recurrent layer updates its hidden state after each input vector – like a memory – with the information from the input and the previous hidden state. Equation 2.10 shows the hidden layer update function of a vanilla RNN at time step t, i.e., when processing the t-th word of the sentence.

$$\boldsymbol{h}^t = f(\boldsymbol{W}^{xh}\boldsymbol{x}^t + \boldsymbol{W}^{hh}\boldsymbol{h}^{t-1} + \boldsymbol{b}) \tag{2.10}$$

Matrix $\boldsymbol{W}^{xh} \in \mathbb{R}^{H \times n}$ weights the current input $\boldsymbol{x}^t \in \mathbb{R}^n$, matrix $\boldsymbol{W}^{hh} \in \mathbb{R}^{H \times H}$ the previous hidden state $\boldsymbol{h}^{t-1} \in \mathbb{R}^H$ and $\boldsymbol{b} \in \mathbb{R}^H$ is a bias vector. Note that, for a fast assessment of the equations, the superscripts of the weight matrices indicate which parts of the neurons they connect. For example $\boldsymbol{W}^{xh}$ connects the input neurons $\boldsymbol{x}$ with the hidden neurons $\boldsymbol{h}$.

Equation 2.10 results in a loop of hidden layer updates as depicted in the left part of Figure 2.6. For training with backpropagation, which is described in Section 2.2.4, this loop is unrolled ("backpropagation through time" (Werbos, 1990)), resulting in a deep neural network with a high number of hidden layers (right part of Figure 2.6).

When propagating an error from the last to the first hidden layer of the unrolled network, the gradients are likely to explode or vanish for long input sequences (Pascanu et al.,

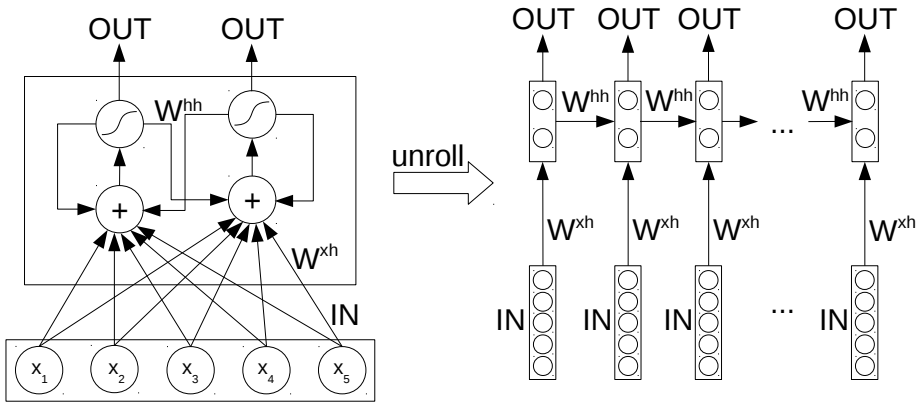


Figure 2.6: Schema of a vanilla recurrent neural network with five input and two hidden units.

2013). To overcome this challenge, Hochreiter and Schmidhuber (1997) propose a long short-term memory (LSTM) architecture and Gers et al. (2000) and Gers et al. (2003) further refine it. An LSTM introduces a variety of gates (input gate $\boldsymbol{i}^t \in \mathbb{R}^H$, forget gate $\boldsymbol{f}^t \in \mathbb{R}^H$ and output gate $\boldsymbol{o}^t \in \mathbb{R}^H$) to the recurrent layer. In the version with peephole connections (Gers et al., 2003), the hidden layer update function from Equation 2.10 becomes:

$$\boldsymbol{h}^t = \boldsymbol{o}^t \odot \tanh(\boldsymbol{c}^t) \tag{2.11}$$
$$\boldsymbol{o}^t = \sigma(\boldsymbol{W}^{xo}\boldsymbol{x}^t + \boldsymbol{W}^{ho}\boldsymbol{h}^{t-1} + \boldsymbol{W}^{co}\boldsymbol{c}^t + \boldsymbol{b}^o) \tag{2.12}$$
$$\boldsymbol{c}^t = \boldsymbol{f}^t \odot \boldsymbol{c}^{t-1} + \boldsymbol{i}^t \odot \tilde{\boldsymbol{c}}^t \tag{2.13}$$
$$\tilde{\boldsymbol{c}}^t = \tanh(\boldsymbol{W}^{xc}\boldsymbol{x}^t + \boldsymbol{W}^{hc}\boldsymbol{h}^{t-1} + \boldsymbol{b}^c) \tag{2.14}$$
$$\boldsymbol{f}^t = \sigma(\boldsymbol{W}^{xf}\boldsymbol{x}^t + \boldsymbol{W}^{hf}\boldsymbol{h}^{t-1} + \boldsymbol{W}^{cf}\boldsymbol{c}^{t-1} + \boldsymbol{b}^f) \tag{2.15}$$
$$\boldsymbol{i}^t = \sigma(\boldsymbol{W}^{xi}\boldsymbol{x}^t + \boldsymbol{W}^{hi}\boldsymbol{h}^{t-1} + \boldsymbol{W}^{ci}\boldsymbol{c}^{t-1} + \boldsymbol{b}^i) \tag{2.16}$$

with t being the index for the current time step, and σ the component-wise sigmoid function. Without peephole connections, the update functions of the gates (Equations 2.12, 2.15 and 2.16) do not depend on the previous cell states $\boldsymbol{c}^{t-1}$.

Chung et al. (2014) show that gated recurrent units (GRUs) (Cho et al., 2014) with only two gates (reset gate $\boldsymbol{r}^t \in \mathbb{R}^H$ and update gate $\boldsymbol{z}^t \in \mathbb{R}^H$) perform similar to LSTMs but are more efficient in training since they introduce fewer additional parameters. The functions for updating the hidden layer of a GRU are given in Equation 2.17 to Equation 2.20.

$$\boldsymbol{h}^t = \boldsymbol{z}^t \odot \boldsymbol{h}^{t-1} + (1 - \boldsymbol{z}^t) \odot \tilde{\boldsymbol{h}}^t \tag{2.17}$$
$$\tilde{\boldsymbol{h}}^t = \sigma(\boldsymbol{W}^{xh}\boldsymbol{x}^t + \boldsymbol{W}^{hh}(\boldsymbol{r}^t \odot \boldsymbol{h}^{t-1})) \tag{2.18}$$
$$\boldsymbol{r}^t = \sigma(\boldsymbol{W}^{xr}\boldsymbol{x}^t + \boldsymbol{W}^{hr}\boldsymbol{h}^{t-1}) \tag{2.19}$$
$$\boldsymbol{z}^t = \sigma(\boldsymbol{W}^{xz}\boldsymbol{x}^t + \boldsymbol{W}^{hz}\boldsymbol{h}^{t-1}) \tag{2.20}$$

with t being the index for the current time step, and σ being the component-wise sigmoid function.

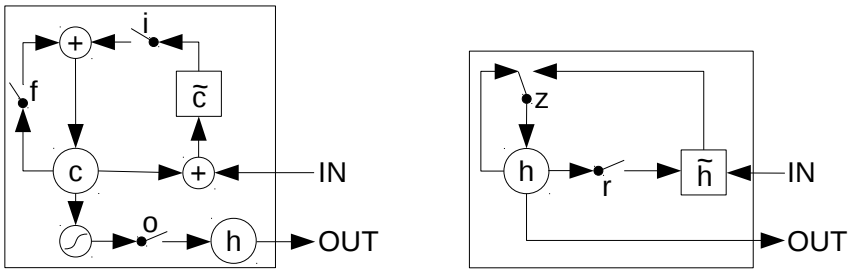


Figure 2.7: Schema of a single LSTM (left) and GRU (right) cell (Chung et al., 2014).

The update gate $\boldsymbol{z}^t$ decides whether the hidden state $\boldsymbol{h}^t$ is updated while the reset gate $\boldsymbol{r}^t$ can ignore the previous hidden state $\boldsymbol{h}^{t-1}$. Figure 2.7 illustrates an LSTM (left) and a GRU cell (right).

Recurrent neural networks are used for different NLP applications, such as language modeling (Bengio et al., 2000; Mikolov et al., 2010), machine translation (Cho et al., 2014; Bahdanau et al., 2015), relation classification (Zhang and Wang, 2015), textual entailment (Rocktäschel et al., 2016), question answering (Hermann et al., 2015) or sequence labeling tasks, for example, part-of-speech tagging (Huang et al., 2015; Gillick et al., 2016; Ma and Hovy, 2016) or named-entity recognition (Huang et al., 2015; Chiu and Nichols, 2016; Gillick et al., 2016; Lample et al., 2016; Ma and Hovy, 2016). In this thesis, we apply them to uncertainty detection in comparison with CNNs.

Hidden Layer: Attention Layer

When applying recurrent neural networks for sentence classification or machine translation tasks, it has been standard to use the last hidden state of the network for the predictions since it accumulates all information from the whole input sequence. However, it is challenging or for some tasks arguably impossible to create a fixed-length vector (hidden state) containing all relevant information for prediction, even when using gates as in LSTMs or GRUs. To mitigate this problem, it is possible to apply max or average pooling over the sequence of intermediate hidden states (Zhang and Wang, 2015; Nguyen and Grishman, 2016; Verga et al., 2016). Another possibility, which has been recently introduced, is attention. It assigns weights to the different intermediate hidden states, depending on their relevance for the prediction. Although first introduced for machine translation with recurrent neural networks as a form of automatic alignment (Bahdanau et al., 2015), it has been successfully integrated into both RNNs and CNNs for a variety of NLP tasks (Bahdanau et al., 2015; Hermann et al., 2015; Rush et al., 2015; He and Golub, 2016; Rocktäschel et al., 2016; Yang et al., 2016c; Yin et al., 2016). Similar to pooling, attention is a selection mechanism that helps the network focus on the most relevant parts of a layer, either an input or a hidden layer. This is especially beneficial for long input sequences, e.g., long sentences or entire documents.

The output $\boldsymbol{a} \in \mathbb{R}^A$ of an attention layer is defined by the following equation:

$$\boldsymbol{a} = \sum_{i=1}^{T} \alpha_i \cdot \boldsymbol{X}_i^\top \tag{2.21}$$

with $\boldsymbol{X} \in \mathbb{R}^{T \times A}$ being the collection of T vectors over which attention should be applied. For RNNs, $\boldsymbol{X}$ is usually the collection of intermediate hidden states. For CNNs, $\boldsymbol{X}$ can be the matrix of convolutional results. In Chapter 4, we show that it can also be beneficial to define $\boldsymbol{X}$ as the matrix of input word embeddings. Each vector $\boldsymbol{X}_i$ (hidden state, n-gram representation or word embedding) is assigned one attention weight α_i. The attention weights are typically normalized to sum to one with the softmax function (see Section 2.2.3):

$$\alpha_i = \frac{\exp(e(\boldsymbol{X}_i))}{\sum_j \exp(e(\boldsymbol{X}_j))} \tag{2.22}$$

with e being a scoring function that computes the attention score for a given input $\boldsymbol{X}_i$. As scoring functions, usually simple neural network layers are applied: Examples are a linear layer or a fully-connected feed-forward layer. When using a linear layer, the attention weights are computed using the following scoring function:

$$e(\boldsymbol{X}_i) = \boldsymbol{w}^\top \boldsymbol{X}_i^\top \tag{2.23}$$

with $\boldsymbol{w} \in \mathbb{R}^A$ being parameters that are learned during training. This approach is also sometimes referred to as "gating" in the literature (Meng et al., 2015). Lin et al. (2017) apply a similar equation but add a non-linearity and call it "self-attention":

$$e(\boldsymbol{X}_i) = \boldsymbol{v}^\top \tanh(\boldsymbol{W} \boldsymbol{X}_i^\top) \tag{2.24}$$

with $\boldsymbol{W} \in \mathbb{R}^{d \times A}$ being a matrix now, $\boldsymbol{v} \in \mathbb{R}^d$ an additional vector of trainable parameters and d another hyperparameter of the network. Alternatively, a fully-connected feed-forward layer can be applied as follows:

$$e(\boldsymbol{X}_i) = \boldsymbol{v}^\top \tanh(\boldsymbol{W}^{xa} \boldsymbol{X}_i^\top + \boldsymbol{W}^{ca} \boldsymbol{c}) \tag{2.25}$$

This equation is applied, for instance, for neural machine translation (Bahdanau et al., 2015) with $\boldsymbol{c} \in \mathbb{R}^B$ being the previous decoder state (i.e., the previous hidden state of the RNN producing the translated output), $\boldsymbol{W}^{xa} \in \mathbb{R}^{d \times A}$ and $\boldsymbol{W}^{ca} \in \mathbb{R}^{d \times B}$. Yang et al. (2016c) build on this equation for document classification but randomly initialize and learn $\boldsymbol{c}$ during training.

Output Layer: Softmax Layer

The most commonly used output layer is the softmax layer. The layer first maps its input $\boldsymbol{h} \in \mathbb{R}^H$ with a linear transformation $\boldsymbol{W}^{hq} \in \mathbb{R}^{C \times H}$ to a vector $\boldsymbol{q} \in \mathbb{R}^C$ of the number of output classes C and then normalizes the entries to obtain a probability distribution

over classes. Equation 2.26 and Equation 2.27 show how the probability for class k can be obtained.

$$P(y = k) = \frac{\exp(q_k)}{\sum_j \exp(q_j)} \tag{2.26}$$

$$\boldsymbol{q} = \boldsymbol{W}^{hq}\boldsymbol{h} + \boldsymbol{b} \tag{2.27}$$

Output Layer: CRF Layer

For token labeling problems, each token of the input sentence should be labeled with an output class, such as a part-of-speech tag or a named-entity class. For this setup, it is possible to train a recurrent neural network with a softmax layer, which computes output class probabilities after processing each input token. However, the activations of the softmax output layer are individual decisions for each token without taking into account the labels of the previous or following tokens. Alternatively, recent work proposes to replace the softmax layer by a linear-chain conditional random field (CRF) layer, e.g., (Collobert et al., 2011; Andor et al., 2016; Lample et al., 2016). CRF models (Lafferty et al., 2001) calculate and optimize the probabilities of *sequences* of predictions instead of individual predictions and have, thus, the ability to correct mistakes in previous decisions and overcome the label bias problem (Lafferty et al., 2001; Andor et al., 2016). Previous work has demonstrated that these models, which globally normalize the score of a whole sequence of predictions, are strictly more expressive than locally normalized ones, which treat individual predictions independent of each other (Raiman and Miller, 2017). For more related work on CRF layers for neural networks, see Section 5.5.2.

The CRF output layer first transforms each part of its input sequence (with length n) linearly to vectors of the number of output classes C (using Equation 2.27) and then creates a matrix $\boldsymbol{Q} \in \mathbb{R}^{C \times n}$ of all input feature scores. In a linear-chain CRF, the score for a class depends on the input feature score of the current time step as well as on the transition score between the class from the previous time step and the current class. These transition scores $\boldsymbol{T} \in \mathbb{R}^{C \times C}$ are learned during training. Assuming that all variables live in the log space, the score for a whole sequence s of length n is, therefore, computed as:

$$score(s) = \sum_{i=0}^{n} T_{s_i s_{i+1}} + \sum_{i=1}^{n} Q_{s_i i} \tag{2.28}$$

where $s_j, 1 \leq j \leq n$ are the constituting items of sequence s and s_0 and s_{n+1} are padding symbols for defining start and end transitions.

In contrast to evaluating a softmax layer at each time step, the CRF layer first computes a score for the whole sequence s and then calculates a sequence probability by normalizing it with the sum over the scores of all possible label sequences S. Equation 2.29 shows this.

$$P(y = s) = \frac{\exp(score(s))}{\sum_{\tilde{s} \in S} \exp(score(\tilde{s}))} \tag{2.29}$$

Since this normalization is on the *global* level of sequences and not on the *local* level of tokens, we use the term "global normalization" interchangeably to "output layer based on structured prediction", following other works (Zhou et al., 2015a; Andor et al., 2016; Raiman and Miller, 2017; Zhang et al., 2017a). As in traditional CRF models, the forward and the Viterbi algorithm (Rabiner, 1989) are applied for computing the sum over the scores of all possible sequences and the score of the best sequence, respectively.

2.2.4 Training

Due to the use of non-linear functions in the network, a closed form solution for optimizing the network, i.e., minimizing its error on the training set, is not possible (Bishop, 1995). Therefore, neural networks are commonly trained with (stochastic) gradient descent. Given a training example or a batch of examples, an objective function is evaluated, which shows how well the network can deal with this example / batch. Then, the weights of the network are updated based on the negative gradients of the objective function with respect to the weights, in order to minimize the error (loss) of the network (Bishop, 1995). In the following paragraphs, we present the cross-entropy loss function, two possibilities to update the weights of a neural network and how the error of the network is backpropagated in order to update the weights of the previous hidden layers. Moreover, we discuss hyperparameter optimization and regularization.

Objective Function

The objective function is the function which is optimized during training. It measures how well the network performs, usually by calculating its loss on the training set. The objective function can consist of one or more loss functions, also called cost functions. In this thesis, we apply cross-entropy loss, the function given by the negative log-likelihood between the training data and model distribution (Bishop, 2006; Goodfellow et al., 2016). The cost function is typically computed as an average over the training set to approximate the following function:

$$\mathcal{L} = -\mathbb{E}_{(x,t)\sim p_{data}} \log P_\theta(t|x) \tag{2.30}$$

with θ being the parameters of the model, p_{data} the empirical distribution defined by the training set, t the true labels and $P_\theta(t|x)$ the probability of the model for output t given input x (Goodfellow et al., 2016).

Parameter Update

For updating the parameters θ of the model, the gradients of the loss function are computed and an optimization step is taken in the direction of the negative gradients in order to reduce the value of the loss. The length of the step depends on the value of the gradients as well as on a learning rate η. Equation 2.31 shows the update step for parameter $W_{ji} \in \theta$.

$$W_{ji} := W_{ji} - \eta \cdot \frac{\partial \mathcal{L}}{\partial W_{ji}} \tag{2.31}$$

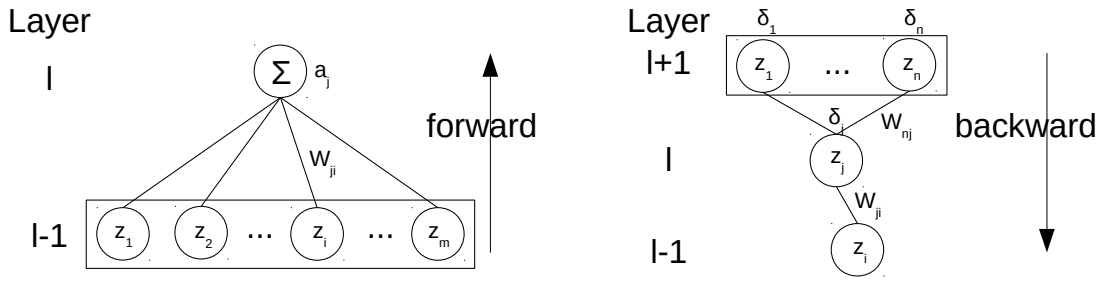


Figure 2.8: Schematic view of a single hidden neuron. Left: forward pass, right: backward pass. Figure inspired by (Bishop, 1995).

Alternatively, it is possible to apply parameter-specific learning rates. An example is *Adagrad* (Duchi et al., 2011), which assigns lower learning rates to frequently updated parameters and higher learning rates to less frequently updated ones. In particular, the update equation for the k-th parameter $\theta_k \in \theta$ becomes:

$$\theta_k := \theta_k - \frac{\eta}{G_{kk}} \cdot \frac{\partial \mathcal{L}}{\partial \theta_k} \tag{2.32}$$

with $\boldsymbol{G}$ being a diagonal matrix which stores the l_2 norms of the previous gradients $\boldsymbol{g}$ of θ. To be more specific, for update step t: $G_{kk}^{(t)} = \sqrt{\sum_{l=1}^{t} (g_k^{(l)})^2}$.

Backpropagation

In order to calculate $\frac{\partial \mathcal{L}}{\partial W_{ji}}$, i.e., the gradient of the objective function with respect to a specific parameter W_{ji}, the error of the network is passed backwards through the network up to parameter W_{ji} by applying the chain rule as shown in Equation 2.33 (Rumelhart et al., 1986; Bishop, 1995, 2006; LeCun et al., 2012; Goodfellow et al., 2016).

$$\frac{\partial \mathcal{L}}{\partial W_{ji}} = \frac{\partial \mathcal{L}}{\partial a_j} \cdot \frac{\partial a_j}{\partial W_{ji}} \tag{2.33}$$

with a_j being the output of the unit which W_{ji} points to, i.e.,

$$a_j = \sum_k W_{jk} \cdot z_k \tag{2.34}$$

$$\frac{\partial a_j}{\partial W_{ji}} = z_i \tag{2.35}$$

with z_k being the activations of the inputs of the unit, c.f., Figure 2.8. (The input activations for the next layer are derived by applying an activation function f to the outputs a_j: $z_j = f(a_j)$.)

The first factor of Equation 2.33, $\delta_j := \frac{\partial \mathcal{L}}{\partial a_j}$, is the error of unit a_j. Substituting δ_j and applying Equation 2.35, then yields:

$$\frac{\partial \mathcal{L}}{\partial W_{ji}} = \delta_j \cdot z_i \tag{2.36}$$

Thus, for getting the partial derivatives for the different weights, it is sufficient to calculate the errors δ_j and multiply them with the input activations z_i. The errors δ_j can be computed recursively by applying the chain rule again, starting from the output layer and then going backwards through the network. This is depicted in the right part of Figure 2.8. For more details, see (Bishop, 2006). Therefore, the update algorithm is called *backpropagation* (Rumelhart et al., 1986).

Minibatch Training

After outlining how the weights of the network can be updated, the question arises when and how often an update step should be performed. In *gradient descent* training, the loss function is evaluated and averaged over the whole training set. This is also referred to as *batch learning* since it works on an entire batch of data at once and performs only one update step of the model parameters θ. In contrast, *stochastic gradient descent* (*online learning*) evaluates the loss function on one training example at a time and updates the parameters after each step (LeCun et al., 2012). Following common practice in the NLP community, we take an intermediate approach and perform *minibatch* training (Bengio, 2012). In particular, we average the loss function over a few training examples – between 10 and 100 – and update the parameters after each of those training batches. The batchsize is treated as a hyperparameter of our models and optimized on the development set.

Development Set

The *development set*, also called validation set, is a held-out part of the training set used to monitor the performance of the model on unseen data during training and estimate its generalization ability (see below). It can also be used for selecting the best hyperparameter values.

Hyperparameter Optimization

Hyperparameters are variables which need to be chosen before the model is trained (Bengio, 2012). Examples are hidden layer sizes, learning rates, minibatch sizes or number of convolutional filters. In order to obtain a good set of hyperparameters, it is common to perform *grid search* over a pre-defined set of values. The model is trained on each possible combination of hyperparameters and then tested on the development set. Based on those results, the best set of hyperparameters is chosen. In this thesis, we follow previous work and use grid search for hyperparameter selection. An alternative option is, for example, random sampling of hyperparameters (Bengio, 2012).

Regularization

Since neural networks are trained on only a limited amount of data, they tend to get biased to the regularities occurring in this set of samples from the real world (Bishop, 1995; Domingos, 2012; Srivastava et al., 2014). This means that although the error on the training set decreases during training, the error on unseen examples starts to increase again at some point (Prechelt, 1998). In order to avoid or reduce such overfitting to the training set and improve generalizability to new test data, we employ the following widely used regularization techniques in this thesis: early stopping on a development set, and l_2 regularization of the parameter values.

Training is typically performed in epochs. In one training epoch, the parameters of the network are updated subsequently on the whole training set (Bengio, 2012). When performing *early stopping*, the performance of the model on the development set is monitored during training, for instance after each training epoch. When the performance on the development set drops, this might indicate overfitting to the training set. In this case, the learning rate can be reduced for the following epochs. After training for a specific number of epochs, not the final model is saved but that intermediate model that has performed best on the development set. The intuition behind this is that this intermediate model has the best generalization ability to unseen data (Bishop, 1995; Prechelt, 1998; Bishop, 2006; Bengio, 2012; Goodfellow et al., 2016).

For l_2 *regularization*, the l_2 norm of the parameters is added to the objective function of the neural network (Bishop, 1995, 2006; Bengio, 2012; Goodfellow et al., 2016). This constrains the network by preventing the parameter values from getting too large and is, therefore, also called "weight decay" (Krogh and Hertz, 1992). It results in layers that put weight on fewer of the input features (Goodfellow et al., 2016). The objective function from Equation 2.30 changes to:

$$\tilde{\mathcal{L}} = \mathcal{L} + \lambda \cdot \theta^\top \theta \tag{2.37}$$

with θ being the parameters of the model and λ being the weight for the regularization term.

Other possible regularization techniques are, for instance, dropout (Srivastava et al., 2014), training with noise (Bishop, 1995) or soft weight sharing (Nowlan and Hinton, 1992).

2.2.5 Implementation of Neural Networks

We implement the neural networks developed in this thesis with Theano (Bergstra et al., 2010; Theano Development Team, 2016), a widely-used python-based framework for implementing and training neural networks. In particular, we define a computation graph, which is compiled and can be run either on CPUs or a GPU. Due to internal graph optimization steps and the generation of optimized C++ or CUDA code during compiling, the network can be trained efficiently despite the python-based interface. The code for the different chapters of this thesis is implemented as different projects. However, several modules, such as input layer, convolutional or feed-forward layers are re-used across tasks.

Chapter 3

Slot Filling

Erklärung nach §8 Absatz 5 der Promotionsordnung: This chapter covers work published at the peer-reviewed international conference North American Chapter of the Association for Computational Linguistics (NAACL) in 2016 (Adel et al., 2016), as well as the slot filling system presented at the Text Analysis Conference (TAC) in 2014 and 2015 (Adel and Schütze, 2014, 2015). Moreover, some of the coreference resolution investigations are based on (Adel and Schütze, 2017a), which has been made available on a pre-print server.

For a declaration of co-authorship and attribution, see page xxv and following.

3.1 Task

The TAC KBP slot filling task addresses the challenge of gathering information about entities (persons, organizations or geo-political entities) from a large amount of unstructured text data (Ji and Grishman, 2011; Surdeanu, 2013; Surdeanu and Ji, 2014). The goal is to fill the information into an either empty or incomplete knowledge base. As defined in Section 2.1.1, the formal description of the task is to find e_2 for (e_1, r, e_2) for given entity e_1 and relation r based on evidence in a large text corpus. In this chapter, we will interchangeably use the terms "relations" and "slots" and refer to second relation arguments e_2 as "slot fillers". The different relations (slots) which are considered in the slot filling task are listed in the appendix in Table A.1.

3.1.1 Input

The slot filling task is query-based, i.e., the input to the slot filling systems are queries. A query consists of the name of the entity (called "query entity" in the remainder of this chapter), the id of a document in which the query entity appears, as well as begin and end offsets of the entity in the document. This given document (referred to as "starting point") can be used for disambiguating entities with the same name. Moreover, the type of the entity (`PER`, `ORG` or `GPE`) is provided and the slot which needs to be filled. The official slot

filling evaluations distinguish between zero-hop and one-hop queries. For zero-hop queries, one slot is provided and needs to be filled, thus e_2 needs to be determined for a triple (e_1, r, e_2). One-hop queries consist of two slots. For the first relation r_1, e_2 with (e_1, r_1, e_2) needs to be found and for the second relation r_2, e_3 with (e_2, r_2, e_3) should be filled based on the previous result e_2 of the system. This corresponds to taking one hop, i.e., following two edges, in the corresponding knowledge graph.

The following one-hop query is an example input for a slot filling system (fictional query with a random query id and document id):

```
<query id="CSSF15_ENG_012abc3456">
 <name>Apple</name>
 <docid>NYT_ENG_20131203.4567</docid>
 <beg>222</beg>
 <end>226</end>
 <enttype>org</enttype>
 <slot0>org:founded_by</slot0>
 <slot1>per:date_of_birth</slot1>
</query>
```

To answer this query, a slot filling system needs to access a given document collection and extract the founders of the organization `Apple` as well as their dates of birth. The slots can be single-valued (like `per:date_of_birth`: a person has only one date of birth) or list-valued (like `org:founded_by`: a company might have more than one founder).

3.1.2 Output

The output of the system is a tab-separated string consisting of the following fields:

- query id (the id from the input query)
- slot (the slot from the input query, i.e., in the example above: `org:founded_by` for the first relation and `per:date_of_birth` for the second relation)
- proof (the document id as well as start and end offsets for a text snippet that supports the slot filler)
- slot filler (i.e., the extracted e_2 or e_3)
- type of slot filler (can be `PER`, `ORG`, `GPE` or `STRING`)
- document id as well as start and end offsets for the slot filler
- confidence score of the system

In the official evaluations, the outputs of the participating systems are assessed manually. Individual results for both hops as well as overall results are reported.

3.1.3 Challenges

The slot filling task comprises a variety of NLP challenges of various characteristics (Min and Grishman, 2012; Pink et al., 2014; Surdeanu and Ji, 2014). The most important ones are given in this subsection.

Alternate Names for the Same Entity

"Cassius Marcellus Clay Jr.", "Muhammad Ali" and "Mohammad Ali", for example, all refer to the American professional boxer. Since the given query only provides one of these names, the slot filling system needs to access additional sources of background knowledge, which inform it about possible alternate names.

Ambiguous Names for Different Entities

"Michael Jordan", for example, may refer to an American professional basketball player, to an actor, to an English business man, to a professor or to any other person with that name. In order to avoid filling slots of the professor Michael Jordan with information about the basketball player, entity disambiguation, such as entity linking, needs to be performed.

Misspellings

Especially in user-generated web documents, misspellings are common. Misspellings of names may prevent systems from finding the query entity in the document and, therefore, lead to the disregard of possibly relevant documents or sentences. Misspellings in sentences expressing the query relation may cause wrong predictions of the slot filler classification component.

Coreference Resolution

Entities are often referred to with anaphoric pronouns or noun phrases, such as "the 60-year-old" or "the New York–based company". For extracting possible slot filler candidates and correctly classifying whether there is a relation between the filler candidate and the query entity, those anaphoric mentions need to be resolved.

Location Inference

If the query relation is, for example, `per:country_of_birth`, i.e., the country of birth of a person, but the slot filling system only extracts a proof for the city of birth in the document collection, it is possible to infer the country of birth. To accomplish this, the slot filling system needs to extract a proof sentence saying which country the extracted city belongs to.

Cross-document Inference

Location inference often happens across documents. There are also other possible cross-document inferences. For example, one document might mention relevant information about the current US president while the name of the president is given in another document.

Documents from Different Domains

The provided text corpus contains documents from different domains (2009–2014: news, web documents and discussion forum entries, 2015: news and discussion forum entries). Domain mismatches pose challenges to machine learning models, as shown for slot filler classification models in Section 3.5.2.

Pipeline Effects

State-of-the-art slot filling systems consist of a pipeline of different modules (Ji and Grishman, 2011). This means that errors from one module are propagated to the next module. Especially recall losses cannot be recovered in the subsequent modules (Pink et al., 2014). Errors in sentence boundary detection or named entity recognition might not immediately lead to losses but might complicate the task of the slot filler classification component.

Relation Extraction

Relation extraction itself is a challenging NLP task since a relation can be expressed in many different ways. Many slot filling systems rely on pattern matching approaches, which provide a high precision but cannot cover the large variety of natural language. Apart from that, relation extraction in the context of slot filling has to deal with additional challenges coming from different domains from the text documents or from pipeline effects: The extracted slot filler candidates as well as the extracted possible proof sentences may be incomplete or too long. Moreover, Zhang and Wang (2015) show that in a TAC KBP–based dataset, the context lengths are much larger than in two traditional benchmark datasets for relation extraction (SemEval-2010 task 8 (Hendrickx et al., 2010) and NYT+Freebase (Riedel et al., 2010)). Huang et al. (2017) further argue that the query entity and filler candidate are often separated by more words than in traditional relation extraction datasets. This makes the extraction task more challenging. Additionally, there is no official training dataset available and most systems rely on noisy training examples labeled with distant supervision (see Section 2.1.2).

One-hop Queries

The challenge of one-hop queries can also be seen as a pipeline effect. The query entities for the second slot are given by the slot fillers from the first slot. Fillers for the second slot will only be assessed as correct if both the query entity and the slot filler are correct.

Thus, a wrong answer in the first slot will inevitably lead to errors in the second slot. This can have negative effects on both the precision and the recall of the system. Recall decreases when the system fails to find the correct answer for the first slot since it is not possible to find correct answers for the second slot in that case. Precision decreases when the system extracts a wrong slot filler for the first slot which can have many (also wrong) answers in the second slot. Consider a query with the following two slots as an example: `<slot0>org:country_of_headquarters</slot0>` and `<slot1>gpe:residents_of_country</slot1>`. If the correct filler for the first slot is "Iceland" but the system wrongly extracts "USA", there is a high chance that it will extract a large number of slot fillers for the second slot which will all be assessed as wrong because the query entity is wrong.

3.2 The CIS Slot Filling System

For answering the input queries, a variety of natural language processing steps needs to be performed. Our slot filling system addresses most of the challenges mentioned in Section 3.1.3 (except for cross-document inference, which we only consider in the context of location inference). It approaches the slot filling task in a modular way. This has several advantages, including extensibility, componentwise analyzability (see Section 3.6.2) and modular development. In the following section, the different components of our system are described. Figure 3.1 provides an overview of the system. The blue boxes show its components, the arrows illustrate the information flow between them. Given the input (the query and the document collection), an alias component extracts aliases for the query entity and an information retrieval component retrieves documents mentioning the name of the query entity. The retrieved documents are further filtered by an entity linking component, limiting the set of documents to documents mentioning the entity from the query (as opposed to another entity with the same name). Afterwards, the candidate extraction component selects sentences mentioning the query entity and extracts a set of slot filler candidates. These candidates are scored by a slot filler classification component depending on how likely they fill the slot from the query. Finally, the postprocessing component selects the best slot fillers for output. We provide the code of our system at `http://cistern.cis.lmu.de/CIS_SlotFilling`.

3.2.1 Component Description

Alias Component

In order to address the challenge of multiple names for the same entity, the query is expanded with aliases of the query entity. For this, we use a list of possible aliases, which we have compiled based on Wikipedia redirects extracted with the Java-based Wikipedia interface JWPL (Ferschke et al., 2011).[1] If, for example, a user queries "Barack H. Obama" on

[1] We use a Wikipedia dump from July 2014.

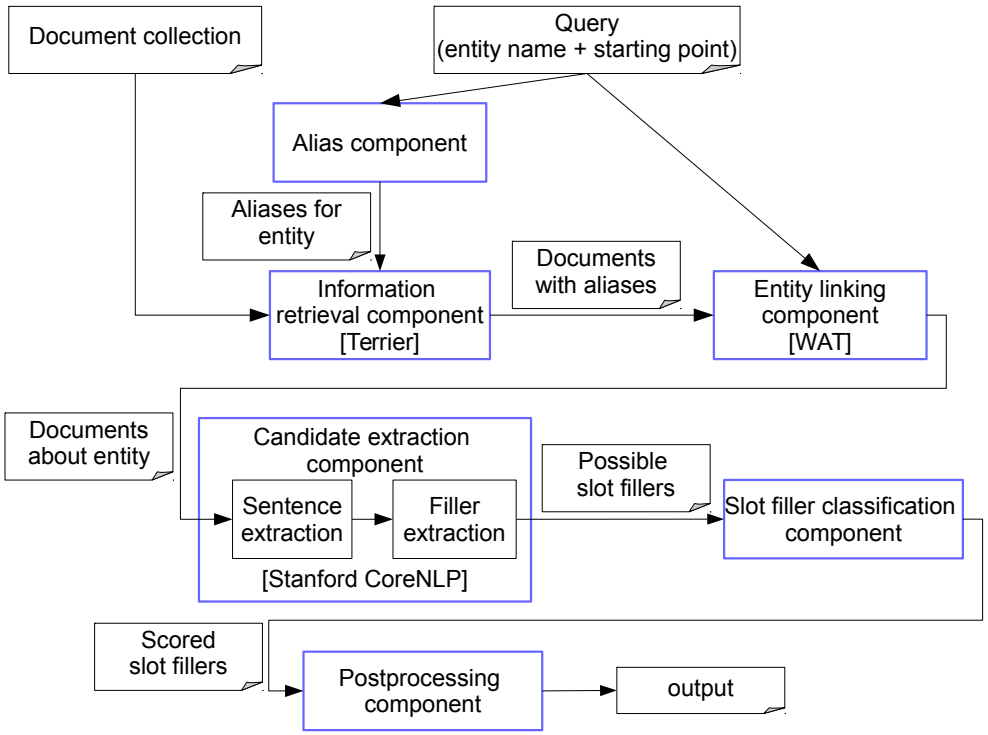


Figure 3.1: System overview: Basic components of the CIS slot filling system.

Wikipedia, he/she will be redirected to the page of "Barack Obama". From this information, "Barack H. Obama" can be extracted as an alias for "Barack Obama". Unfortunately, the data resulting from those redirects can sometimes be noisy. For example, querying for "Eric Iler" redirects to "Jamie Lee Jones" who is a different entity but related via news stories. Another example is "Gaynor Holmes" who will be extracted as an alias for "BBC Scotland", an entity with even a different type. Therefore, we apply several constraints for cleaning the list of aliases, such as minimum number of characters of alias names or no aliases with another named entity type as the given entity.

After expanding the query with possible aliases, we also apply some rules based on the type of the entity (which is provided by the query): If the query entity is an organization, we also add various company-specific suffixes to the list of aliases, such as "Corp", "Co", "Inc". If the query entity is a person, we include nicknames taken from the web[2] into the list of aliases.

A high-level overview of the alias component is depicted in Figure 3.2.

[2]male nicknames: `http://usefulenglishru/vocabulary/mensnames`,
female nicknames: `http://usefulenglishru/vocabulary/womensnames`.

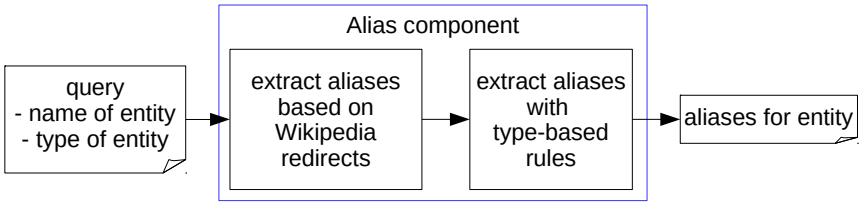


Figure 3.2: Overview of the alias component.

Information Retrieval Component

Based on the name of the query entity and its aliases, documents mentioning this name are retrieved to reduce the large search space to a limited number of relevant documents. For this retrieval, we do not use all extracted aliases since this has led to many false positive retrievals in initial experiments. Instead, we compute the Levenshtein distances (Levenshtein, 1966) between the query name and each alias and use only that alias with the lowest distance to the query name. We call it "IR alias". This helps to cover spelling variations and, thus increase recall, while keeping the number of false positives low. Note that we use the full set of aliases later in the candidate extraction component.

For the document retrieval, we apply the open-source information retrieval (IR) system `Terrier` (Ounis et al., 2006). To be able to use it with the corpus provided by the slot filling task, we clean the corpus, e.g., remove html tags from it, and index the documents with `Terrier`. When searching for the query entity in the corpus, we create the following queries for information retrieval:

- $q_1 = \bigwedge_i t_i$, the conjunction of the tokens t_i of the query entity name
- $q_2 = \bigwedge_i a_i$, the conjunction of the tokens a_i of the IR alias
- $q_3 = \bigvee_i t_i$, the disjunction of the tokens t_i of the query entity name

For geo-political entities, we only use q_1 and q_2 (conjunction of tokens). In prior experiments, we have also investigated phrase queries but they have not worked well with spelling variations, resulting in a considerably lower overall recall of the system. After retrieval, we instead filter the extracted list of documents by fuzzy string matching with the whole name and IR alias to skip documents mentioning both the first and the last name of a person but not in a phrase.

The results of `Terrier` for q_1 and q_2, the two queries with conjunction, are merged and sorted according to the relevance score `Terrier` assigned to them. Afterwards, they are restricted to the top 300 documents. If q_1 and q_2 resulted in less than 300 documents, the top results from q_3 are added. The reason is that we expect the results from q_3 to be noiser than the results from the two conjunction queries. Therefore, we only use them if necessary to provide the slot filling pipeline with a reasonable number of documents. Thus, the number of documents which are passed to the next pipeline components is $min(|r_{q_1}| + |r_{q_2}| + |r_{q_3}|, 300)$ with $|r_{q_x}|$ denoting the number of documents which `Terrier` returned for query q_x.

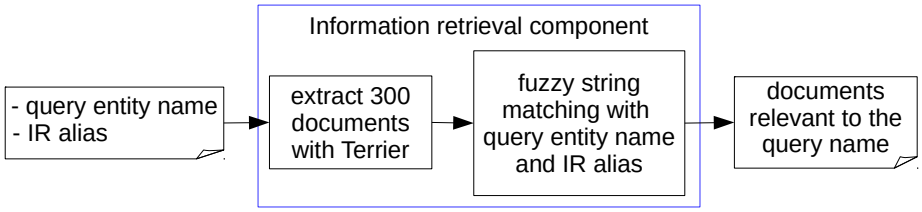


Figure 3.3: Overview of the information retrieval component.

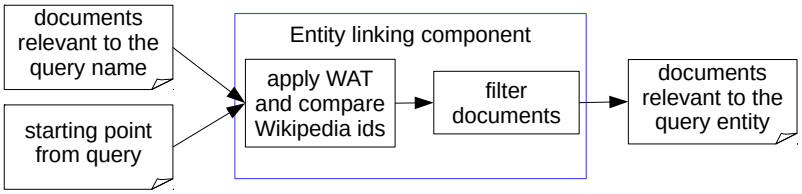


Figure 3.4: Overview of the entity linking component.

Note that we use the term "relevant" to denote those top documents returned by `Terrier`. An overview of the information retrieval component is provided in Figure 3.3.

Entity Linking Component

In order to cope with the challenge of ambiguous names, i.e., different entities having the same name, we apply entity linking (EL). In particular, we use the entity linking system `WAT` (Piccinno and Ferragina, 2014), which links entities in a given sentence to Wikipedia based on co-occurring entities. It outputs the Wikipedia ids of all entities occurring in the input. We apply `WAT` to the query entity and the context given by the starting point and offsets of the query. If the query entity cannot be linked to Wikipedia, we do not perform entity linking but process all documents mentioning the query entity name. If `WAT` extracts a Wikipedia id for the query entity, we apply it to all possible mentions of the query entity found in the documents extracted by `Terrier` and compare their Wikipedia id to the Wikipedia id of the query entity. If they do not match, we delete the document from the set of documents.

For the following pipeline steps, we limit the set of documents to the top 100 documents (ordered by the relevance score from `Terrier`, see Section 3.2.1, and filtered by the entity linking component). This number has been determined heuristically based on prior experiments: We have observed that 100 documents are a good trade-off between recall and processing time.

Figure 3.4 illustrates the entity linking component.

Candidate Extraction Component

The candidate extraction component extracts possible slot fillers (filler candidates) based on sentences mentioning the query entity. Figure 3.5 provides an overview of the different

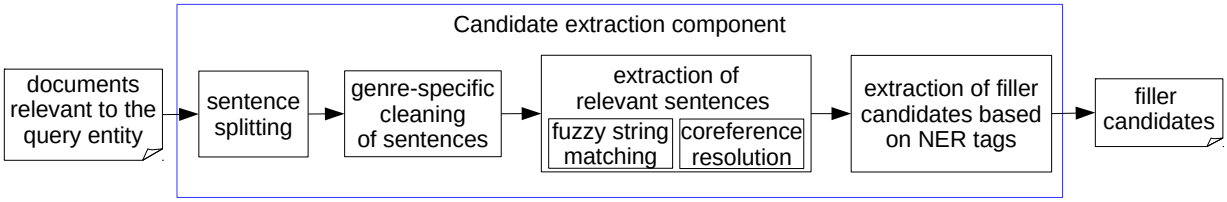


Figure 3.5: Overview of the candidate extraction component.

steps performed.

Genre-specific Document Processing. First, the documents are split into sentences using Stanford `CoreNLP` (Manning et al., 2014). Then, they are cleaned, e.g., from html tags.

The TAC 2015 evaluation corpus consists of documents from different genres: news and discussion forums. Since those genres have different characteristics, our document processing is genre-dependent: For discussion forum documents, we apply additional cleaning steps, such as ignoring text inside `<quote>` tags and normalizing casing of strings (e.g., mapping "sErVice" to "service"). We also use different `CoreNLP` flags for the different genres when performing sentence splitting. An initial analysis has shown that the genre-specific processing is crucial for the precision of the system since it reduces the noise in the input to the following pipeline components.

Sentence extraction. In order to find all sentences mentioning the query entity, we apply two strategies: fuzzy string matching with all the aliases of the query entity name, and automatic coreference resolution.

For string matching, we compute the Levenshtein distance and apply a heuristical threshold dependent on the type and length of the string: A string is regarded a mention of the query entity if it exactly matches one of its aliases in case it is an acronym or if less than $\frac{1}{7}$ of its characters are different to one of the aliases otherwise. Moreover, for a fuzzy string match, our system still requires the first letters to be the same, with the exception of $K \leftrightarrow C$, $I \leftrightarrow Y$ and $F \leftrightarrow P$, which can be interchanged to cover spelling variations. Note that fuzzy string matching is more generally applicable to sentence extraction than, for example, `WAT` since it also works for rare entities without a Wikipedia page.

For automatic coreference resolution, we apply Stanford `CoreNLP`. The importance of coreference resolution for slot filling is widely acknowledged (Min and Grishman, 2012; Surdeanu and Ji, 2014). In our system, we use coreference resolution not only for the query entity but also when extracting the filler candidate if the type of the filler candidate is `PER`. Since this is the case for 32% of all slots, coreference resolution for filler candidates can improve the recall of the system considerably. For example, for the slot `org:students` and the sentence "He went to University of Munich", we would extract the referent of the pronoun "he" as a possible filler candidate. In Section 3.6.4, we show the positive impact of coreference resolution on the slot filling pipeline.

Errors of automatic coreference resolution affect the final performance of the slot filling

system. While coreference resolution increases the number of false positive filler candidates,[3] we find that almost all of those can be ruled out by the slot filler classification component of our system. The errors from which the system cannot recover in the subsequent modules are recall losses (Pink et al., 2014). In a manual analysis, we found three common errors of coreference resolution which can lead to recall losses:

- Wrongly linked pronoun chains: The pronoun mentions are linked to the wrong entity.
- Unlinked pronoun chains: The chains only consist of pronouns. Without an explicit postprocessing, it is, therefore, not possible to determine whether they refer to the query entity or to another entity.
- No recognition of nominal anaphora: Phrases like "the 30-year-old" are often not recognized as being coreferent to an entity.

To cope with the last error category, our system employs the following heuristic: If the entity from the query occurs in sentence t and sentence $t+1$ starts with a phrase like "the XX-year-old", "the XX-based company", "the XX-born" and this phrase is not followed by another entity, this phrase is considered to be coreferent to the query entity.

Filler candidate extraction. Filler candidates are extracted based on a manually compiled mapping of slots to expected named entity types of the fillers. First, `CoreNLP` is applied to tag the words of the extracted sentences with named entity tags. The 7-class tag list is `PER, ORG, LOC, DATE, NUMBER, MISC, O` (Finkel et al., 2005).

Second, the system extracts possible filler candidates based on the mapping. For example, the slot `per:date_of_birth` can only have `DATE` fillers, thus, the system considers all words and phrases tagged with `DATE` as filler candidates. A slot like `org:members`, in contrast, can have organizations, locations and persons as fillers. Therefore, the system considers all words and phrases tagged with either `PER`, `ORG` or `LOC` as filler candidates.

The candidate extraction step is different for slots with string fillers, namely `per:title`, `per:charges`, `per:religion`, `org:political_religious_affiliation` and `per:cause_of_death`. For them, we automatically compile lists of possible fillers from Freebase (Bollacker et al., 2008) and manually clean them in order to improve their precision. Another exception is the slot `org:website` for which we apply a regular expression which matches URLs.

Finally, the candidate extraction component filters out impossible filler candidates, such as floating point numbers for `org:number_of_employees` or `per:age`.

Slot Filler Classification Component

The extracted filler candidates are classified into valid and invalid slot fillers based on their textual context. This is a relation classification task but poses the additional challenges that no official training data is available and that the classifier inputs are the results from

[3] Pink et al. (2014) even argue that the higher precision without coreference resolution might lead to better overall results than the higher recall with coreference.

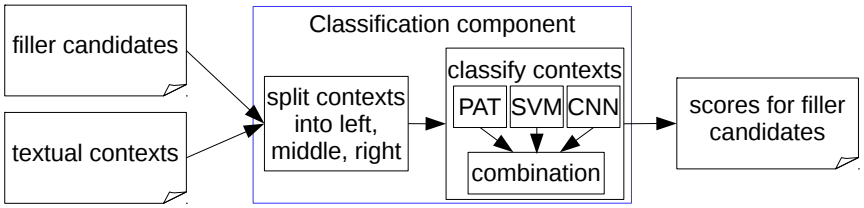


Figure 3.6: Overview of the slot filler classification component.

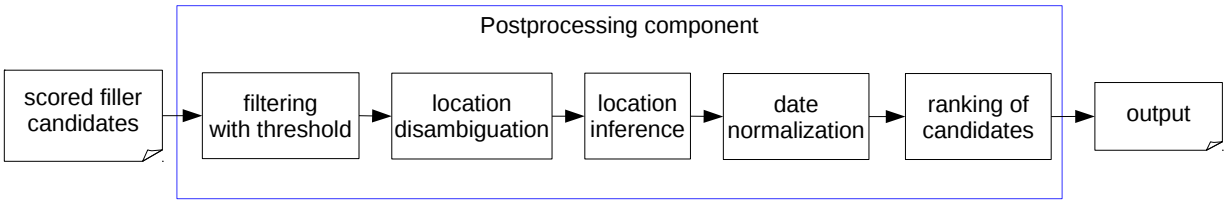


Figure 3.7: Overview of the postprocessing component.

previous pipeline steps and can, thus, be noisy (e.g., due to wrong coreference resolution, wrong named entity recognition or insufficiently cleaned text snippets from discussion forum documents) or consist of too long or incomplete sentences (due to erroneous sentence splitting).

For classifying a filler candidate with its context as correct or wrong, we use the distantly supervised patterns published by Roth et al. (2013), and train support vector machines (SVMs) as well as convolutional neural networks (CNNs). The scores of those models are combined by linear interpolation. The interpolation weights are tuned based on previous TAC evaluation data.

An overview of the slot filler classification component is given in Figure 3.6. Section 3.3 provides more information on this module, which can be considered as one of the most important parts of the slot filling system since it directly influences the output.

Postprocessing Component

The last step of the slot filling pipeline is a postprocessing of the results. Afterwards, the valid filler candidates are output along with their confidence scores from the slot filler classification component and their supporting contexts. Figure 3.7 illustrates the postprocessing component.

Filtering of Filler Candidates. The classification module assigns scores to each filler candidate, which indicate how likely it is a valid filler for the query slot given its surrounding context. The postprocessing component discards all filler candidates with a score below a specific threshold. We tune slot-specific thresholds in order to maximize the slot filling system performance on previous TAC evaluation data. For the second slot (hop 1) of one-hop queries, we increase the thresholds by 0.1 in order to mitigate the challenge of many false positive answers as described in Section 3.1.3.

Location Disambiguation. In our slot filler classification module, we do not distinguish between cities, states or provinces, and countries (see Section 3.3.1). For the system output, however, the extracted locations need to be disambiguated. To decide to which category a location belongs, we employ city, state and country lists.[4]

Location Inference. If the system has extracted a city or state while the slot given in the query is a state or country, the postprocessing module automatically infers the corresponding state or country based on city-to-state, city-to-country and state-to-country mappings extracted from Freebase.

Date Normalization. The expected output format for dates is YYYY-MM-DD. Therefore, the extracted fillers for date slots are normalized to match this format.

Ranking of Filler Candidates. Finally, the classification score of the filler candidates is used to rank the extracted slot fillers. For single-valued slots, only the top filler candidate is output. For list-valued slots, the top N filler candidates are output. The threshold N is slot-dependent and has been determined heuristically on previous evaluation data in order to balance precision and recall of the system.

3.3 Models for Slot Filler Classification

In this section, the models used for slot filler classification are described: Those are a pattern matcher based on distantly supervised patterns, support vector machines and convolutional neural networks. Afterwards, we explain how we combine the outputs of the different models.

3.3.1 General Remarks

In general, we apply all three models to each slot filler candidate and its context, and merge their results afterwards by linear interpolation with slot-dependent weights. However, there are slots for which only little training data (less than 100 positive examples, see Table A.3) could be extracted. Those slots are `per:charges`, `per:other_family`, `per:religion`, `org:number_of_employees_members`, `org:date_dissolved`, `org:shareholders`, `org:political_religious_affiliation`. For them, we do not train classifiers but only apply pattern matching.

Label List

We do not use the whole slot list as labels for the classification models. In particular, we use only one label for each slot and its inverse to avoid redundant training. To model inverse slots, we reverse the two relation arguments in the input. We also merge the "city", "country" and "state-or-province" labels to one "location" label since we expect their fillers to appear in the same contexts.

[4] from Freebase, `http://www.listofcountriesoftheworld.com`, and Wikipedia (`http://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations`).

	binary	multiclass
input	class-specific data	data from all classes
output	2 classes (yes or no)	$\|C\|$+1 classes (one negative class)
negative examples	negative only for current class	negative for all classes
number of models	$\|C\|$	1

Table 3.1: Comparison of binary and multiclass models. $|C|$ denotes the number of positive classes.

Input for the Models

As input, we only use information which is directly available from the input context, i.e., words and combinations of words, but no hand-crafted features like part-of-speech tags or dependency paths. Thus, the models can learn which input words or phrases are relevant. With this, we want to avoid additional noise in the inputs due to wrong tags or wrong dependency paths. For the support vector machine and the convolutional neural network, we split the input context at the relation arguments (query entity and slot filler candidate) into three parts: left of the arguments (left context), between the arguments (middle context) and right of them (right context). The relation arguments are not part of the input to prevent the models from overfitting to entities of the training dataset. We expect this to be useful especially for slot filling since many queries include rare or unseen entities.

Binary Models vs. Multiclass Models

Having binary models, i.e., one model per slot, facilitates extensions of the slot list with more slots since the existing models do not need to be re-trained. However, the more slots there are, the more models need to be optimized, maintained and evaluated. Also, comparisons of confidence scores across the individual models might not always be justifiable theoretically since there is no objective across slots which correlates their scores to each other. Therefore, we train both binary and multiclass models and compare their results in our experiments. Table 3.1 provides a direct comparison of the most important differences of binary and multiclass models.

3.3.2 Distantly Supervised Patterns

Pattern matchers can provide high precision judgements for slot filler classification. However, they suffer from coverage problems, especially when they are manually created. Thus, they provide only low recall results. Roth et al. (2013) publish distantly supervised patterns, which consist of the token sequences between two relation arguments found by distant supervision (see Section 2.1.2) using Freebase relations and the TAC source corpus. Due to the distant supervision, we expect their recall to be higher than with manual patterns but their precision to be lower. To increase precision, Roth and Klakow (2013) apply noise reduction methods, which reduce the number of false positive matches. We use the resulting pattern set for a pattern matching component in our slot filling system. Table A.2 in the appendix provides statistics about the pattern set.

3.3.3 Support Vector Machines

Support vector machines (SVMs) are maximum-margin classifiers. This means that they try to find a linear decision boundary between two classes which maximizes the distance ("margin") to both classes and should, thus, be able to generalize better to unseen data (Vapnik, 1995; Bishop, 2006). The decision boundary can be unambiguously defined by a subset of data points, so-called "support vectors", which are located at the boundaries of the maximum-margin hyperplane. In order to discriminate data which is not linearly separable in the original feature space, SVMs with non-linear kernels can be applied that move the data into a higher-dimensional space in which the data points might be linearly separable (Bishop, 2006). In this thesis, we apply linear support vector machines since they are often sufficient for natural language processing tasks. The main reason is that bag-of-word text representations have a high number of features and, therefore, mapping the data to a higher dimensional space does not improve performance (Hsu et al., 2003).

Input Features

For the SVMs, we use the following set of features:

- 1-gram, 2-gram and 3-gram bag-of-word vector of the left context
- 1-gram, 2-gram and 3-gram bag-of-word vector of the middle context
- 1-gram, 2-gram and 3-gram bag-of-word vector of the right context
- skip 3-gram, skip 4-gram, skip 5-gram bag-of-word vector of the middle context
- flag indicating whether the query entity or the filler candidate appears first in the sentence

A skip n-gram keeps only the first token and the last token of an n-gram and wildcards the tokens in the middle. For example, "founder of" would be a possible skip 4-gram of the context "founder and director of".

Implementation

The SVMs are implemented with liblinear (Fan et al., 2008) by using LinearSVC from scikit learn.[5] For training the multiclass SVM, we apply automatically adjusted class weights and the one-vs-rest training strategy. To extract confidence values from the SVMs, we apply the sigmoid function (cf., Equation 2.5), which maps the distances of the examples to the decision hyperplane to a value $v \in [0, 1]$.

[5]For documentation, see `http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`.

3.3.4 Convolutional Neural Networks

Convolutional neural networks (CNNs, see Section 2.2.3) are successfully applied to several NLP tasks (Collobert et al., 2011; Kalchbrenner et al., 2014), including relation classification (Zeng et al., 2014; dos Santos et al., 2015). We propose to integrate them into a slot filling pipeline. In contrast to prior work, we train them on noisy distantly supervised training data and use them in a pipeline, which can provide them with noisy or wrong inputs. Our results show that they are still able to classify the relations and improve the final performance of the system.

CNNs are promising models for slot filler classification because of the following reasons:

(i) They recognize n-gram patterns independent of their position in the sentence.

(ii) They recognize similar words or phrases since they use word embeddings as input and create internal phrase and sentence representations.

This section describes `contextCNNs`, the convolutional neural networks we have designed for the slot filler classification component. Source code and implementation details are provided at `http://cistern.cis.lmu.de/CIS_SlotFilling`.

Input

For the `contextCNNs`, we split the sentence at the positions of the relation arguments into three contexts (left, middle and right) and extract the following inputs:

- embeddings of the words of the left context
- embeddings of the words of the middle context
- embeddings of the words of the right context
- flag indicating whether the query entity or the filler candidate appears first in the sentence

The words of the input sentence are represented with word embeddings trained with `word2vec` (Mikolov et al., 2013) on English Wikipedia (cf., Section 2.2.3).

Model Architecture

The network applies convolution to each of the three contexts individually with convolutional filter weights shared across them. The number of filter matrices $m \in \{100, 300, 1000\}$ is tuned on the development set. The hyperparameters of all models are provided in the appendix in Section A.5. After convolution, 3-max pooling (Kalchbrenner et al., 2014) is applied. For equations of the convolutional layer and k-max pooling, see Section 2.2.3.

After 3-max pooling, the results are concatenated to one large vector and extended with a flag v indicating whether the entity or the filler candidate appears first in the sentence.

The final vector is passed to a multi-layer perceptron with one hidden layer, which creates a sentence representation $\boldsymbol{s} \in \mathbb{R}^H$ by combining the results of the different convolutional filters.

Finally, a softmax layer is applied to compute the probabilities $P(r|c)$ of the relation labels r given the initial split context c. For binary models, one model per output label is trained and the softmax layer classifies whether the sentence expresses the slot (output 1) or not (output 0). For multiclass models, there is only one model in total and the softmax layer classifies which of the output labels (the slots plus an artificial negative relation) best matches the sentence. For inverse relations, e.g., `per:parents` and `per:children`, there is only one output label in the softmax layer. By reversing the relation arguments in the input sentence and adapting the flag v, examples for `per:parents` can be cast into examples for `per:children` (or vice versa). This avoids redundant training.

The architecture of `contextCNN` with the context splitting at the relation arguments and the weight sharing among the convolutional filters is specifically designed for relation classification. The input flag is added for handling inverse relations. Figure 3.8 depicts the structure of the CNN.

3.3.5 Combination

Finally, we combine the results of pattern matching, support vector machines and `contextCNN`. In particular, we compute the linear combination of the scores of the different models:

$$\text{score}_{\text{cmb}} = \sum_{m \in M} \alpha_m \cdot \text{score}_m \tag{3.1}$$

with $M = \{\texttt{PAT}, \texttt{SVM}, \texttt{CNN}\}$ and α_m denoting a weight tuned on previous slot filling evaluation data. For the pattern matching module `PAT`, we create scores by mapping the result (match or no-match) to 1 or 0. For the support vector machine `SVM`, we apply the sigmoid function as described in Section 3.3.3. For the convolutional neural network `CNN`, we directly take the softmax probability of the positive class as the score.

3.4 Datasets

The datasets we use for training, optimizing and evaluating our slot filler classification models are automatically labeled on the sentence level. The reason is to reflect the scenario the models will be exposed to in the slot filling pipeline: The proof for the slot filler which needs to be provided by the pipeline is a single sentence, thus, the models need to score each sentence with respect to how likely it expresses the relation from the query.

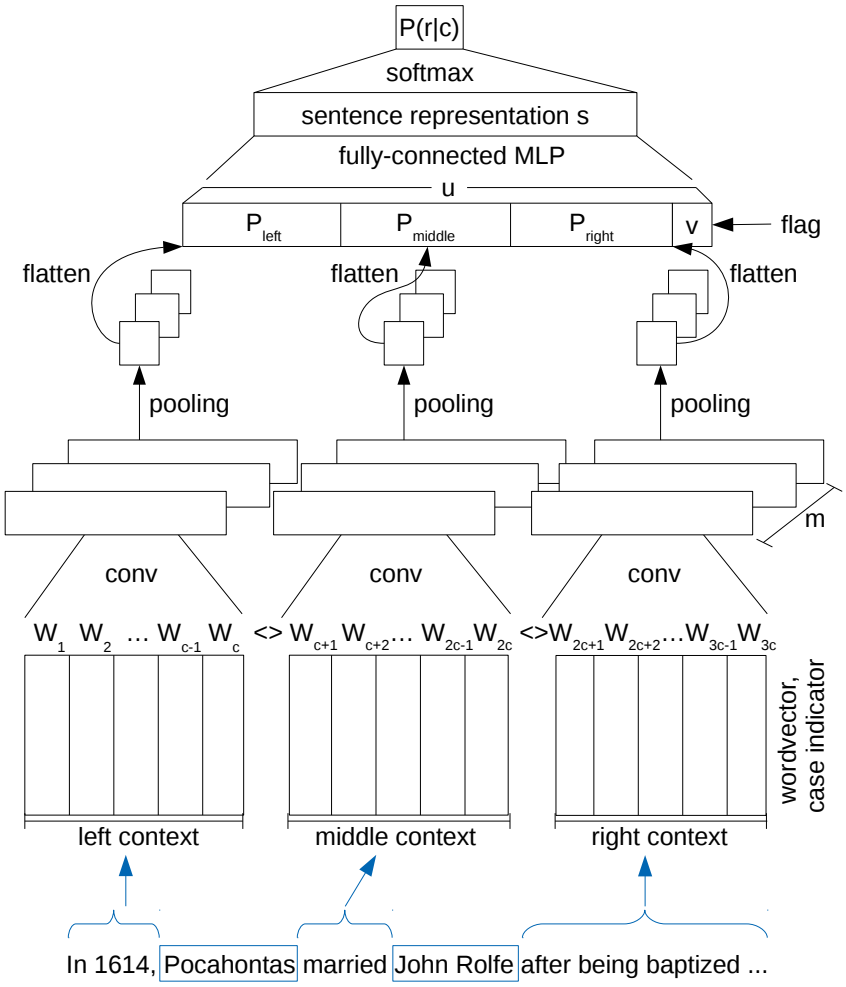


Figure 3.8: `ContextCNN`: convolutional neural network for slot filling.

	source corpus	evaluation corpus
news documents	1,000,257	8,938
web documents	999,999	-
discussion forum documents	99,063	40,186

Table 3.2: Statistics of TAC source and evaluation corpora (2015).

3.4.1 Official Slot Filling Source and Evaluation Corpus

TAC and LDC (Linguistic Data Consortium)[6] provide the participants of the slot filling shared task with two corpora: A source corpus which was also used as the evaluation corpus from 2009 until 2014, as well as an additional evaluation corpus for each year since 2015. Table 3.2 shows statistics for the English source corpus (LDC2013E45) and the English evaluation corpus 2015 (LDC2015E77).

The news documents have been selected from English Gigaword Fifth Edition.[7] The web documents are from GALE web collections.[8] The discussion forum documents are taken from the BOLT Phase 1 discussion forum source data.[9]

The domain mismatch between source and evaluation corpus is obvious from Table 3.2 and justifies our genre-dependent document pre-processing.

3.4.2 Training Data

Creation

TAC provides the source and evaluation corpus but only very few annotated examples which can be used for training. Therefore, most participants create their own corpora, mainly using distant supervision (see Section 2.1.2). We follow them and extract a large set of training examples with distant supervision over Freebase relation instances from the following corpora:

- TAC source corpus[10]
- NYT corpus[11]
- Subset of ClueWeb[12]
- Wikipedia[13]
- Freebase description fields (Bollacker et al., 2008)

[6] `https://www.ldc.upenn.edu`.
[7] LDC2011T07.
[8] LDC2009E93, LDC2009E14, LDC2008E54, LDC2008E41, LDC2007E102.
[9] LDC2012E04, LDC2012E16, LDC2012E21, LDC2012E54.
[10] LDC2013E45.
[11] LDC2008T19.
[12] `http://lemurproject.org/clueweb12`.
[13] Wikipedia dump from May 2014.

For creating negative examples for the different relations, we extract sentences with entity pairs with the correct named entity tags for the given slot but without the given relation according to Freebase. The resulting negative examples are similar to the negative examples our system will be given as input during evaluation since the filler candidates are also extracted based on their named entity tags. However, due to the incompleteness of Freebase it is not sure that a relation does not exist between two entities if it is not stored in Freebase (cf., Section 2.1.2). Therefore, we clean the negative examples with short patterns, such as "born in" for the relation `per:location_of_birth`: If a pattern of the given relation appears in the sentence, we do not include it into the set of negative examples.

Selection by Self-Training

With distant supervision and a large collection of text data, it is possible to extract a large number of training instances. However, due to the distant supervision assumption, the labels contain noise (cf., Section 2.1.2). To reduce the number of wrong labels, we perform an automatic training data selection process, similar to self-training strategies, which are used for bootstrapping additional training examples (Mihalcea, 2004; Rosenberg et al., 2005; McClosky et al., 2006a,b; Angeli et al., 2015).[14] The general process is depicted in Figure 3.9. Algorithm 1 shows the selection procedure in more detail. First, the extracted training samples are divided into k batches B. While in theory, each extracted sample could be processed individually, this is not efficiently feasible in practice given the large amount of instances which are extracted by distant supervision. Then, we train one SVM per slot on the annotated slot filling dataset released by Stanford (Angeli et al., 2014b). This dataset is not very large but it has been labeled using crowdsourcing. Thus, we expect the labels to be correct in most cases. As a result, the initial classifiers are trained on presumably clean data and should, therefore, be able to help in the process of selecting additional data. For each batch of training samples, we use the classifiers to predict labels for the samples e and select those samples for which the distantly supervised label corresponds to the predicted label with a high confidence of the classifier. The confidence thresholds are chosen heuristically: For positive examples, the confidence should be high to create clean examples (see line 5 and 17 of Algorithm 1) while for negative examples it should not get too low to include not only easy examples (line 6 and 15). Those selected samples are, then, divided into ten chunks (line 19) and successively added to the training data. The goal is to add as many examples as possible without decreasing the performance of the classifier on the development set too much (line 22). The resulting training set T is then used to train the SVMs and CNNs of the slot filler classification component.

Note that we do not train models on the dataset before selection by self-training. The reason is that for many relations the large number of extracted instances does not allow an efficient training of neural network models. Thus, we cannot assess the impact of self-training on the final performance of the model. Instead, we manually assess random

[14] In difference to co-training (Blum and Mitchell, 1998), which could be used for this setup as well, we have decided to train only one classifier since our initial set of labeled examples is not large. Nigam and Ghani (2000) refer to self-training as a hybrid of co-training and expectation maximization (EM).

examples which are deleted by self-training and find that the number of false positive labels is reduced considerably. Table A.3 in the appendix provides statistics of the training dataset after selection by self-training.

Algorithm 1 Selection by self-training.

```
1: procedure SELECTION
2:     T ← Stanford SF corpus
3:     B ← batches of extracted training examples
4:     classifier ← support vector machine
5:     θ+ ← (0.6, 1.0]
6:     θ− ← (0.25, 0.3)
7:     while |B| > 0 do
8:         B_next ← B.pop()
9:         classifier.train(T)
10:        T_next ← []
11:        for each e ∈ B_next do
12:            distant_label ← e.label
13:            predicted_label, conf ← classifier.predict(e)
14:            if distant_label = predicted_label then
15:                if distant_label = − and conf ∈ θ− then
16:                    T_next ← T_next ∪ {e}
17:                else if distant_label = + and conf ∈ θ+ then
18:                    T_next ← T_next ∪ {e}
19:        T_chunks ← split(T_next)
20:        for each chunk ∈ T_chunks do
21:            classifier_next.train(T ∪ chunk)
22:            if classifier_next.F1 >= 0.95 · classifier.F1 then
23:                T ← T ∪ chunk
24:    return T
```

3.4.3 Development Data

The resulting training set is less noisy than before the selection process but still contains wrong labels. For optimizing the hyperparameters of our models, it would be beneficial to use a dataset which has correct labels and is as similar as possible to the examples the models will be exposed to during the slot filling evaluation. Therefore, we leverage the existing manually labeled system outputs from the previous slot filling evaluations: We extract the sentences from the system outputs and automatically determine the position of the query entity and the filler. Then, we label each sentence as correct or wrong according to the manual assessment. Due to differences in the offset calculation of some systems, we cannot use all available data. However, the resulting dataset still has a reasonable number of examples with presumably clean labels. Table A.4 in the appendix provides statistics. We assign the manual assessments from 2012 and 2013 to a development set and the assessments from 2014 to a test set. In Section 3.5.2 and Section 3.5.3, we report classification results on those two sets. For tuning our models for the official evaluation from 2015, however, we use both sets in order to exploit all available resources. In the

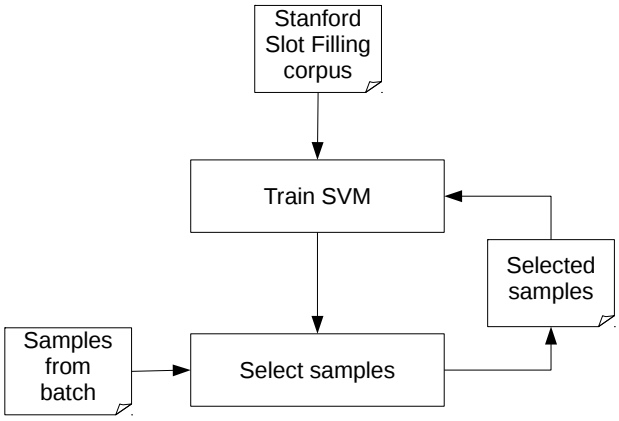


Figure 3.9: Training data selection process.

	dev	test
news	87.5%	73.4%
web+forum	12.5%	26.6%

Table 3.3: Genre (domain) distribution in the slot filling benchmark dataset.

following sections, we will refer to this dataset as "slot filling benchmark dataset". In order to assess its quality and usefulness for tuning slot filler classification models, we compute the correlation between results on the benchmark dataset and results of the whole slot filling pipeline. The Pearson's correlation coefficient when using the data from the slot filling evaluations 2013 is 0.89, the correlation coefficient when using 2014 slot filling data is 0.82. Because of these positive correlations, this dataset will be beneficial for everyone working on slot filling. It provides possibilities for tuning models outside of the slot filling pipeline and comparing the quality of slot filler classification components independent of other pipeline components. This has not been possible so far given only the manual assessments of slot filling pipeline outputs. Therefore, this benchmark dataset opens the possibility to assess and improve the quality of slot filler classification components more effectively and more efficiently. We publish the scripts to reproduce the dataset at `http://cistern.cis.lmu.de/SFbenchmark`.

When looking at the genres (domains) in the development and test dataset (see Table 3.3), it can be seen that the distribution of genres is quite different. Therefore, we also provide genre-specific splits of the dataset. The statistics can be found in Table A.5 in the appendix.

3.4.4 Data for Multiclass Models

The training and development datasets described above are designed for binary models: For each relation, they contain positive and negative examples specific to this relation. For the multiclass setting, a set of negative instances, which do not belong to any of the predefined relations, is needed. However, a negative instance for the slot `per:date_of_birth`

category	number
number of chains (total)	53,929,525
chains per document: min	0
chains per document: max	2061
chains per document: avg	26.18
chains per document: median	15
number of mentions (total)	197,566,321
mentions per chain: min	1
mentions per chain: max	3428
mentions per chain: avg	3.66
mentions per chain: median	2
words per mention: min	1
words per mention: max	900
words per mention: avg	3.05
words per mention: median	2
pronoun mentions	51,139,283
singletons	13,189
chains with identical mentions	16,016,935

Table 3.4: Statistics of coreference resource.

is not automatically a negative instance for the slot `per:date_of_death`. Therefore, the negative examples for each slot are postprocessed: We filter them with the same pattern lists we used for the binary classification data: A negative instance that includes a trigger for any of the positive slots is deleted from the set. The remaining negative instances are labeled with an artificial class `N`. This process might filter too many examples (for example if a sentence contains both the place of birth and the place of death of a person). However, the resulting set of negative examples is still reasonably large for training the classifier. Note that a simple intersection of all the negative data for binary classification would lead to less negative instances in total since the entities do not necessarily overlap across relations.

Further note that we only modify the training set and still use the original dev and test sets for our experiments in order to compare the multiclass models with the binary models. For assessing the performance of the multiclass models on the binary dev and test sets, we take the maximum prediction and map all relations except for the one from the given binary data to the negative class.

3.4.5 Coreference Resource

For a more efficient processing of the slot filling source corpus, we have pre-processed all the documents by computing their coreference chains using Stanford `CoreNLP` (Manning et al., 2014). Since this can be an important resource for researchers working on the slot filling task (see Section 3.1.3 and Section 3.6.4), we make it available to other participants at `http://cistern.cis.lmu.de/corefresources`. Although `CoreNLP` is publicly available, our resource can save researchers much time and resources, given the large size of the slot filling source corpus. Table 3.4 lists statistics about the extracted coreference chains and

their mentions. In addition to the minimum, maximum, average and median numbers of chains per document, mentions per chain and words per mention, we also report the number of mentions which are pronouns, the number of singletons (chains consisting of only one mention) and the number of chains with only identical mentions.

3.5 Results

In the following subsections, we present the results of our binary and multiclass slot filler classification models using the slot filling benchmark dataset described in Section 3.4.3 as well as the overall performance of our system in the official 2015 slot filling evaluations. The hyperparameters of all models are tuned on the development part of the slot filling benchmark dataset using grid search. The resulting values as well as the applied regularization techniques are provided in the appendix in Section A.5.

3.5.1 Evaluation Measures

For the classification results, we calculate per-slot F1 scores as well as an average F1 score over all slots ("macro F1"). For the models we use in our slot filling system, we also report the "micro F1" score, which is an average over all individual decisions, since the whole slot filling pipeline is also evaluated with micro F1. As a result, the micro F1 score is biased towards classes with more instances while the macro F1 score treats all classes equally and can, thus, also indicate the performance of the model on classes with few instances (Sokolova and Lapalme, 2009).

The slot filling system scores are calculated using the official shared task scoring scripts. We report the official measures "CSLDC max micro", which is the micro precision, recall and F1 score over all queries (thus, "micro"), and "CSLDC max macro", which is the average F1 score over all slots (thus, "macro"). If the hop 0 sub-query occurred several times in the query set (with different hop 1 sub-queries), that answer to the hop 0 sub-query is scored which leads to the maximum results over both hops (thus, "max").[15]

3.5.2 Slot Filler Classification: Binary Results

Model Choices

Table 3.5 shows a comparison of different model choices. First, we compare patterns created with distant supervision (`PATdist`) (Roth et al., 2013) to patterns extracted from phrases found with universal schema (Riedel et al., 2013) (`PATuschema`) (Roth et al., 2014b). The distantly supervised patterns outperform the patterns from universal schema. When we use patterns in the following experiment, we, therefore, use distantly supervised patterns.

Second, we compare support vector machines with and without skip n-gram features as input (`SVMskip` and `SVMbow`, respectively). Skip n-gram features perform better on the

[15]For more details, see `https://tac.nist.gov/2015/KBP/ColdStart/tools/README.md`.

model	dev	test
PATuschema	.33	.33
PATdist	**.35**	**.36**
SVMbow	.59	.44
SVMskip	**.62**	**.48**
PCNN	.52	.39
PCNNext	.55	.42
contextCNN	**.60**	**.46**

Table 3.5: Macro F1 results of different model choices on slot filling benchmark dataset.

development set and can also generalize better to an unseen test set. In the official slot filling evaluations 2015, we used SVMbow in the slot filler classification module. In all other experiments, we use SVMskip because of their superior performance. In Section 3.5.5, we also show the slot filling pipeline results when using SVMskip on the 2015 evaluation data.

Finally, Table 3.5 shows that contextCNN, which splits the sentence into three contexts before convolution, outperforms the state-of-the-art piecewise CNN model (PCNN) (Zeng et al., 2015), even if we extend PCNN with a fully-connected hidden layer and k-max pooling (PCNNext) to make the number of parameters more comparable to our model. Approximating the PCNN architecture to our architecture (PCNNext) improves the results on both dev and test set but does not reach the results of contextCNN. This confirms that both our CNN model architecture and input representation are effective and can outperform a state-of-the-art neural network for relation extraction.

Slot-wise results for the different model choices can be found in the appendix in Table A.6 and Table A.7.

Final Models vs. Baselines

Table 3.6 provides slot-wise results of the three models which perform best on dev (PATdist, SVMskip, contextCNN), their combination (CMB, as described in Section 3.3.5) and two baseline systems: Mintz++ and MIMLRE. Mintz++ is a model based on the Mintz features (lexical and syntactic features for relation extraction) (Mintz et al., 2009). It has been implemented by Surdeanu et al. (2012), who also use it as a baseline model. MIMLRE (Surdeanu et al., 2012) is a graphical model designed to cope with multiple instances and multiple labels in distantly supervised data (see Section 2.1.2). It is trained with Expectation Maximization (Dempster et al., 1977). For both Mintz++ and MIMLRE, we use the implementations by Surdeanu et al. (2012).

Note that all models in Table 3.6 are trained on the same training data, namely the data described in Section 3.4.2 (except for PATdist whose patterns are kept unchanged).

The SVM and CNN models outperform the baseline models as well as the pattern matcher. This shows that our approaches of creating training data for slot filler classification and representing the contexts are effective. The CNN, for example, outperforms patterns on test for about 67% of all slots. Overall, the CNNs perform slightly worse but comparable to the SVMs: The results of the CNNs on test are better than the results of the

	Mintz++		MIMLRE		PATdist		SVMskip		contCNN		CMB	
	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
per:age	.84	.71	.83	.73	.69	**.80**	**.86**	.74	.83	.76	**.86**	.77
per:alternate_names	.29	.03	.29	.03	**.50**	**.50**	.35	.02	.32	.04	**.50**	**.50**
per:cause_of_death	.76	.42	.75	.36	.44	.11	**.82**	.32	.77	**.52**	**.82**	.31
per:children	.76	.43	.77	.48	.10	.07	.81	.68	.82	.61	**.87**	**.76**
per:date_of_birth	**1.0**	.60	.99	.60	.67	.57	**1.0**	.67	**1.0**	**.77**	**1.0**	.67
per:date_of_death	.67	.45	.67	.45	.30	.32	**.79**	**.54**	.72	.48	**.79**	**.54**
per:empl_memb_of	.38	.36	.41	.37	.24	.22	.42	.36	.41	.37	**.47**	**.39**
per:location_of_birth	.56	.22	.56	.22	.30	.30	.59	.27	.59	.23	**.74**	**.36**
per:loc_of_death	.65	.41	.66	**.43**	.13	.00	.64	.34	.63	.28	**.70**	.35
per:loc_of_residence	.14	.11	.15	.18	.10	.03	**.31**	**.33**	.20	.23	**.31**	.31
per:origin	.40	.48	.42	.46	.13	.11	**.65**	**.64**	.43	.39	**.65**	**.59**
per:parents	.64	.59	.68	.65	.27	.38	.65	**.79**	.65	.78	**.72**	.71
per:schools_att	.75	**.78**	.76	.75	.27	.26	.78	.71	.72	.55	**.79**	.71
per:siblings	**.66**	.59	.64	.59	.14	.50	.60	.68	.63	**.70**	.65	**.70**
per:spouse	.58	.23	.59	.27	.40	.53	.67	.32	.67	.30	**.78**	**.57**
per:title	.49	.39	.49	.40	.48	.42	.54	**.48**	.57	.46	**.59**	.46
org:alternate_names	.49	.46	.50	.48	.70	**.71**	.62	.62	.65	.66	**.72**	.67
org:date_founded	.41	.71	.42	**.73**	.47	.40	.57	.70	.64	.71	**.68**	.68
org:founded_by	.60	.62	.70	.65	.39	.62	.77	.74	.80	.68	**.85**	**.77**
org:loc_of_headqu	.13	.19	.14	.20	.39	.30	.43	.42	.43	.45	**.50**	**.46**
org:members	.58	.06	.55	.16	.03	**.29**	.70	.13	.65	.04	**.76**	.13
org:parents	.32	.14	.36	.17	.31	.18	.37	.20	.41	.16	**.52**	**.21**
org:subsidiaries	.32	.43	.35	.35	.32	**.56**	.38	.37	.36	.44	**.42**	.49
org:top_memb_empl	.35	.44	.37	.46	.53	.46	.43	**.55**	.43	.53	**.58**	.51
micro F1	.32	.37	.46	.39	.39	.39	.53	.48	.52	.44	**.62**	**.51**
macro F1	.53	.41	.54	.42	.35	.36	.62	.48	.60	.46	**.68**	**.53**

Table 3.6: F1 results of binary models on slot filling benchmark dataset. `contCNN` is short for `contextCNN`, `CMB` denotes the combination of `PATdist`, `SVMskip` and `contextCNN`.

		train on NEWS				train on WEB			
		SVM		CNN		SVM		CNN	
		dev	test	dev	test	dev	test	dev	test
test on news	per:age	.79	.80	**.88**	**.87**	.78	.76	.85	.83
	per:children	**.85**	**.86**	.78	.78	.75	.80	.00	.07
	per:spouse	.74	.64	.76	**.71**	**.77**	.65	.73	.67
	org:alt_names	.22	.32	**.69**	.67	.65	**.70**	.66	.66
	org:loc_headqu	.51	.50	**.53**	.51	.51	**.53**	**.53**	.50
	org:parents	**.30**	.32	.29	**.34**	.26	.33	**.30**	**.34**
test on web	per:age	.33	.73	**.57**	**.83**	.00	.67	**.57**	**.83**
	per:children	.59	.33	**.70**	.33	.63	**.57**	.00	.00
	per:spouse	.52	.50	.60	.57	.56	.57	**.67**	**.62**
	org:alt_names	.27	.19	.51	.37	**.60**	**.49**	.56	.38
	org:loc_headqu	.39	.46	.43	.44	**.44**	**.48**	.36	.47
	org:parents	.09	**.08**	.11	.07	.10	**.08**	**.15**	**.08**

Table 3.7: Genre specific F1 scores. Genre specific training data (of the same sizes). Top: news results. Bottom: web results.

SVMs for only 42% of all slots. One reason for that might be limited training data for some slots. The SVM is more robust against this than the CNN. More possible explanations for the performance difference between patterns, SVMs and CNNs are given in Section 3.6.3.

The combination of pattern matcher, SVM and CNN yields the best results for most of the slots and overall (macro and micro F1). This confirms our intuition that the different models provide complementary information, which can be combined for a more effective classification.

Domain Effects

As mentioned before (see Section 3.4.1 and Section 3.4.3), the distribution of genres/ domains in the slot filling evaluations has changed over the years. Since most teams train their systems on previous evaluation data, their systems face a domain adaptation challenge during the official evaluations. To quantify the effect of different domains on our models, we split our datasets according to genres (news and web). For a fair comparison, we further adapt the training sets to have the same size by subsampling the larger one (the news dataset for most slots).

Table 3.7 shows the results of the genre-specific models on data from the same genre as well as from the other genre. For a better overview and comparison, we only present numbers for a subset of the slots here. The results for the other slots follow the same trends and are given in the appendix in Table A.13 and Table A.14. In all cases, the results on dev and test set differ much and seem to be not correlated to each other (especially for slot filling on web — bottom part of Table 3.7). Models trained on news (left part) show higher performance in the within-genre evaluation than cross-genre. This is different, however, for models trained on web (right part). Since web data is much noisier, we assume that it is less predictable even for models trained on web. In general, our results confirm that domain differences are an important challenge of slot filling.

		macro F1				micro F1			
		SVM		CNN		SVM		CNN	
		dev	test	dev	test	dev	test	dev	test
binary		**.62**	**.48**	.61	.45	**.54**	**.47**	.52	.43
multi	#N ≈ #non-N	.51	.37	**.52**	**.40**	**.50**	**.43**	.27	.21
multi	#N ≈ 2· #non-N	.46	.32	.49	.38	.47	.37	.24	.18
multi	#N ≈ 4· #non-N	.41	.30	.45	.36	.41	.34	.22	.18

Table 3.8: Macro F1 scores for binary vs. multiclass classification. #N denotes the number of negative instances in the training data, #non-N is the number of non-negative instances, i.e., instances with a relation.

3.5.3 Slot Filler Classification: Multiclass Results

Table 3.8 provides the macro and micro F1 results of multiclass models in comparison to the previously presented binary models. As described in Section 3.4.4, we use the same development and test data for evaluating both model types. Slot-wise results can be found in the appendix in Table A.15. In particular, we investigate the impact of the number of negative instances (with label N) in the training dataset on the multiclass models.

In general, the binary models perform better than the multiclass models. In contrast to the results of the binary models (see Table 3.6), the multiclass CNN outperforms the multiclass SVM in terms of macro F1, especially on test. However, the SVM performs considerably better in terms of micro F1. This shows again that SVM and CNN have complementary strengths and weaknesses. If the micro F1 score is considerably lower than the macro F1 score, this indicates that the model identifies classes with lower frequency better than classes with higher frequency.

A reason for the lower performance of both multiclass models in comparison to the binary models might be the high number of output classes. The models need to recognize and distinguish much more patterns than in the binary case since they are trained to jointly disambiguate all classes. We assume that this is especially challenging for sentences containing several relations.

Note that the aggregated scores for the binary models reported in Table 3.6 and Table 3.8 are slightly different. This is because we use only one label per inverse slot pair in the multiclass models to avoid redundant training, thus, we do not evaluate it for `per:children` and `per:parents` but only for `per:children` (same for `org:parents` and `org:subsidiaries`). For a fair comparison, we also omit those slots for the binary models in Table 3.8.

While it has not been necessary to change the ratio of positive and negative instances for the binary classification models,[16] we find that limiting the number of negative instances (with label N) is beneficial for multiclass classification. However, we only limit them until they have the same number as the positive instances since the number of negative instances

[16]Changing the ratio of positive and negative instances might further improve the results also for the binary case. We have chosen not to do that for the binary classification dataset and to adapt the ratio for the multiclass classification dataset only slightly in order to keep the datasets more realistic.

	setup	micro P	micro R	micro F1	macro F1
hop 0	baseline	19.11	22.32	20.59	29.06
hop 0	+ CNN	31.67	23.97	27.29	32.38
hop 0	+ CNN + EL	**31.71**	**24.13**	**27.41**	**32.67**
hop 1	baseline	5.08	4.11	4.54	5.25
hop 1	+ CNN	10.46	6.33	7.89	7.53
hop 1	+ CNN + EL	**11.82**	**7.00**	**8.79**	**7.78**
all	baseline	14.48	14.76	14.62	20.54
all	+ CNN	23.99	16.65	19.66	23.48
all	+ CNN + EL	**24.63**	**17.02**	**20.13**	**23.76**

Table 3.9: CSLDC max micro/macro scores from 2015 assessments, SVM without skip n-gram features.

(i.e., wrong filler candidates) in the slot filling pipeline is very high as well. Note that again the dev and test sets remain unchanged. In Section 3.5.5, we present the results of the multiclass models when applied in the slot filling pipeline.

3.5.4 Official Assessment Results 2015

In this section, we provide the official results for our system in the 2015 slot filling evaluations. We report results for three runs here:

(i) a baseline run using only patterns and support vector machines without skip n-gram features for slot filler classification;

(ii) a run which combines the baseline classification models with the CNNs presented in Section 3.3.4;

(iii) a run which also includes an entity linking component in the slot filling pipeline.

The baseline run can be seen as a "traditional" slot filling system. The comparison of the baseline run and baseline + CNN shows the impact of `contextCNN` on slot filler classification. Table 3.9 shows that the CNNs can improve the slot filling pipeline results by a large margin, even though they have been trained on noisy (distantly supervised) training data. In general, all performance trends are consistent across hops: Adding CNNs helps to improve the final results and adding entity linking leads to further performance gains. We provide slot-wise evaluation results of the best model (baseline + CNN + EL) in the appendix (Table A.16 and Table A.17).

Comparison to State of the Art

Compared to other slot filling systems, our system achieved rank three in the official 2015 evaluation (Adel and Schütze, 2015). Table 3.10 provides the scores of the top five systems. The performance difference of our system to rank two is quite small, only the top-ranked

rank	team	micro F1
1	Stanford	31.06
2	UGENT	22.38
3	Our (baseline+CNN+EL)	20.13
4	UMass	17.20
5	UWashington	16.44

Table 3.10: CSLDC max micro F1 scores for top five systems in 2015 evaluations.

system performs considerably better. We assume that the main reason is their manually labeled training data (see Section 3.7). Thus, our system is one of the state-of-the-art systems for distantly supervised slot filling.

3.5.5 Additional Slot Filling System Results

In the official evaluation 2015, we have participated with a system using only binary models and no skip n-gram features for the SVMs. In order to evaluate skip n-gram features and multiclass models, we run our system again on the 2015 evaluation data. Since we do not assess the results manually, the comparability to the official results from Table 3.9 is limited: If the system finds a new slot filler which is correct but not provided in the pool of correct assessments, it will be scored as wrong (Chaganty et al., 2017). Nevertheless, this is the usual way of assessing slot filling system results outside of the official shared task evaluations.

Table 3.11 provides the scores of our system with different classification models: SVM with skip n-gram features (in both binary and multiclass variants) and the multiclass version of `contextCNN`. Note that we only report results for comparing pure binary to pure multiclass classification modules. We have also experimented with combinations, e.g., combining binary skip n-gram SVMs with the multiclass CNN, but have not obtained additional performance gains. Similar to the models we used in the official slot filling evaluations 2015 (see Section 3.5.4), we combine the machine-learning models with the pattern module and regard a combination of patterns and SVMs as the traditional system, which we then enrich with the scores from the CNNs.

The results in Table 3.11 again show a discrepancy between micro and macro scores: While the binary models lead to the best macro F1 scores, which is compatible to the slot filler classification macro F1 results in Table 3.8, the multiclass models perform comparable or even better when considering micro precision and F1. This could indicate that the multiclass models have a low performance for some slots which are infrequent in the evaluations. Hence, their impact on the macro F1 score is higher than on the micro F1 score. When comparing the binary and multiclass CNN (without named entity recognition (NER) features) in Table 5.6, there are indeed some slots for which the multiclass models perform significantly worse, such as `per:cause_of_death`, `per:origin` or `org:location_of_headquarters`. Note that until 2014, micro F1 scores have been the only evaluation measure for slot filling (Surdeanu and Ji, 2014).

The comparable performance of multiclass models to binary models confirms their

	SVM	CNN	micro P	micro R	micro F1	macro F1
hop 0	binary skip	–	29.12	26.18	27.57	32.50
hop 0	binary skip	binary	31.79	**28.23**	29.91	**34.20**
hop 0	multi skip	–	25.25	12.07	16.33	18.44
hop 0	multi skip	multi	**34.42**	26.66	**30.04**	32.82
hop 1	binary skip	–	7.36	4.78	5.80	6.13
hop 1	binary skip	binary	9.80	**7.00**	**8.17**	**8.28**
hop 1	multi skip	–	7.11	3.67	4.84	4.34
hop 1	multi skip	multi	**12.59**	3.89	5.94	7.78
all	binary skip	–	21.75	17.30	19.27	23.06
all	binary skip	binary	23.80	**19.42**	21.39	**24.92**
all	multi skip	–	17.38	8.58	11.49	13.39
all	multi skip	multi	**29.60**	17.20	**21.76**	23.86

Table 3.11: CSLDC max micro/macro scores for SVMs with skip n-gram features and multiclass models.

usefulness and indicates their generalizability to unseen data despite their poor performance on the slot filling benchmark dataset (see Table 3.8). It might further indicate that the decision threshold is an important hyperparameter when working with multiclass models. Following the same paradigm as we have used for the binary models, we have not tuned the threshold on the slot filling benchmark dataset but have optimized the output thresholds in the slot filling pipeline (see Section 3.2.1). For multiclass models, output thresholds in the range [0.2, 0.55] lead to better performance than higher thresholds as used for the binary models.

3.6 Analysis

For analyzing the slot filling system, we perform a recall analysis, a manual error analysis and several ablation studies. Furthermore, we analyze `contextCNN` and the weights for combining the different models for slot filler classification.

3.6.1 Recall Analysis

Our first analysis investigates the recall of the different components and is similar to the analysis by Pink et al. (2014). In particular, we evaluate the components of our system before the slot filler classification module. Thus, we measure which recall our system could achieve with a perfect slot filler classification module that does not lose any recall.

Table 3.12 shows the results on the slot filling assessment data from 2015 for hop 0: The information retrieval component is able to achieve a recall of 78.82% with aliases and 74.54% without aliases, respectively. Unexpectedly, the IR alias rather hurts recall on this test set instead of improving it as on previous TAC slot filling evaluations (from 2013 and 2014). A possible reason could be misleading aliases, which introduce false

component	recall	Δ
`Terrier` IR (300 documents)	83.21%	-16.79%
- IR alias	83.58%	-16.42%
fuzzy string match	78.82%	-4.39%
- alias	74.54%	-8.67%
entity linking	76.82%	-2.00%
top 100 documents	71.96%	-4.86%
sentence extraction	65.01%	-6.95%
- coreference	62.77%	-9.19%
- alias	58.64%	-13.32%
candidate extraction	59.64%	-5.37%
- coreference	56.23%	-6.54%
- alias	53.69%	-4.95%

Table 3.12: Analysis of recall after the application of the different pipeline components.

positive retrievals and change the ranking of the correct retrievals. Similar, the entity linking component hurts recall a bit. However, it also increases precision and leads to better overall results (cf., Table 3.9). Evaluating only the top 100 documents instead of all extracted documents from `Terrier` (maximum 300), leads to a recall loss of almost 5%. Thus, allowing the slot filling system a longer run time for processing all extracted documents could lead to a higher final recall (but potentially also to more false positive extractions and, thus, a lower precision). As mentioned before, choosing only the 100 most relevant documents has led to the best time-performance trade-off on data from previous evaluations (2013 and 2014). The sentence extraction component extracts the relevant sentences quite successfully with an additional recall loss of only 6.95%. Evaluating this component in more detail shows the importance of coreference resolution and aliases: The recall loss without coreference resolution is almost 10%, the recall loss without aliases is more than 13%. Finally, the candidate extraction component is able to extract most of the relevant candidates, yielding an overall recall of 59.64% before slot filler classification. Without coreference resolution for sentence extraction, the overall recall is 56.23%, without alias information for sentence extraction, the overall recall is 53.69%. Assuming a perfect slot filler classification component with $P = 100\%$ and $R = 100\%$, the maximum F1 score of the whole slot filling system would be 74.72%. This number is more than twice as high as the performance of the best slot filling system 2015 (Angeli et al., 2015) but still low compared to other NLP tasks. This illustrates the difficulties of the slot filling task and the importance of all individual components of the pipeline since especially recall losses cannot be recovered by subsequent components.

3.6.2 Error Analysis

We manually analyze 120 false positive predictions of our system, randomly picked from the output for the official 2015 evaluation. Table 3.13 shows which pipeline component is responsible for how many errors. The numbers do not sum to 100% since for 7% of the

error category	ratio
alias component	9%
entity linking component	2%
candidate extraction component	21%
classification component	61%

Table 3.13: Error analysis of the pipeline.

cases, we could not unambiguously identify a single component as the error source. This analysis is complementary to the one before (in Section 3.6) since we analyze the wrong predictions of our system, thus, we analyze precision loss rather than recall loss.

The alias component especially struggles with acronyms which can refer to several entities. In the candidate extraction component, most errors (16% of 21%) are due to wrong named entity recognition despite using a state-of-the-art NER tool (Manning et al., 2014). This is in line with the analysis by Ji et al. (2011) and Min and Grishman (2012), who have identified named entity recognition as one of the key sources of error. For some instances (4% of 21%), the sentence splitting of the document is incorrect. In the remaining cases (1% of 21%), coreference resolution fails. The classification component has to cope with very challenging input data since most extracted filler candidates are false positives. Thus, it has to establish precision while keeping as much recall as possible. Based on a manual inspection of errors, the most important challenge for the classification component is long contexts which mention multiple relations between several entity pairs.

3.6.3 Analysis of Slot Filler Classification

Contributions of the Different Models

For `CMB` (see Section 3.5.2), we compute the linear interpolation of the scores of `PATdist`, `SVMskip` and `contextCNN`. The interpolation weights are optimized with grid search on the development set. Figure 3.10 shows the distribution of weights for the three different models. All three models contribute to `CMB` for most of the slots. The `contextCNN`, for instance, is included in the combination for 14 of 24 slots. The results of the slot filling pipeline system with and without the `contextCNN` show that this effect generalizes to another test dataset as well as to a pipeline evaluation scenario (see Table 3.9 and Table 3.11).

Analysis of CNN Pooling

To investigate which n-grams the CNN considers to be most important, we extract which n-grams are selected by its five most representative filters. To rank the filters according to their influence on the final classification score, we compute the correlation of the activations of each filter with the final score of the positive class. Then, we take the five filters with the highest correlation and extract to which n-grams they give the highest values. Figure 3.11 shows the result for an example sentence expressing the relation `org:parents`. The height of each bar corresponds to the number of times the 3-gram around the corresponding word

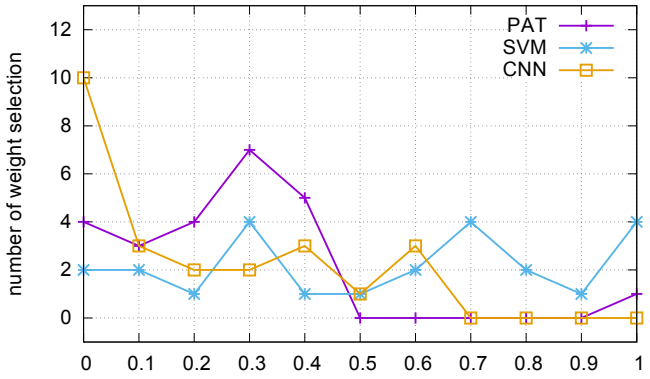


Figure 3.10: Distribution of combination weights.

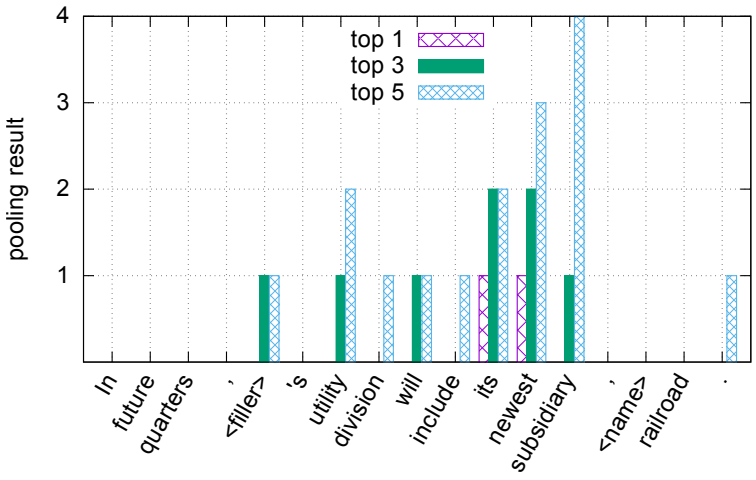


Figure 3.11: Analysis of CNN pooling.

is selected by 3-max pooling. The bar above "newest", for example, shows the result for the trigram "its newest subsidiary". As shown by the Figure, the convolutional filters are able to assign high weights to phrases that trigger a relation, e.g., "its subsidiary". An advantage of CNNs over patterns and SVMs is that they do not rely on exact matches. First, they are able to cope with insertions, similar to the skip n-gram features we use in the SVMs. An example is "newest" which is not important for recognizing the relation. Second, they use embeddings to represent the inputs. Since similar words have similar embeddings, they are able to recognize synonyms and phrases that are similar but not exactly the same as the phrases seen during training. For patterns and SVMs, this type of generalization is more difficult.

entity linking	P	R	F1	ΔF1
+	31.71	24.13	27.41	
–	31.67	23.97	27.29	-0.12

Table 3.14: Impact of entity linking on hop 0 performance.

coreference	P	R	F1	ΔF1
+	31.67	23.97	27.29	
–	19.33	22.40	20.75	-6.54

Table 3.15: Impact of coreference resolution on hop 0 performance.

3.6.4 Ablation Studies

In the following paragraphs, we show the impact of different system components (entity linking, coreference, `contextCNN`) on the final slot filling performance with ablation studies on the hop 0 results.

Impact of Entity Linking

Table 3.14 compares the slot filling system performance with (+) and without (–) the entity linking component. The system performance is slightly reduced when omitting entity linking. However, the difference of the F1 scores is rather small. This shows that the main challenges of the system lie in other components and ambiguous names play a rather small role for the final results of the system.

Impact of Coreference

Table 3.15 shows the results of the overall system with (+) and without (–) coreference resolution in the candidate extraction component. The number of true positives is reduced considerably (from 361 to 321) when the system does not use coreference information. The number of false positives is also lower, but the final results show that the impact of the number of true positives is larger: The F1 score drops by more than six points when omitting coreference resolution.

The impact of coreference resolution for slot filling is also highlighted in (Surdeanu and Ji, 2014): The authors identify coreference resolution as the most important error source in the overall slot filling results and even vote for a specific coreference evaluation in the context of KBP and the development of additional resources. In this context, the coreference resource we make available for the KBP community (see Section 3.4.5) is especially useful for teams who participate in the TAC KBP shared tasks for the first time since it saves them a lot of computational efforts.

Impact of Neural Networks

Table 3.16 provides the results for the slot filling system with (+) and without (–) CNNs in the slot filler classification component when using SVMs with or without skip n-gram

SVM	CNN	P	R	F1	ΔF1
no skip	+	31.67	23.97	27.29	
no skip	–	19.11	22.32	20.59	-6.70
skip	+	33.87	28.39	30.89	
skip	–	31.39	26.34	28.64	-2.25

Table 3.16: Impact of neural networks on hop 0 performance.

features. Adding CNNs to the slot filler classification component improves performance in both cases. When using no skip n-gram features in the SVM models, the performance difference with and without CNNs is considerably higher than when using skip n-gram features. This confirms the strength of skip n-gram features compared to simple bag-of-word n-grams. Similar to CNNs, they are more robust in the case of word insertions. Since training methods for word embeddings assign similar vectors to similar words (see Section 2.2.3), we expect the CNN to detect phrases which are similar to the ones seen during training. If, for example, the CNN has learned that the context "is based in" indicates the relation `location_of_headquarters`, it can also identify "is located in" as a trigger for this relation if the embeddings for "based" and "located" are similar. The SVMs, on the other hand, rely on bag-of-word vectors and, thus, do not know that those two phrases have a similar meaning, especially if the phrase "is located in" does not occur in the training set. Therefore, the CNNs improve the overall system performance, even when using the SVMs with skip n-gram features. In particular, they improve both precision and recall.

3.7 Related Work

3.7.1 Slot Filling

The slot filling shared task has been organized since 2009. There are about 20 teams participating each year (Surdeanu and Ji, 2014). Most systems apply a modular pipeline structure and combine multiple approaches, such as machine learning with distant supervision and manual or automatic patterns (Ji and Grishman, 2011; Surdeanu and Ji, 2014), as we also do in this thesis. In 2015, we were one of the first teams using neural networks (Adel and Schütze, 2015; Angeli et al., 2015; Roth et al., 2015).

In this subsection, different approaches for realizing the slot filling pipeline are described, followed by a more detailed description of two systems that are most relevant to our work: the top-ranked system from 2013 (Roth et al., 2013) since we use their distantly supervised patterns and similar features in our support vector machines; and the winning system from 2015 (Angeli et al., 2015) since we evaluate our system on the assessment data from 2015. When describing different slot filling pipelines, we use participant identifiers for a better overview. A mapping from identifiers to system description papers is given in Table 3.17.

With only a few exceptions, slot filling systems consist of an information retrieval–

team id	reference
ARPANI	(Rawal et al., 2013)
AUSTIN	(Bentor et al., 2013, 2014)
BIT	(Xu et al., 2013a)
CIS	(Adel and Schütze, 2014, 2015)
CMU	(Kisiel et al., 2013, 2014, 2015; Yang et al., 2016a)
COMP	(Xu et al., 2012)
DCD	(Zhang et al., 2015a, 2016a)
GDUFS	(Qiu et al., 2012)
GEOL	(Amaral, 2014)
IBCN	(Feys et al., 2014; Sterckx et al., 2015)
INDIANA	(Wazalwar et al., 2014; Viswanathan et al., 2015)
IRT	(Rahman et al., 2016)
LSV	(Roth et al., 2012, 2013)
NAIST	(Sudo et al., 2016)
NYU	(Min et al., 2012; Nguyen et al., 2014)
OpenKN	(Lin et al., 2014)
PAPELO	(Malon et al., 2012)
PRIS	(Li et al., 2012, 2013; Zhang et al., 2014; Qin et al., 2015; Bao et al., 2016)
RPI	(Hong et al., 2014; Yu et al., 2016)
SAIL	(Lim et al., 2016)
SOOCHOW	(Hong et al., 2016)
STANFORD	(Angeli et al., 2013, 2014a, 2015; Zhang et al., 2016c)
SWEAT	(Liu and Zhao, 2012)
SYDNEY	(Pink and Curran, 2014)
TALP	(Gonzàlez et al., 2012; Ageno et al., 2013)
THU	(Zhang et al., 2015b)
UCD	(Byrne and Dunnion, 2012; Byrne et al., 2013, 2014)
UMASS	(Singh et al., 2013b; Roth et al., 2014b, 2015; Chang et al., 2016)
UNED	(Garrido et al., 2013)
USC	(Chalupsky, 2013, 2014)
UW	(Soderland et al., 2013, 2015)

Table 3.17: Mapping from participant identifier to system description papers. Note that this table does not include all participants and all years of participation but only those systems mentioned in this section.

based pipeline of different modules. Exceptions are UNED2013 and UW2013 who retrieve sentences from an entity mention index and entity-linked corpus, respectively. Other exceptions are STANFORD2014–2015 as well as the system of INDIANA2014 which rely on relational databases consisting of one table storing all sentences from the corpus and another table storing all entity mentions. STANFORD2014 and INDIANA2014 use the data management system "Deep Dive" for this.[17] All the other slot filling systems apply information retrieval systems on the evaluation corpus either using only the query entity name, or expanding the query with aliases. Table 3.18 provides information about which systems apply which design choices.

Our system is also based on information retrieval and uses query expansion to cover alternate name and spelling variations. For query expansion, the shared task participants use redirect information from Wikipedia as we also do, Wikipedia link anchor texts, aliases obtained from Freebase or mentions of entities in an entity linked corpus, such as given by the FACC ClueWeb annotations. For document retrieval, most participants apply `Lucene` (Hatcher and Gospodnetic, 2004), except for us and, e.g., UCD2012–2014 who use `Terrier` (Ounis et al., 2006).

For sentence extraction, only a subset of systems use coreference information. In all our experiments and analysis, coreference information improves the final results though. Similar to us, STANFORD2016 add hyphenated location expressions explicitly to the set of coreferent mentions. Even fewer systems apply entity linking or another form of disambiguating different entities with the same name. Also, our results with entity linking are mixed: Although it slightly improves the final pipeline results, it leads to recall losses due to wrong links. Candidate extraction based on named-entity tags, as we do it, is also implemented by a variety of other systems (see Table 3.18).

Especially in 2012 and 2013, many systems relied only on pattern matching for identifying slot fillers. Now, more and more teams use machine learning models for slot filler classification, such as Naive Bayes, logistic regression classifiers, conditional random fields or support vector machines. `Mintz` and `MIML`, our baseline systems in Section 3.5.2, are used by a few systems as well. More recently, participants also train neural networks, such as bidirectional GRUs, bidirectional LSTMs or CNNs as we do. PAPELO2012 already trained a CNN for slot filling in 2012 but did not submit it to the official evaluation. Other approaches, which are more rarely used, are open information extraction systems, universal schema, `MultiR`, community graph–based relation validation (IRT2016), PageRank (RPI2016), Bayesian Logic Programs (AUSTIN2013–2014) or Markov Logic Networks (INDIANA2014–2015).

Most of the teams applying machine learning models use a combination of different models, mostly also integrating patterns. We follow this line of work since combining different models considerably improves the final results. With the exception of IBCN2014–2015, NYU2014, STANFORD2014–2016 and CMU2016, who use human labels or manual cleaning of noisy labels, e.g., in connection with active learning, the machine-learning models are trained with distant supervision. UMASS2016 clean noisy labels by querying the

[17] `http://deepdive.stanford.edu`.

design choice	slot filling system
no IR	UNED2013, UW2013, INDIANA2014, STANFORD2014–2015
IR with query entity name only	COMP2012, PAPELO2012, CMU2013–2016, SYDNEY2014, SOOCHOW2016
IR with aliases	GDUFS2012, LSV2012–2013, NYU2012/2014, PRIS2012–2016, SWEAT2012, TALP2012–2013, AUSTIN 2013, BIT2013, STANFORD2013, UMASS2013/2016, USC2013–2014, IBCN2014–2015, OpenKN2014, RPI2014, DCD2015–2016, THU2015, NAIST2016
aliases from Wiki redirects	GDUFS2012, NYU2012, IBCN2014–2015, USC2014, UMASS2016
aliases from Wiki anchor texts	LSV2012–2013
aliases from Freebase	DCD2015–2016
aliases from FACC ClueWeb	CMU2013–2016
entity linking	TALP2012, STANFORD2015
coreference resolution	PAPELO2012, PRIS2012–2015, CMU2013–2016, STANFORD2013/2015–2016, UMASS2013, UNED2013, USC2013–2014, UW2013, GEOL2014, NYU2014, SYDNEY2014, DCD2015–2016, IBCN2015, RPI2016
candidate extraction with NER	LSV2012–2013, OpenKN2014, SYDNEY2014, UCD2014, IBCN2015, PRIS2015–2016, IRT2016, NAIST2016, RPI2016, SAIL2016, SOOCHOW2016
pattern matching	GDUFS2012, PRIS2012/2014, SWEAT2012, TALP2012/13, UCD 2012–2014, ARPANI2013, USC2013, GEOL2014, SYDNEY2014, SOOCHOW2016
Naive Bayes	UCD2014
logistic regression	PAPELO2012, IBCN2014–2015, NYU2014, USC2014, STANFORD 2015–2016, THU2015, CMU2016
conditional random fields	CMU2013–2015, OpenKN2014, PRIS2014
support vector machines	COMP2012, LSV2012–2013, BIT2013, UNED2013, IBCN2014, UMASS2015, STANFORD2016
Mintz	STANFORD2014/16, IRT2016
MIMLRE	STANFORD2013–2014, NYU2014, DCD2015, SAIL2016
bidirectional GRU	PRIS2016
bidirectional LSTM	PRIS2015, STANFORD2015–2016, CMU2016, RPI2016, SAIL2016, UMASS 2016
CNN	IBCN2014, UMASS2015, DCD2016, NAIST2016, PRIS2016, STANFORD2016
open IE systems	UW2013/2015, CMU2014, STANFORD2015
universal schema	UMASS2013–2016
MultiR	UW2015, IRT2016
combination of models	COMP2012, LSV2012–2013, NYU2012/2014, BIT2013, STANFORD2013–2016, UCD2013, UNED2013, CMU2014–2015, IBCN 2014–2015, OpenKN2014, USC2014, THU2015, UMASS2015, NAIST2016, SAIL2016
binary models	LSV2012–2013, BIT2013, UCD2013, UNED2013, IBCN2014–2015, USC2014, THU2015, UMASS2015, PRIS2016
multiclass models	CMU2013–2016, STANFORD2013–2015, IBCN2014, DCD2015, UMASS2015

Table 3.18: Design choices of different slot filling systems.

web for relation patterns and argument candidates. In contrast, we approach the problem of noisy labels by an automatic self-training procedure. While some of the participants use binary models, others train multiclass models. We use binary models for our entry in the slot filling evaluation 2015 but compare them with multiclass models in Section 3.5.3 and Section 5.4 of this thesis.

Similar to us, USC2014 merge the three different location relations into one place-of-X relation for classification and disambiguate the extracted location afterwards. CMU2013–2015 validate fillers by querying the web as a post-processing step.

The top-ranked system in 2013, LSV2013 by Roth et al. (2013), follows the main trends in slot filling and applies a modular system based on distant supervision, which is called `RelationFactory` (Roth et al., 2014a). Its pipeline is similar to ours except that it uses neither entity linking nor coreference resolution. It is also applied by other groups, e.g., AUSTIN2014, UMASS2015–2016. LSV2013 achieves the best results by combining patterns and support vector machines. Therefore, we follow them and use their distantly supervised patterns and add skip n-gram features to the feature set of our support vector machines. An important difference to their work, however, is that we also integrate neural networks and train not only binary models but also multiclass models. For training their models, they apply aggregate training and global parameter optimization. In contrast, we use our slot filling benchmark dataset for tuning our models and show that results on this dataset are correlated with the slot filling pipeline results.

The top-performing system in 2015, STANFORD2015 by Angeli et al. (2015), uses manually labeled training data (Angeli et al., 2014b) as well as a bootstrapped self-training strategy in order to avoid distant supervision. The idea behind their self-training is similar to the way we clean the distantly supervised labels. However, they do not use it to clean labels from distant supervision but rather to infer labels on an unlabeled corpus. In contrast to most other slot filling systems, they do not apply an information retrieval–based system but store pre-processed versions of all sentences and entity mentions from the source corpus in a relational database, which they access during evaluation. As relation extractors, they apply a combination of patterns, an open information extraction system, logistic regression, a bidirectional long short-term memory network and special extractors for website and alternate-names slots. In contrast to their system, we apply a traditional slot filling pipeline based on information retrieval and train convolutional neural networks. In 2016, the successor system STANFORD2016 applies a combination of CNNs and LSTMs and finds that this model performs better than a model only based on LSTMs. They hypothesize that the phrasal information which a CNN is able to capture is highly relevant for slot filling relation extraction, which is in line with our findings. In CIS2015 (Adel and Schütze, 2015), we also combine convolutional and recurrent neural networks and find that adding recurrent neural networks increases the performance only little while it increases both training and testing time considerably. We assume that the main reason for STANFORD's better performance in 2015 is less noise in the labels of their training data.

In 2016, the slot filling task was changed to a multilingual challenge: Teams can now build systems for English, Spanish and Chinese or a cross-lingual combination of those languages. However, only a few teams participated in more than one language in 2016,

namely RPI2016, STANFORD2016 and UMASS2016.

The following paragraph summarizes the most recent developments to date in slot filling research: Yu and Ji (2016) present a method based on trigger extraction from dependency trees which does not require (distantly) supervised labels and can work for any language as long as named entity recognition, part-of-speech tagging, dependency parsing and trigger gazetteers are available. Huang et al. (2017) follow our work in extracting training and development data and in using convolutional neural networks for slot filler classification. In contrast to our work, they input dependency paths into the network and apply attention in order to account for the larger middle contexts in slot filler classification. Zhang et al. (2017c) propose position-aware attention, which calculates attention weights based on the current hidden state of their LSTM, the output state of the LSTM and the position embeddings, which encode the distance of the current word to the two relation arguments. Moreover, they publish a supervised relation extraction dataset, obtained by crowdsourcing, for training slot filler classification models. Chaganty et al. (2017) address the issue of evaluating new slot filling systems outside of the official shared task evaluations. They build an evaluation method based on importance sampling and crowdsourcing and make it publicly available.

3.7.2 Neural Models for Relation Extraction

In this section, we describe neural models applied to standard relation extraction (not in the context of slot filling). In particular, we focus on methods relying only on text and do not report work about combining relation extraction from text with inference in knowledge bases since this is beyond the scope of this thesis. For related work on type-aware relation extraction models, see Section 5.5.

Datasets

A popular benchmark for traditional relation classification is the SemEval2010-task 8 dataset (Hendrickx et al., 2010). It contains manually labeled sentences with nine semantic relations between pairs of nominals: `Cause-Effect`, `Instrument-Agency`, `Product-Producer`, `Content-Container`, `Entity-Origin`, `Entity-Destination`, `Component-Whole`, `Communication-Topic` and `Member-Collection`. The relations are directed and there is also an artificial negative relation label, which makes it a 19-class classification task. In contrast to named-entity relation extraction, the entity pairs are general noun phrases. As a result, the relations are also semantically different. It is used by many recent studies, such as (Socher et al., 2012; Hashimoto et al., 2013; dos Santos et al., 2015; Ebrahimi and Dou, 2015; Gormley et al., 2015; Hashimoto et al., 2015; Liu et al., 2015; Nguyen and Grishman, 2015; Xu et al., 2015a,b; Zhang and Wang, 2015; Cai et al., 2016; Nguyen and Grishman, 2016; Vu et al., 2016; Wang et al., 2016a; Yang et al., 2016b; Zhou et al., 2016).

A manually labeled dataset which includes relations between named entities is ACE 2005 (Automatic Content Extraction) (Walker et al., 2006). Its relations can be divided into six major types, namely `geo-political-entity affiliation`, `physical`, `person-social`,

`employment-organization`, `agent-artifact`, `part-whole`. In the last years, the dataset has been used by, e.g., (Gormley et al., 2015; Nguyen and Grishman, 2015; Yu et al., 2015; Nguyen and Grishman, 2016). For a more detailed description, see Section 5.5.

Datasets which better reflect the relations in a knowledge base are mostly automatically created based on distant supervision since human annotations are expensive and time-consuming. A widely used dataset which applies distant supervision with Freebase relations to the New York Times corpus (Sandhaus, 2008) has been created by Riedel et al. (2010). Recently, for instance, Chen et al. (2014), Fan et al. (2014), Grave (2014), Nagesh et al. (2014), Zeng et al. (2015), Jiang et al. (2016), Lin et al. (2016) and Wu et al. (2017) have experimented with that dataset. We have created our training set for slot filler classification in a similar way since clean (manually cleaned or labeled) large-scale training sets were not available at the time of our participation in slot filling. Recently, Zhang et al. (2017c) have announced to publish a large-scale training set for slot filler classification which has been labeled via crowdsourcing.

Models

While the SemEval2010-task 8 shared task winning system was a support vector machine (Rink and Harabagiu, 2010), the current state-of-the-art systems for the different relation extraction datasets are neural networks. Some groups train convolutional neural networks, e.g., (dos Santos et al., 2015; Liu et al., 2015; Nguyen and Grishman, 2015; Xu et al., 2015a; Cai et al., 2016; Lin et al., 2016; Nguyen and Grishman, 2016; Vu et al., 2016; Wang et al., 2016a; Yang et al., 2016b), while other groups use variants of recurrent neural networks (vanilla RNNs, GRUs or LSTMs) (Xu et al., 2015b; Cai et al., 2016; Nguyen and Grishman, 2016; Verga et al., 2016; Vu et al., 2016; Zhou et al., 2016; Wu et al., 2017). Sometimes model variations, e.g., a recurrent layer followed by a max-pooling layer, are applied (Zhang and Wang, 2015; Nguyen and Grishman, 2016; Verga et al., 2016) and some studies integrate attention layers into CNNs, e.g., (Wang et al., 2016a) and RNNs, e.g., (Zhou et al., 2016). One of the current state-of-the-art models on the SemEval2010-task 8 dataset is a CNN-based model with two layers of attention. Only a few groups train both convolutional and recurrent neural networks for a direct comparison: Vu et al. (2016) and Nguyen and Grishman (2016) conclude that a CNN outperforms an RNN on the SemEval2010-task 8 dataset and a combination of both models by voting performs best. While the results of Xu et al. (2015b) also support this performance ranking of models, Zhang and Wang (2015), on the other hand, find an RNN outperforming a CNN model on the same dataset. On the ACE 2005 dataset, Nguyen and Grishman (2016) find the performance of CNN and RNN to be comparable. In this thesis, we use CNN models for relation classification since they are better suited for extracting phrasal patterns of relations (cf., Zhang et al. (2016c)). Moreover, in our experiments for (Adel and Schütze, 2015), we have found CNNs outperforming RNNs on our slot filling benchmark and RNNs adding only little performance gains to the final slot filling results when comparing them with CNNs. In Chapter 4, we compare CNNs with RNNs for the task of uncertainty detection and also find CNNs outperforming RNNs. The main differences of our `contextCNN` compared to other

CNNs for relation extraction are the splitting of the context at the positions of the relation arguments and the flag indicating the order of the relation arguments for distinguishing inverse relations.

Zeng et al. (2015) present a network called piecewise CNN (`PCNN`) which is especially designed for the relation classification task. After applying convolution, it splits the sentence representation into three parts around the two relation arguments. Then, it performs max pooling over the three parts individually. Sudo et al. (2016) apply it to slot filler classification, Wu et al. (2017) and Jiang et al. (2016) apply it to relation extraction on the NYT+Freebase dataset created by Riedel et al. (2010). In contrast, we propose to split the sentence even earlier and apply the convolutional filters to each part separately. This leads to a cleaner context split since the convolutional filters are not applied across context boundaries as they are in the `PCNN`. Because of its state-of-the-art performance for relation extraction, we use `PCNN` as a baseline model for our experiments in Section 3.5.2 and Section 5.2.3. In both sections, our `contextCNN` outperforms `PCNN` on different distantly supervised datasets.

Other state-of-the-art models for relation extraction are recursive neural networks (Hashimoto et al., 2013; Ebrahimi and Dou, 2015; Liu et al., 2015), as proposed by Socher et al. (2012), or models based on matrix factorization (Fan et al., 2014; Rocktäschel et al., 2015; Verga et al., 2016), for instance `Universal Schema` (Yao et al., 2012; Riedel et al., 2013). In `Universal Schema`, a matrix is built consisting of entity pairs as rows and relations (either from a knowledge base or textual patterns) as columns. After learning latent features with a ranking objective, new facts can be inferred by reasoning over the matrix. Recently, Ye et al. (2017) have presented a model which learns correlations between relational classes and Wu et al. (2017) have proposed adversarial training to train more robust models.

Model Input

While many studies directly use the sequence of words, represented by word embeddings, as input to the neural network models, several groups create the model input from constituency parse trees, e.g., (Socher et al., 2012; Hashimoto et al., 2013), or dependency parse trees, e.g., (Ebrahimi and Dou, 2015; Liu et al., 2015; Xu et al., 2015a,b; Cai et al., 2016; Yang et al., 2016b). Models relying on word sequences are often provided with position embeddings (Nguyen and Grishman, 2015; Lin et al., 2016; Nguyen and Grishman, 2016; Vu et al., 2016) or position indicators (Zhang and Wang, 2015; Vu et al., 2016; Zhou et al., 2016) as additional input. Position indicators are special tokens which mark the position of the two relation arguments while position embeddings encode the distance of each word to the two relation arguments. Especially for the SemEval2010-task 8 dataset, most studies also use a variety of linguistic features as input, such as part-of-speech tags, hypernyms, etc.

3.8 Summary of Contributions

In this chapter, we described the modular system we built for the slot filling shared task evaluations. It tackles several natural language processing challenges, such as alternate or misspelled names, ambiguous entity names, coreference resolution, location inference and domain-specific characteristics of documents. Ranked on third position in the official slot filling evaluation 2015, it is one of the state-of-the-art systems for that task. We provide its source code at `http://cistern.cis.lmu.de/CIS_SlotFilling`.

Moreover, we proposed `contextCNN`, a convolutional neural network with context splitting for the slot filler classification module and showed that it outperforms `PCNN`, a state-of-the-art CNN architecture for relation extraction. Its implementation is provided at `http://cistern.cis.lmu.de/CIS_SlotFilling`. When adding it to the slot filler classification module and combining it with traditional models for slot filling (patterns and support vector machines), it improves the classification as well as the overall pipeline results. With this system, we were one of the first to successfully apply neural networks to slot filler classification in the context of challenging model inputs (long contexts which are potentially skewed due to pipeline effects) and noisy distantly supervised training data.

In order to enable a faster system development and a better comparison of individual components among participating teams, we created a benchmark development and test set for slot filler classification based on the manual assessments from the previous years. This dataset also includes a genre-based split of documents in order to investigate and improve the genre-dependent performance of models. We publish our scripts for reproducing the dataset at `http://cistern.cis.lmu.de/SFbenchmark`.

Moreover, we empirically validated the importance of coreference resolution for the slot filling task and made the result of our coreference pre-processing of the slot filling source corpus available at `http://cistern.cis.lmu.de/corefresources`.

Chapter 4

Uncertainty Detection

Erklärung nach §8 Absatz 5 der Promotionsordnung: This chapter covers work published at the peer-reviewed international conference European Chapter of the Association for Computation Linguistics (EACL) in 2017 (Adel and Schütze, 2017c).

For a declaration of co-authorship and attribution, see page xxv and following.

4.1 Task and Motivation

When using knowledge bases or knowledge graphs as world knowledge sources, e.g., in question answering or information-retrieval systems, it is essential that the information stored in the knowledge base is correct. When populating knowledge bases automatically, as described in Chapter 3, the precision of the extractions is important. Another factor, which needs to be considered carefully, is factuality. In this chapter, we will use the term *uncertainty* to refer to speculation, opinion, vagueness and ambiguity. Uncertainty can, for example, stem from missing information (Vincze, 2014b; Zhou et al., 2015b). An example relation which often appears in non-factual contexts is the cause of death of a person. Consider sentences like "X apparently died of Y", "X may have died of Y", "X likely died of Y" or "X who reportedly died of Y".[1] For knowledge base population, we need to be able to distinguish between facts (certain information) und uncertain information. For example, we want to add the fact `Basque(X)` to a knowledge base only if it is extracted from a sentence like "X is Basque" but not if it is extracted from "X may be Basque" or "X was rumored to be Basque". Besides knowledge base population, the distinction between uncertain (non-factual) and certain (factual) information is also important for many other natural language processing tasks, such as information extraction, question answering, medical information retrieval, opinion detection or sentiment analysis (Karttunen and Zaenen, 2005; Vincze, 2014a; Díaz et al., 2016).

We conduct our experiments on the dataset from the CoNLL2010 hedge cue detection task (Farkas et al., 2010). It consists of two medium-sized corpora from different

[1] Examples from the TAC slot filling shared task assessments.

corpus	label	sentence
Wiki	u	Aganju is heavily associated with Shango, with some stating that he is Shango's father, if not at least his brother.
Wiki	c	Formerly, both the A50 Leicester to Stoke-on-Trent road and the A453 Birmingham to Nottingham road passed through the town centre.
Wiki	u	Their style of music may be described as post-punk with the elements of ska, Latin American music and funk.
Wiki	c	The total lung capacity (TLC) may be reduced through alveolar wall thickening; however this is not always the case.
Bio	u	In human disease, the mRNA levels of both IL-23p19 and IL-12p40 were shown to be increased in skin lesions of psoriatic patients (24), suggesting that elevated IL-23 contributed to the pathogenesis of psoriasis.
Bio	c	It was of interest in this regard that the Th17 cells were found to selectively produce IL-21 (35-38).
Bio	u	As O-GlcNAcylation appears to have a yin-yang relationship with phosphorylation, the same may apply for O-GlcNAcylation.
Bio	c	But it has to be emphasized that O-GlcNAcylation of Sp1 does not generally inhibit transcription.

Table 4.1: Examples from the CoNLL2010 hedge cue detection dataset, u: uncertain, c: certain.

domains (Wikipedia and Biomedical) that allow us to run a large number of comparative experiments with different neural networks and investigate different architectural choices.

4.1.1 Hedge Cues

Farkas et al. (2010) mention the following categories of triggers for uncertainty ("hedge cues"):

- auxiliaries (examples: "may", "might", "can", "would")
- verbs with speculative content (examples: "suggest", "presume", "seem")
- adjectives or adverbs (examples: "probable", "likely")
- more complex phrases (example: "raises the question of")

Table 4.1 shows randomly picked examples from the Wikipedia and Biomedical test sets and their sentence-level labels. Annotators have manually labeled the sentences with hedge cues (and their scopes in the Biomedical datasets). For all sentences containing at least one cue, "uncertain" is chosen as the sentence-level label (Farkas et al., 2010). Note that negation is not part of the uncertain category (see last row of Table 4.1). The examples from Wikipedia show that recognizing keywords is not enough for accurately predicting the uncertainty label. For instance, the modal "may" can occur both in an uncertain and

a certain sentence. The examples from the Biomedical domain show that the language is very specific with many technical terms, which may pose challenges to statistical models.

4.2 Attention-based Models

Since uncertainty is often expressed by specific trigger patterns (hedge cues, see Section 4.1.1), such as "may have", "probably", "hardly" or "has been suggested", we investigate attention-based models for uncertainty detection. In particular, we train CNNs and RNNs with gated recurrent units (GRU) and integrate different attention layers.

Although attention layers have been quite successful in NLP (see Section 2.2.3 or Section 4.7), the design space of architectures with attention layers has not been fully explored. Therefore, we investigate different dimensions of attention in this thesis and develop novel ways to calculate attention weights and integrate them into neural networks. In particular, we make a first attempt to systematize the design space of attention and investigate three dimensions of this space: weighted vs. unweighted selection, sequence-agnostic vs. sequence-preserving selection, and internal vs. external attention. Our models are motivated by the characteristics of the uncertainty detection task. However, we believe that they can be useful for other NLP tasks as well since, for example, weighting can increase the flexibility and expressivity of a neural network, and external resources can add information which can be essential for good performance. Moreover, word order is often critical for meaning and can, therefore, be an important feature for different NLP tasks.

4.2.1 Overview of Model

Figure 4.1 depicts a high-level view of our model. The input sentence is tokenized using Stanford `CoreNLP` (Manning et al., 2014) and each token is represented by a word embedding. As before, we use word embeddings trained on Wikipedia (see Section 2.2.3). The resulting word embedding matrix is then either processed by a CNN+3-max-pooling layer or by a bidirectional gated RNN (RNN-GRU) with gradient clipping to avoid exploiding gradients. Then, attention is applied, either on the word embedding matrix or on the CNN/RNN-GRU output (indicated by dashed lines in Figure 4.1; cf., Section 4.2.2). Since k-max pooling (CNN) and recurrent hidden layers with gates (RNN-GRU) have strengths complementary to attention, we experiment with concatenating the attention information $\boldsymbol{a} \in \mathbb{R}^A$ to the neural sentence representations $\boldsymbol{r} \in \mathbb{R}^m$ which is either the CNN pooling result or the last hidden state of the RNN-GRU. The final hidden layer then has the following form:

$$\boldsymbol{h} = \tanh(\boldsymbol{W}^{ah}\boldsymbol{a} + \boldsymbol{W}^{rh}\boldsymbol{r} + \boldsymbol{b}) \tag{4.1}$$

with $\boldsymbol{W}^{ah} \in \mathbb{R}^{H \times A}$ and $\boldsymbol{W}^{rh} \in \mathbb{R}^{H \times m}$ being weight matrices and $\boldsymbol{b} \in \mathbb{R}^H$ being the bias vector learned during training.

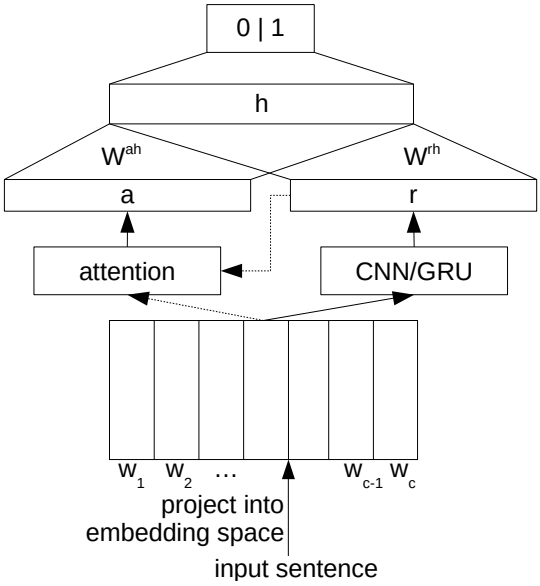


Figure 4.1: Network overview: combination of attention and CNN/RNN-GRU output. For details on attention, see Figure 4.2.

4.2.2 Dimensions of Attention Design Space

In this section, we identify three dimensions of the design space of attention. Furthermore, we propose new architectures along those dimensions.

Weighted vs. Unweighted Selection

Pooling, as widely applied in CNN models, can be seen as unweighted selection: it extracts the average or maximum values without applying any weights to them. In contrast, attention can be thought of as a weighted selection mechanism: The input elements are weighted by the attention weights, which allows the model to focus on a few highly weighted elements and ignore other elements, which have received weights close to zero. The advantage of weighted selection is that the model learns to decide based on the input how many values it should select. In contrast, pooling either selects all values (average pooling) or k values (k-max pooling). Thus, if we apply pooling to uncertainty detection and there are more than k uncertainty cues in a sentence, k-max pooling is not able to focus on all of them.

Sequence-agnostic vs. Sequence-preserving Selection

Attention is generally implemented as a *weighted average* of input vectors:

$$\boldsymbol{a} = \sum_i \boldsymbol{a}_i \tag{4.2}$$

with $\boldsymbol{a}_i \in \mathbb{R}^A$ being the weighted input vectors: $\boldsymbol{a}_i = \alpha_i \cdot \boldsymbol{X}_i^\top$ (see Equation 2.21). As in Equation 2.21, α_i denotes the attention weight and $\boldsymbol{X}_i$ is the vector which should be

weighted by attention. We call this sum an average because the α_i are normalized to sum to 1 (see Equation 2.22) and the standard term for this is "weighted average". In this work, we introduce a variant of this: *k-max average* attention:

$$\boldsymbol{a} = \sum_{\text{rank}_j(\alpha_j) \leq k} \boldsymbol{a}_j \tag{4.3}$$

where $\text{rank}_j(\alpha_j)$ is the rank of α_j in the list of attention weights $\boldsymbol{\alpha}$ in descending order. This type of averaging may be more robust than the normal average over all elements because elements with low weights (which may just be noise) will be ignored. Note that the weights $\boldsymbol{\alpha}$ are still normalized to sum to 1 for the whole sequence, not just for the k selected values.

Taking an average of input values implies that all ordering information from the input is lost and cannot be recovered by the next layer. Average pooling or max pooling is also sequence-agnostic. However, we argue that order information is needed for some NLP classification tasks. An example is when negation or uncertainty scopes need to be considered. For uncertainty detection, for instance, the order of the input can help distinguish phrases like "it is not uncertain that X is Basque" and "it is uncertain that X is not Basque". For pooling, there exists a variant which is (at least partly) sequence-preserving: k-max pooling (Kalchbrenner et al., 2014) (see Section 2.2.3). It outputs the subsequence with the highest elements of the input sequence in their original order. Similarly, we propose two sequence-preserving ways of attention. The first one is *k-max sequence*:

$$\boldsymbol{a} = [\boldsymbol{a}_j | \operatorname*{rank}_j(\alpha_j) \leq k] \tag{4.4}$$

where $[\boldsymbol{a}_j | P(\boldsymbol{a}_j)]$ denotes the subsequence of sequence $A = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_J]$ from which members not satisfying predicate P have been removed. Note that the resulting sequence $\boldsymbol{a} \in \mathbb{R}^{A \times k}$ is in the original order of the input, i.e., not sorted by value. The second sequence-preserving attention method is *k-max pooling*. It ranks each dimension of the vectors individually, thus the resulting values can stem from different input positions. This is the same as standard k-max pooling in CNNs except that each vector element in $\boldsymbol{a}_j$ has been weighted (by its attention weight α_j), whereas in standard k-max pooling it is considered as is. Below, we also refer to k-max sequence as "*per-pos*" (since it selects all values of the k positions with the highest attention weights) and to k-max pooling as "*per-dim*" (since it selects the k largest weighted values per dimension) to distinguish it from k-max pooling done by the CNN. Note that CNNs and RNNs capture some information of sequence as well in their outputs: The length of sequences CNNs are able to handle is limited to the width of their filters. RNNs can theoretically capture information of longer sequences as well, however all information gets conflated in their hidden state with a limited storage capacity. In contrast, sequence-preserving attention explicitly stores the most relevant inputs in their order of appearance.

Internal vs. External Attention

When designing the attention layer, we distinguish between *focus* and *source* of attention.

The focus of attention is that layer of the network which is reweighted by attention weights, corresponding to $\boldsymbol{X}$ in Equation 2.21. We investigate two options in this thesis: focus on the input, i.e., the matrix of word vectors, and focus on the hidden representation, i.e., the output of the convolutional layer of the CNN or the hidden layers of the RNN-GRU. For focus on the input, we first apply tanh to the word vectors to improve results (see Figure 4.2).

The source of attention is the information source used to compute the attention weights, corresponding to the input of e in Equation 2.22.

Note that source and focus of attention are slightly related to keys and values used in memory networks (Miller et al., 2016): Similar to sources, keys are used to calculate weights. Similar to focuses, values are weighted and passed to the next layer of the network. However, while there is only one key for each memory cell, our conceptualization allows several sources. Moreover, we do not group specific sources and focuses to pairs like key-value pairs. In contrast, sources and focuses are independent of each other. Finally, there is no need of storing focus vectors in a memory.

For attention in the literature, both focus and source are based only on information internally available to the network (through input or hidden layers). We call this *internal attention*.[2] This is also formalized by Equation 2.23 and Equation 2.24 with $\boldsymbol{X}$ being the internal information used for focus and source of attention. Attention in machine translation or question answering, for example, computes attention weights by comparing two vectors, namely the source sentence to the previously translated target representation or the document to the question representation (see Equation 2.25). In this case, the source of attention ($\boldsymbol{X}$ and $\boldsymbol{c}$) contains more information than the focus ($\boldsymbol{X}$), however, all those information are given by the intermediate hidden or output representations (previous translations) or the task input (question). We, therefore, also categorize this as internal attention.

In this work, we investigate increasing the scope of the source beyond the input and, thus, making the attention mechanism more powerful. We refer to an attention layer as *external attention* if its source includes an external resource. For uncertainty detection, for instance, it can be beneficial to give the model a lexicon of seed cue words or phrases. Thus, we provide the network with additional information to bear on identifying and summarizing features. This can simplify the training process by guiding the model to recognizing uncertainty cues. The specific external-attention scoring function we use for uncertainty detection is similar to the one used, e.g., in machine translation (see Equation 2.25) with $\boldsymbol{c} := \boldsymbol{C_j}$ being an additional input from an external source. It is parameterized by $\boldsymbol{U}^1 \in \mathbb{R}^{d \times A}$, $\boldsymbol{U}^2 \in \mathbb{R}^{d \times E}$ and $\boldsymbol{v} \in \mathbb{R}^d$ and defined as follows:

$$e(\boldsymbol{X}_i, \boldsymbol{C}) = \sum_j \boldsymbol{v}^\top \tanh(\boldsymbol{U}^1 \boldsymbol{X}_i^\top + \boldsymbol{U}^2 \boldsymbol{C}_j) \tag{4.5}$$

where $\boldsymbol{C}_j \in \mathbb{R}^E$ is a vector representing a cue phrase j of the training set. We compute $\boldsymbol{C}_j$ as the average of the embeddings of the constituting words of j.

[2] Gates, e.g., the weighting of h^{t-1} in Equation 2.17, can also be viewed as internal attention mechanisms.

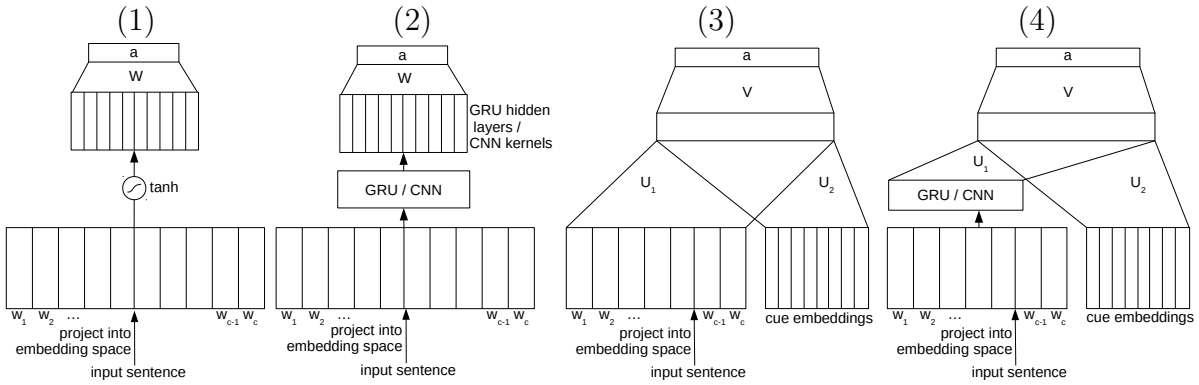


Figure 4.2: Internal attention on (1) input and (2) hidden representation. External attention on (3) input and (4) hidden representation. For the whole network structure, see Figure 4.1.

Thus, the attention layer consists of a feed-forward hidden layer of size d, which compares the input word $\boldsymbol{X}_i$ to each cue vector $\boldsymbol{C}_j$ and then sums the results. The weights $\boldsymbol{U}^1$, $\boldsymbol{U}^2$ and $\boldsymbol{v}$ are learned during training. When applying this function e in Equation 2.22, the attention weights α_i show how important each input $\boldsymbol{X}_i$ is for uncertainty detection given the external knowledge about possible cue phrases $\boldsymbol{C}_j$. The underlying intuition is similar to attention for machine translation, which learns alignments between source and target sentences. Instead, we learn "alignments" between the input and external cue phrases. Since we use embeddings to represent words and cues, uncertainty-indicating phrases that did not occur in training but are similar to training cue phrases can also be recognized. For using external attention with the proposed scoring function e to another task, only a set of vectors $\boldsymbol{C}_j$ needs to be created which can be used to determine the relevance of the input embeddings to the task at hand.

Note that external attention is not the only way of incorporating prior knowledge into a neural network model. (An alternative could be the extension of the input word embeddings with flags indicating whether the word is part of a cue phrase or not.) It has rather been designed to improve and speed up the training of the attention module of a neural network. Typically, the attention module is fully unsupervised. In contrast, the cue vectors of external attention provide additional signals to guide the training of the network in the desired direction.

Figure 4.2 shows the four settings arising from our distinction between source and focus of attention: (1) internal attention on the input, (2) internal attention on the hidden representation, (3) external attention on the input, and (4) external attention on the hidden representation. A schematic view of internal and external attention as defined via source and focus is given in Figure 4.3.

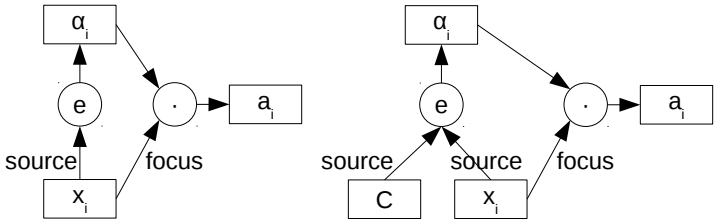


Figure 4.3: Schemes of focus and source: left: internal attention, right: external attention.

domain	train	test
Wikipedia	11,111	9634
Biomedical	14,541	5003

Table 4.2: Statistics of CoNLL2010 hedge cue detection datasets.

4.3 Dataset

We evaluate our neural networks and attention architectures on the datasets from the CoNLL 2010 shared task "Learning to Detect Hedges and their Scope in Natural Language Text" (Farkas et al., 2010). The shared task provides two datasets from different domains: one from Wikipedia and one from the Biomedical domain. The hedge cue detection task is a binary sentence classification task: For each sentence, the model has to decide whether it contains uncertain information (a hedge cue) or not. The sentences have been annotated manually, making use of so-called Weasel words[3] for Wikipedia, which go beyond the hedge cue categories provided in Section 4.1.1 since they also include, for example, numerically vague expressions, such as "many", "one group of", "more than 60%". In the annotations, not only the sentence-level labels are given but also the hedge cues are marked. An example is the following training instance:

> Earths and clays <ccue>may</ccue> have provided prehistoric peoples with <ccue>some of their first medicines</ccue>.

The hedge cues are marked with the xml tag <ccue>. We use those marked hedge cues from the training set to build the lexicon of hedge cues for the external attention mechanism.

The statistics provided in Table 4.2 show that the datasets are medium-sized. Thus, neural networks might not have a per-se advantage over non-neural models. On the other hand, training the models does not take a long time, which allows conducting experiments with different architectural choices. For hyperparameter tuning, we split the training set into core-train (80%) and dev (20%) sets. Table B.1 in the appendix provides the hyperparameters for the different models.

For evaluation, we apply the official shared task measure: F1 of the uncertain class.

[3] http://en.wikipedia.org/wiki/Weasel_word.

	model	Wiki	Bio
(1)	baseline SVM	62.01⋆	78.64⋆
(2)	baseline RNN-GRU	59.82⋆	**84.69**
(3)	baseline CNN	**64.94**	84.23

Table 4.3: F1 results for uncertainty detection. Baseline models without attention. ⋆ indicates significantly worse than best model (in bold).[5]

	model	Wiki	Bio
(2)	baseline RNN-GRU	59.82⋆	84.69
(4)	RNN-GRU attention-only	62.02⋆	**85.32**
(5)	RNN-GRU combined	58.96⋆	84.88
(3)	baseline CNN	64.94⋆	84.23
(6)	CNN attention-only	53.44⋆	82.85
(7)	CNN combined	**66.49**	84.69

Table 4.4: F1 results for uncertainty detection. Attention-only vs. combined architectures. Sequence-agnostic weighted average for attention. ⋆ indicates significantly worse than best model (bold).

4.4 Results

In this section, we present the results of our models on the uncertainty detection task and compare them with previously published state-of-the-art results.

4.4.1 Baselines without Attention

To be able to assess the impact of our attention architectures on the classification results, we implement a standard CNN model as well as a standard RNN-GRU model without attention. Those architectures correspond to the right branch of the model shown in Figure 4.1. Moreover, we re-implement the top-ranked system on Wikipedia in the CoNLL-2010 shared task (Georgescul, 2010). It uses bag-of-word feature vectors with only hedge cues from the training set as vocabulary. Although we follow the instructions provided by Georgescul (2010) carefully and also set the SVM parameters to their values, our re-implementation is slightly better than the published result: 62.01 vs. 60.20 on Wiki, 78.64 vs. 78.50 on Bio.

The results of the baselines are given in Table 4.3. The CNN (line 3) outperforms the SVM (line 1) on both datasets, presumably because it considers all words in the sentence – instead of only pre-defined hedge cues – and makes effective use of this additional information. The RNN-GRU (line 2) performs better than the SVM and CNN on Biomedical data, but worse on Wikipedia. In Section 4.5.2, we investigate possible reasons for the different performance of CNN and RNN-GRU on the different datasets.

[5] randomization test with $p<0.05$.

4.4.2 Experiments with Attention Mechanisms

In this section, we investigate the different attention architectures which can be derived from the dimensions of attention we described in Section 4.2.2.

Weighted vs. Unweighted Selection

For the first set of experiments, we follow the standard approach in NLP and use the sequence-agnostic weighted average for attention (see Equation 4.2). We compare the weighted attention average (weighted selection) against k-max pooling in CNNs (unweighted selection) as well as against the gating mechanism in RNN-GRUs. Moreover, we compare attention-only architectures against combining the attention output with the CNN/RNN-GRU output. Table 4.4 shows the resulting F1 scores. An attention-only model works well for attention on the RNN-GRU hidden states but not for attention on the CNN hidden representation. This indicates that attention is more powerful than the gating mechanism of RNN-GRUs alone. Attention makes the task of the RNN-GRU – remembering the entire sentence over long distances and focusing on the relevant parts for classification – much easier. For CNNs, however, pooling (unweighted selection) outperforms attention (weighted selection). This indicates that k-max pooling is already effective for focusing on the key parts of the sentence. The CNNs achieve the best results when the pooling output is combined with the attention output. This shows that attention (weighted selection) extracts complementary information. Therefore, we combine the RNN-GRU/CNN hidden representations with the attention layer for all of the following experiments, as depicted in Figure 4.1. Thus, we explore the benefits of adding attention to existing architectures in the following, as opposed to developing attention-only architectures.

Internal vs. External Attention

As depicted in Figure 4.2, we investigate four different configurations with different sources (S) and focuses (F) of attention: internal attention on the word embeddings (S=I, F=W), internal attention on the RNN-GRU/CNN hidden representation (S=I, F=H), external attention on the word embeddings (S=E, F=W) and external attention on the RNN-GRU/CNN hidden representation (S=E, F=H). Table 4.5 provides the results.

For both RNN-GRU (line 8 of Table 4.5) and CNN (line 13), the best result is obtained by focusing attention directly on the word embeddings.[6] These results suggest that it is best to optimize the attention mechanism directly on the input, so that information can be extracted that is complementary to the information extracted by a standard RNN-GRU/CNN.

For the CNN, external attention (13) is significantly better than internal attention (11) when focusing on the input word embeddings (F=W). This shows that by designing an architectural element – external attention – that simplifies the identification of hedge cue

[6]The small difference between the RNN-GRU results on Bio on lines (5) and (8) is not significant.

	model	S	F	Wiki	Bio
(2)	baseline RNN-GRU	-	-	59.82⋆	84.69
(5)	RNN-GRU combined	I	H	58.96⋆	84.88
(8)	RNN-GRU combined	I	W	62.18⋆	84.81
(9)	RNN-GRU combined	E	H	61.19⋆	84.62
(10)	RNN-GRU combined	E	W	61.87⋆	84.41
(3)	baseline CNN	-	-	64.94⋆	84.23⋆
(7)	CNN combined	I	H	66.49	84.69
(11)	CNN combined	I	W	65.13⋆	84.99
(12)	CNN combined	E	H	64.14⋆	84.73
(13)	CNN combined	E	W	**67.08**	**85.57**

Table 4.5: F1 results for uncertainty detection. Focus (F) and source (S) of attention: Internal (I) vs. external (E) attention; attention on word embeddings (W) vs. on hidden layers (H). Sequence-agnostic weighted average for attention. ⋆ indicates significantly worse than best model (bold).

properties of words, the whole learning problem is apparently made easier. For the RNN-GRU, however, external attention on the input (line 10) is not better than internal attention on the input (line 8). The results are roughly tied for both domains. External attention on the RNN-GRU hidden representation (line 9) is better than internal attention (line 5) on Wikipedia and roughly tied on Biomedical. We assume that the combination of an external resource with the more indirect sentence representation as produced by the RNN-GRU is more difficult. The RNN-GRU accumulates the information from the entire sentence in its hidden representation while the CNN creates representations of n-gram phrases in order to recognize hedge cue patterns in them. Those short-phrase representations of the CNN might be combined more effectively with external attention: If there is, for example, a strong external-attention evidence for uncertainty, then the effect of a hedge cue pattern (hypothesized by a convolutional filter) on the final decision can be boosted. As a result, the CNN with external attention achieves the best results overall. It is significantly better than the standard CNN using only pooling, both on Wikipedia and Biomedical texts. Thus, the CNN can make effective use of external information – a lexicon of uncertainty cues in our case.

Sequence-agnostic vs. Sequence-preserving Selection

As described in Section 4.2.2, we expect that sequence information is important for many NLP tasks. Since commonly used attention mechanisms simply average the vectors in the focus of attention and, thus, lose sequential information (sequence-agnostic), we propose two possibilities of sequence-preserving attention: "per-dim" and "per-pos" (see Section 4.2.2). Sequence-preserving attention is similar to k-max pooling, which also selects an ordered subset of inputs. While traditional k-max pooling is unweighted, our sequence-preserving ways of attention still make use of the attention weights.

Table 4.6 compares standard attention ("average all") with the two sequence-preserving ways of attention as well as with a hybrid design ("k-max average"), as described in

	average		k-max sequence	
	all	k-max	per-dim	per-pos
Wiki	67.08	**67.52**	66.73	66.50
Bio	**85.57**	84.36	84.05	84.03

Table 4.6: F1 results for uncertainty detection. Model: CNN, S=E, F=W (13). Sequence-agnostic vs. sequence-preserving attention.

Section 4.2.2. For these experiments, we use the CNN with external attention focused on word embeddings (Table 4.5, line 13), the best performing configuration in the previous experiments.

When "sharpening" attention and only considering the values with the top k attention weights, the performance of the model increases on Wikipedia (from 67.08 to 67.52) and decreases on Biomedical (from 85.57 to 84.36). In general, we do not expect large performance differences since attention values tend to be peaked, so for common values of k ($k \geq 3$ in most prior work on k-max pooling) we are effectively comparing two similar weighted averages, one in which most summands get a weight of 0 (k-max average) and one in which most summands get weights close to 0 (average over all, i.e., standard attention). Only for attention weight distributions which are not clearly peaked, the impact of k-max average can be higher.

The sequence-preserving architectures in Table 4.6 are slightly worse than standard attention (sequence-agnostic averaging of all vectors), but not significantly: The performance is different by about half a point. This shows that k-max sequence and attention can similarly be used to select a subset of the information available, a parallel that has not been highlighted and investigated in detail before. Our motivation for introducing sequence-preserving attention is that the semantic meaning of a sentence can vary depending on where an uncertainty cue occurs. However, the core of uncertainty detection is keyword and keyphrase detection. Thus, the overall sentence structure might be less important for this task. For tasks with a stronger natural language understanding component, such as summarization or relation extraction, on the other hand, we expect sequences of weighted vectors to outperform averaged vectors. For sentiment analysis, sequence-preserving attention can help for detecting the scope of negation and sentiment words. In (Adel and Schütze, 2017c), we show that sequence-preserving attention indeed improves results on a sentiment analysis benchmark dataset.

4.4.3 Comparison to State of the Art

Table 4.7 compares our models to the state of the art on the uncertainty detection benchmark datasets. On Wikipedia, our CNN outperforms the state of the art by more than three points. On the Biomedical domain, the best model is a structured prediction model (conditional random field) using a large number of manually designed features (i.a., lemmas, chunks, part-of-speech tags) based on an exhaustive corpus pre-processing (Tang

model	Wiki	Bio
SVM (Georgescul, 2010)	62.01	78.64
HMM (Li et al., 2014)	63.97	80.15
CRF + ling (Tang et al., 2010)	55.05	**86.79**
our CNN with external attention	**67.52**	85.57

Table 4.7: Comparison of our best model with the state of the art.

et al., 2010). Our models still achieve comparable results without pre-processing[7] or feature engineering.

4.5 Analysis

In this section, we analyze the attention layer and provide comparisons of CNNs and RNNs for uncertainty detection.

4.5.1 Analysis of Attention

Advantages of Attention

We manually look at examples for which pooling (i.e., the baseline CNN) fails but attention correctly detects an uncertainty. From this analysis, two patterns emerge: First, we find that there are many cues that have more words than the filter size (which was 3 in our experiments), e.g., "it is widely expected", "it has also been suggested". The convolutional layer of the CNN is not able to detect phrases longer than the filter size while for attention there is no such restriction. Second, some cues are spread over the whole sentence, as in the following example from the Wikipedia dataset: "Observations of the photosphere of 47 Ursae Majoris *suggested* that the periodicity *could not* be explained by stellar activity, making the planet interpretation *more likely*". We have set the uncertainty cues that are distributed throughout the sentence in italics. To see which words of this sentence get the highest attention weights using external attention, we extract and plot the attention weight distribution in Figure 4.4. With attention, the model focuses the most on the three words/phrases "suggested", "not" and "more likely" that correspond almost perfectly to the true uncertainty cues. K-max pooling of standard CNNs, on the other hand, can only select the k maximum values per dimension, i.e., it can pick at most k uncertainty cues per dimension. Since k is a hyperparameter to the model, it needs to be chosen with a hard decision before model training. With attention, on the other hand, the model can choose the optimal number of words to attend to for each sentence individually.

[7] We only tokenize the input sentences as described before.

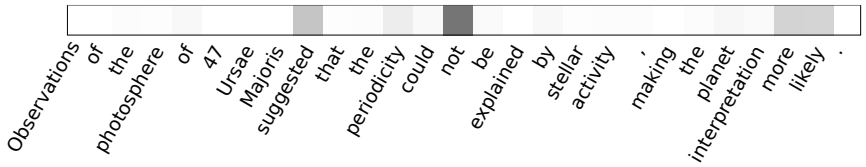


Figure 4.4: External attention weight heat map.

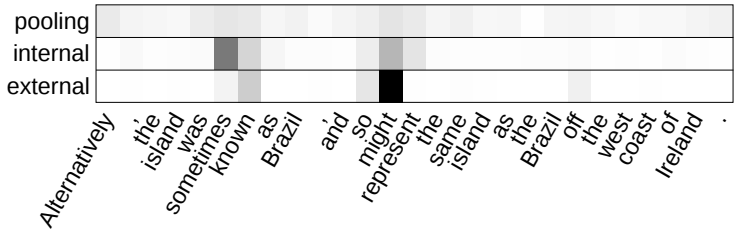


Figure 4.5: Pooling vs. internal vs. external attention.

Pooling vs. Internal vs. External Attention

For a qualitative analysis of the different selection mechanisms (pooling, internal attention, external attention), we randomly pick sentences from the Wikipedia test set and extract which parts of the input they select. For internal and external attention, this is straightforward since we can directly plot the attention weights α_i. For pooling, we calculate the relative frequency that a value from an n-gram centered around a specific word is picked. In particular, we divide the absolute frequency by the total number of pooling results (for k-max pooling, this is k times the number of convolutional filters). Figure 4.5 shows the results of the three mechanisms for an exemplary sentence. We provide figures for more sentences in the appendix (Section B.2). The observable patterns are similar across all sentences we have randomly picked: Pooling forwards information from different parts all over the sentence. It has minor peaks at relevant n-grams (e.g., "was sometimes known as" or "so might represent") but also at non-relevant parts (e.g., "Alternatively" or "the same island"). There is no clear focus on uncertainty cues. Internal attention is more focused on the relevant words for uncertainty detection. External attention finally has the clearest focus since its training is guided by prior knowledge of cue phrases.

4.5.2 Analysis of CNN vs. RNN-GRU

The behavior of CNN and RNN-GRU is different on the two domains Wikipedia and Biomedical. While the results of the CNN and the RNN-GRU are comparable on Biomedical, the CNN clearly outperforms the RNN-GRU on Wikipedia. Table 4.8 shows a comparison of characteristics of the two datasets that might affect model performance.

Although the Wikipedia dataset has a richer vocabulary (almost twice as many different words as the Biomedical dataset), it is better covered by our word embeddings, probably because they have also been trained on Wikipedia. Thus, the average number of out-of-

	Wikipedia	Biomedical
average sentence length[8]	21	27
size of vocabulary	45,100	25,300
average #OOVs per sentence	4.5	6.5

Table 4.8: Differences of Wikipedia and Biomedical dataset.

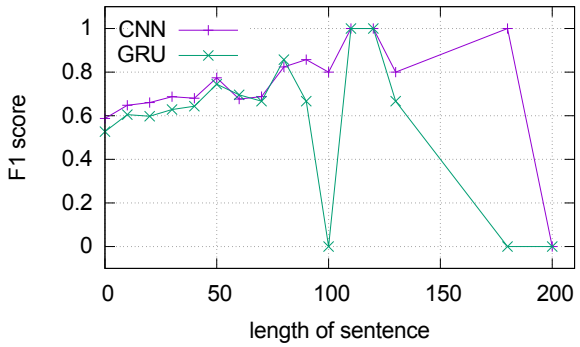


Figure 4.6: F1 results for different sentence lengths.

vocabulary (OOV) words per sentence is lower. Also, the sentences are shorter on average. All of those features can influence model performance, especially because of the different way of sentence processing: While the RNN-GRU merges all information into a single vector, the CNN extracts the most important phrases and ignores all the rest. In the following paragraphs, we analyze the behavior of the two models with respect to sentence length, number of OOVs and precision and recall scores.

Sentence Lengths

Figure 4.6 shows the F1 scores on Wikipedia of the CNN and the RNN-GRU with external attention for different sentence lengths. For a better overview, we discretize the lengths to intervals of 10, i.e., index 50 on the x-axis includes the scores for all sentences of length $l \in [50, 60)$. Most sentences (96.2%) have lengths $l < 50$. Only 0.1% of sentences have length $l > 100$. The CNN outperforms the RNN consistently across sentence lengths, with larger differences for longer sentences.

Number of Out-of-Vocabulary Words

Figure 4.7 shows a similar plot for F1 scores depending on the number of OOVs per sentence. Again, the CNN consistently outperforms the RNN-GRU independent of the number of OOVs. This indicates that the uncertainty detection task seems to be more challenging for the RNN-GRU in general, not depending on the number of out-of-vocabulary words.

[8] after tokenization with Stanford `CoreNLP`.

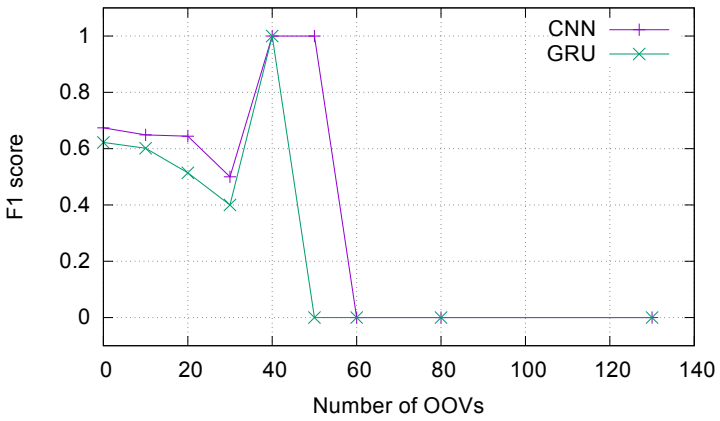


Figure 4.7: F1 results for different number of OOVs per sentence.

model	P	R
CNN	52.5	85.1
CNN + external attention	58.6	78.3
RNN-GRU	75.2	49.6
RNN-GRU + external attention	76.3	52.0

Table 4.9: Precision and recall scores of CNN and RNN-GRU on Wikipedia.

Precision and Recall

So far, we have only compared F1 scores of the different models. In this paragraph, we investigate precision and recall of the CNN and the RNN-GRU. Table 4.9 provides the values for four models on the Wikipedia dataset: CNN and RNN-GRU with and without external attention.

The scores show a very important difference between the two models: The CNN models offer high-recall predictions while the predictions of the RNN-GRU models have higher precision. This suggests that the RNN-GRU predicts uncertainty more reluctantly than the CNN. The same analysis on Biomedical reveals that the precision and recall values are almost the same for both models. This might be a reason why the performance of the models is similar on Biomedical but different on Wikipedia. A reason for that might be the different characteristics of the two corpora as shown in Table 4.8. The larger vocabulary of the Wikipedia dataset, for example, might pose more challenges to the RNN than to the CNN in identifying uncertain sentences. As a result, its recall is considerably lower on the Wikipedia dataset.

Note that F1 scores can be optimized by tuning the prediction thresholds for the different classes. However since the classification task is binary, we do not tune them here but decide for the uncertain or certain class depending on which output score is higher.

	uncertainty detection	micro P	micro R	micro F1	macro F1
hop 0	–	31.79	**28.23**	29.91	**34.20**
hop 0	+	**33.15**	28.00	**30.35**	33.72
hop 1	–	9.80	**7.00**	8.17	8.28
hop 1	+	**10.63**	6.89	**8.36**	**8.35**
all	–	23.80	**19.42**	21.39	**24.92**
all	+	**25.21**	19.23	**21.82**	24.64

Table 4.10: CSLDC max micro/macro scores with and without uncertainty detection.

4.6 Application to Slot Filling

We apply uncertainty detection as an additional component in the slot filling pipeline between the slot filler classification and the postprocessing component. In particular, we run the best uncertainty detection network trained on Wikipedia (with external attention on the input word embeddings and k-max average) on the middle contexts of the slot filling candidates. If it detects an uncertainty cue, we do not consider the filler candidate for output. We choose the network trained on Wikipedia since we expect Wikipedia texts to be more similar to news and discussion forum texts than Biomedical texts. We decide to only consider uncertainty cues in the middle contexts in order to reduce the possibility of detecting an uncertainty cue which is not in the scope of the relation. An example is the following sentence (taken from the slot filling dataset, LDC2015E77):

> While the world is convinced that Barack Obama, the US' liberal, Democratic president, is a tech-savvy politician who...

The phrase "while the world is convinced" indicates that the statement "Barack Obama is a tech-savvy politician" is an opinion rather than a proven fact. However, the triple (`Barack Obama`, `per:title`, `president`) is still a fact. By considering only the middle context between `Barack Obama` and `president` for uncertainty detection (i.e., ", the US' liberal, Democratic"), the model is not distracted by the phrase indicating uncertainty. On the other hand, consider the following sentence (also taken from the slot filling dataset, LDC2015E77):

> ... Abubakar Shekau, may have died of gunshot wounds some weeks after a clash...

By considering the middle context between `Abubakar Shekau` and `gunshot wounds`, it is possible to recognize that the `per:cause_of_death` relation does not necessarily hold between the entity and the phrase because of the modal verb "may". Several studies on relation classification, e.g., (Zeng et al., 2014), even classify relations considering only the middle contexts. Thus, the assumption that most relation specific context is contained in the middle context is reasonable.

Table 4.10 provides the results of the slot filling system using patterns, binary support vector machines with skip n-gram features and binary `contextCNNs` with and without

integrating the uncertainty detection module. The uncertainty detection module leads to higher precision values with only slightly reduced recall values. As a result, it can improve the micro F1 scores of the slot filling pipeline.

For example, the uncertainty detection module correctly filters the context "e_1 is believed to have been born in Connemara in e_2" and, thus, improves precision of the final system. However, by wrongly identifying uncertainty in "e_1 who is considered one of the world's foremost Legal e_2", it prevents the system from detecting a possibly correct slot filler and, thus, reduces recall.

4.7 Related Work

4.7.1 Uncertainty Detection

The concept of uncertainty has been extensively studied in linguistics, e.g., (Kiparsky and Kiparsky, 1970; Karttunen, 1973; Lakoff, 1975; Karttunen and Zaenen, 2005). Different terms used in linguistics which may denote slightly different but still similar linguistic phenomena are, i.a., modality (Palmer, 2001; Szarvas et al., 2012), factuality (Saurí and Pustejovsky, 2012) or veridicality (Karttunen and Zaenen, 2005; De Marneffe et al., 2012). Another related concept is negation. Although its semantics is different since it might refer to certain information which is not true, it is often studied in combination with speculation or modality (Baker et al., 2012; Velldal et al., 2012). See (Vincze, 2014b) for more detailed elaborations on the linguistic background.

Domains and Datasets

In natural language processing, many studies on uncertainty detection have focused on the Biomedical domain, e.g., (Friedman et al., 1994; Light et al., 2004; Farkas and Szarvas, 2008; Kilicoglu and Bergler, 2008; Vincze et al., 2008; Kim et al., 2009; Morante and Daelemans, 2009; Özgür and Radev, 2009; Uzuner et al., 2009). Other studied domains include news (Sauri, 2008; Saurí and Pustejovsky, 2009), Wikipedia (Ganter and Strube, 2009; Farkas et al., 2010) or social media (Wei et al., 2013). Recently, uncertainty detection has been applied in the review domain (Pang and Lee, 2004; Wilson et al., 2005; Cruz et al., 2016), for political statements (Štajner et al., 2016) and tweets (Reichel and Lendvai, 2016). Szarvas et al. (2012), Vincze (2014b) and Zhou et al. (2015b) conduct cross-domain experiments. Benchmark corpora include FactBank (Saurí and Pustejovsky, 2009), used by, e.g., (Sauri, 2008; Saurí and Pustejovsky, 2009, 2012; Szarvas et al., 2012; Zhou et al., 2015b), WikiWeasel (Ganter and Strube, 2009), used by, e.g., (Ganter and Strube, 2009; Szarvas et al., 2012; Zhou et al., 2015b; Jean et al., 2016), BioScope (Vincze et al., 2008), used by, e.g., (Morante and Daelemans, 2009; Özgür and Radev, 2009; Szarvas et al., 2012; Zhou et al., 2015b; Jean et al., 2016), or datasets prepared for shared task evaluations, such as the BioNLP'09 Shared Task on Event Extraction (Kim et al., 2009) or the CoNLL 2010 shared task on detecting hedge cues (Farkas et al., 2010). The annotation of FactBank, for instance, includes perspective, level of factuality and polarity. De Marneffe et al. (2012)

extend FactBank by crowdsourcing. In this work, we use the CoNLL 2010 shared task dataset instead since it provides larger train/test sets and its annotation consists of only two labels (certain/uncertain) instead of various perspectives and degrees of uncertainty. Saurí and Pustejovsky (2012) categorize related work into two main approaches: identifying speculative cues (and their scope) vs. determining the factuality values from the identified cues. In this work, we contribute to the first category. We consider this a reasonable first step when aiming at applying uncertainty detection to information extraction tasks like KBP slot filling.

Models and Features

Early systems on uncertainty detection apply rules with hand-crafted lexicons of uncertainty cues (Friedman et al., 1994; Light et al., 2004; Farkas and Szarvas, 2008; Sauri, 2008; Conway et al., 2009; MacKinlay et al., 2009; Uzuner et al., 2009; Van Landeghem et al., 2009; Saurí and Pustejovsky, 2012; Štajner et al., 2016) or algorithms based on linguistic knowledge (Sauri, 2008; Saurí and Pustejovsky, 2012). Other studies train machine learning models, such as Naive Bayes classifiers (Štajner et al., 2016), decision trees (Morante and Daelemans, 2009), generalized linear models (Reichel and Lendvai, 2016), maximum entropy models (MacKinlay et al., 2009; Saurí and Pustejovsky, 2009; Clausen, 2010; Szarvas et al., 2012; Štajner et al., 2016), support vector machines (Light et al., 2004; Özgür and Radev, 2009; Uzuner et al., 2009; Georgescul, 2010; Velldal, 2010; Cruz et al., 2016; Jean et al., 2016; Štajner et al., 2016) or conditional random fields (Li et al., 2010; Rei and Briscoe, 2010; Tang et al., 2010; Szarvas et al., 2012). To the best of our knowledge, we are the first to use neural networks for uncertainty detection. (In the CoNLL shared task evaluations, there was only one team using average perceptrons (Ji et al., 2010) but no multi-layer neural network.)

In contrast to our model, most models represent the input by a variety of features, such as, but not limited to, stems, lemmas, prefixes and suffixes, part-of-speech tags, grammatical relations, document position features or features derived from chunking or dependency parsing. Also effective are dictionaries of possible hedge cues or modal verbs. Table 4.11 shows which studies use which features. Saurí and Pustejovsky (2009) emphasize the importance of context and world knowledge for uncertainty detection.

Several systems treat uncertainty detection as a token-labeling task instead of a sentence classification task and derive a label on the sentence level by aggregating the token labels. In the official CoNLL 2010 shared task evaluations, this approach has often performed better than the sentence classification approach (Farkas et al., 2010). In contrast, our experimental results show that treating uncertainty detection as a sentence classification task can also be effective. Velldal (2010) and Özgür and Radev (2009), for instance, take a third approach and directly classify each uncertainty keyword whether or not it expresses uncertainty in its current context.

feature	example studies
stems	(Özgür and Radev, 2009; Szarvas et al., 2012; Zhou et al., 2015b)
lemmas	(Morante and Daelemans, 2009; Saurí and Pustejovsky, 2009; Clausen, 2010; Li et al., 2010; Rei and Briscoe, 2010; Tang et al., 2010; Velldal, 2010; Jean et al., 2016)
prefixes, suffixes	(Tang et al., 2010; Szarvas et al., 2012)
part-of-speech tags	(Kilicoglu and Bergler, 2008; Morante and Daelemans, 2009; Özgür and Radev, 2009; Clausen, 2010; Li et al., 2010; Rei and Briscoe, 2010; Tang et al., 2010; Velldal, 2010; Szarvas et al., 2012; Zhou et al., 2015b; Jean et al., 2016)
grammatical relations	(Rei and Briscoe, 2010)
document position features	(Özgür and Radev, 2009)
chunking	(Morante and Daelemans, 2009; Li et al., 2010; Tang et al., 2010; Szarvas et al., 2012; Zhou et al., 2015b)
dependency parsing	(Kilicoglu and Bergler, 2008; Sauri, 2008; MacKinlay et al., 2009; Özgür and Radev, 2009; Rei and Briscoe, 2010; Velldal, 2010; Saurí and Pustejovsky, 2012)
dictionaries	(Saurí and Pustejovsky, 2009; Georgescul, 2010; Tang et al., 2010; Jean et al., 2016; Štajner et al., 2016)

Table 4.11: Example studies for features used in uncertainty detection.

Applications

Uncertainty detection can be helpful in various natural language processing task, such as information extraction, c.f., Karttunen and Zaenen (2005), sentiment analysis (Pang and Lee, 2004; Wilson et al., 2005; Benamara et al., 2012; Cruz et al., 2016), textual entailment or question answering (Benamara et al., 2012).

4.7.2 Attention

As described in Section 2.2.3, attention is mainly used in recurrent neural networks to allow the network to focus on relevant parts of the input (Bahdanau et al., 2015; Hermann et al., 2015; Peng et al., 2015; Rush et al., 2015; Rocktäschel et al., 2016; Yang et al., 2016c). Some studies in vision also integrate attention into CNNs (Stollenga et al., 2014; Chen et al., 2015; Xiao et al., 2015) as we do in this thesis while this is not common in NLP so far. Exceptions are, e.g., Meng et al. (2015), Wang et al. (2016a) and Yin et al. (2016). Meng et al. (2015) use several layers of local and global attention in a complex machine translation model with a large number of parameters. We have re-implemented their network to compare our models with it but it performs poorly for uncertainty detection (F1 score is 51.51/66.57 on Wiki/Bio). One reason for that might be our limited amount of training data. The dataset used by Meng et al. (2015) is an order of magnitude larger. However, our results show that our CNN architectures with attention are effective even with a smaller training set. Yin et al. (2016) compare attention-based input representations and attention-based pooling. In contrast, we keep the convolutional and pooling layers unchanged and combine their strengths with attention. Allamanis et al. (2016) apply a convolutional layer to

compute the attention weights (instead of the linear or feed-forward layer in Equation 2.23 and Equation 2.25). Experimenting with such an attention layer in combination with our attention architectures would be an interesting direction for future work.

4.8 Summary of Contributions

In this chapter, we presented our work on attention mechanisms for uncertainty detection. We experimented with different neural architectures (CNN and RNN-GRU) and made a first attempt to systematize the design space of attention. Attention is currently a popular architectural component of models for different NLP tasks. We identified three dimensions of the design space of neural selection mechanisms: weighted vs. unweighted selection, sequence-agnostic vs. sequence-preserving selection, and internal vs. external attention. Along those axes, we proposed several new attention mechanisms: external attention, k-max average attention and sequence-preserving attention. External attention allows the incorporation of external (task-specific) resources in the attention mechanism, such as a lexicon of uncertainty cues for uncertainty detection or a sentiment dictionary for sentiment analysis. K-max average attention uses only the vectors with the k maximum attention weights to compute the attention result. This can help the model to focus on the most relevant parts of the input when the distribution of attention weights is not sharp, i.e., if some weight is also put on unimportant input vectors. Sequence-preserving attention maintains the order information from the input sequence, which is lost by standard attention with average. This can be beneficial whenever word order is crucial for the correct prediction. Our models are motivated by the characteristics of the uncertainty detection task but can be easily applied to other NLP tasks as well.

We are the first to apply convolutional neural networks and recurrent neural networks with gated recurrent units to uncertainty detection. Moreover, we analyzed the different behavior of CNNs and RNN-GRUs by comparing their performance with respect to different input lengths, different numbers of out-of-vocabulary words in the input, as well as their precision and recall results. Our CNN models with external attention on the input embeddings set the new state of the art on the Wikipedia uncertainty detection dataset. In an analysis, we showed that internal and external attention provide a better focus on relevant input words or phrases than pooling, with external attention having the clearest focus. We publish the source code of the CNNs and RNN-GRUs with the different attention mechanisms at `http://cistern.cis.lmu.de/attentionUncertainty`.

Finally, we added an uncertainty detection module to the slot filling system and showed that it improves results, especially precision. To the best of our knowledge, we are the first to use such a module in the slot filling pipeline.

Chapter 5

Type-Aware Relation Extraction

Erklärung nach §8 Absatz 5 der Promotionsordnung: This chapter covers work published at two peer-reviewed international conferences, namely (Yaghoobzadeh et al., 2017) at European Chapter of the Association for Computational Linguistics (EACL) in 2017 and (Adel and Schütze, 2017b) at Empirical Methods in Natural Language Processing (EMNLP) in 2017.

For a declaration of co-authorship and attribution, see page xxv and following.

5.1 Task and Motivation

As motivated in Chapter 3, relation extraction is a key component of slot filling or knowledge base population in general. In order to extract a triple like (`Bill Gates`, `live_in`, `Medina`) from text, a relation extraction model needs to determine that the `live_in` relation holds in a given context between the two entities `Bill Gates` and `Medina`. While we have presented various relation extraction models in Section 3.3, we now argue that knowledge about the types of the involved entities can improve their results. Thus, in this chapter, we consider type-aware relation extraction, i.e., joint entity classification (EC) and relation extraction (RE).

In slot filling pipelines but also in many other NLP systems, the two tasks are treated as a sequential pipeline: First, a named entity recognition tool is applied, and then the relations between named entity pairs are extracted. However, named entity types and relations are mutually dependent (Roth and Yih, 2004; Yao et al., 2010; Singh et al., 2013a; Li and Ji, 2014). Knowledge of the classes (types) of the relation arguments can help relation extraction, e.g., by reducing the search space of possible relations. Moreover, results from relation extraction can guide entity classification, for instance, in the case of ambiguities. The mention `Medina`, for example, can refer to, i.a., a person, a location, a music album or a board game, depending on its context. If the model can extract from the given context that it is the second argument for the relation `live_in`, it can conclude that the given entity is most likely a location.

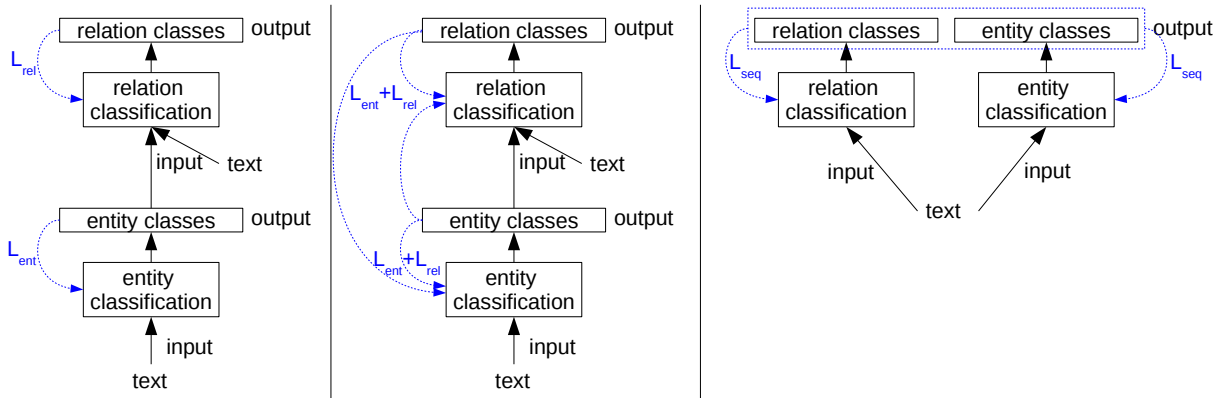


Figure 5.1: Type-aware relation extraction models. The blue dashed lines indicate the propagated loss. Left: pipeline model, middle: jointly trained model, right: globally normalized model with structured prediction output layer. $\mathcal{L}_{ent}$, $\mathcal{L}_{rel}$ and $\mathcal{L}_{seq}$ denote the loss of entity classification, relation classification and the loss of the sequence of entity and relation predictions, respectively.

Therefore, we develop type-aware relation extraction neural networks, i.e., single neural networks for both tasks as depicted in Figure 5.1. In the first part of this chapter (Section 5.2), we propose a network which is trained jointly on (fine-grained) entity and relation classification and uses the entity classification results as additional input features for relation classification (cf., middle part of Figure 5.1). We compare this model against traditional pipeline approaches, which first evaluate EC and then RE (left part of Figure 5.1). In the second part (Section 5.3), we propose a network with a structured prediction output layer, which optimizes a sequence consisting of the (coarse-grained) types of two candidate entities and the relation between them (right part of Figure 5.1). Both presented approaches consider EC and RE together in a single model. Their main difference is two-fold:

(i) In the jointly trained model, the integration of the entity type classes is below the output level of the relation classes, thus, the entity type classes form an additional input to the relation classification model. With this, we expect relation classification to benefit from entity classification. On the other hand, entity classification cannot directly benefit from relation classification. In the model with structured prediction, in contrast, the classification of entity and relation classes is at the same level, thus they directly share the underlying layers for computing the context and entity representations. Given our design of the output sequence, the relation prediction is directly influenced by the predicted class of the first entity and the classification of the second entity is directly influenced by the predicted relation.

(ii) The objective functions handle the combination differently: While the objective function of the jointly trained model is a weighted average of the two task-specific loss

functions, the objective function of the model with structured prediction handles the combination more explicitly and optimizes a sequence of entity and relation classes.

For our experiments, we cannot use the datasets described in Section 3.4.2 and Section 3.4.3 because they do not provide entity class annotations. Therefore, we create a new distantly supervised dataset for training and evaluating the joint model based on automatic entity linking annotations of ClueWeb. These annotations have also been used for entity classification by Yaghoobzadeh and Schütze (2015). The dataset is described in detail in Section 5.2.2. Note that ClueWeb is also one of the datasets we use in Section 3.4.2. For the structured prediction model, we use a manually labeled dataset instead (see Section 5.3.2). This dataset is of a smaller scale than the distantly supervised datasets but does not include noisy labels. Moreover, it provides the possibility of investigating a fine-grained table filling evaluation scenario, which would be considerably more difficult with noisy labels. In order to directly compare the two models for joint entity and relation classification, we finally evaluate them on the same slot filling datasets in Section 5.4.

5.2 Jointly Trained Model

In this section, we describe a relation classification model which uses entity classes as additional input features. Both the entity and the relation classification model are neural networks which can be trained jointly using a weighted sum of the task-specific costs.

5.2.1 Model

We design a neural network based on `contextCNN`, which has been introduced in Section 3.3.4, for the joint entity and relation classification task.

Task-individual Model Parts

Entity Classification Part. For EC, we split the context into two parts (left and right of the entity) and apply a convolutional layer to each of them. Note that the two parts do not contain the entity itself, forcing the model to focus on the context only in order to generalize better to unseen entities in the test set. After convolution, we use max pooling to get context representations $\boldsymbol{u}_{e_1} \in \mathbb{R}^{m_{EC}}$ and $\boldsymbol{u}_{e_2} \in \mathbb{R}^{m_{EC}}$ for the two entities e_1 and e_2. The context representations are then fed into a fully-connected hidden layer and a sigmoid layer to identify the classes of the entities. We use a sigmoid instead of a softmax layer to model that an entity can have several of the fine-grained FIGER classes (Ling and Weld, 2012) we use in this study, such as `POLITICIAN` and `PERSON`.

Relation Classification Part. For RE, we apply `contextCNN` as proposed in Section 3.3.4. Thus, we split the input into three contexts (left, middle and right). To give the model information about the relation arguments, we explicitly model them with entity embeddings (as described in Section 5.2.2). The entity embeddings are included into the contexts, i.e., the left context also includes the first entity, the right context includes the

second entity and the middle context includes both entities. Thus, the contexts "overlap" (cf., Figure 5.2). After 3-max pooling, we concatenate the three context representations to a sentence representation $\boldsymbol{u} \in \mathbb{R}^{3 \cdot 3 \cdot m_{RE}}$ for relation classification. Since the dataset we use for our experiments (see Section 5.2.2) does not include inverse relations, we do not include a flag for the order of the relation arguments to the sentence representation. The sentence representation is then fed into a fully-connected hidden layer, followed by a softmax layer to identify the relation between the two entities. By using the softmax, we assume that only one relation holds between two entities. This is not generally true but a valid assumption for most of the relations considered in our dataset (see Table 5.1).

In this study, we use distantly supervised data without cleaning or selecting the data as in Section 3.4.2. Therefore, we apply multi-instance learning for relation classification. In particular, we use the loss function proposed by Zeng et al. (2015). Given a pair of entities (e_i, e_j), it computes the probabilities for the different relations r based on all available contexts $c \in C$. Then it selects the highest probability for each relation to compute the loss of the network. Thus, the loss function is as follows:

$$\mathcal{L} = -\mathbb{E}_{((e_i,e_j),r)\sim p_{data}} \log P_\theta(r|e_i, e_j) \tag{5.1}$$

$$P_\theta(r|e_i, e_j) = \max_{c \in C} P_\theta(r|e_i, e_j, c) \tag{5.2}$$

Joining the Model Parts

For the convolutional and hidden layers of the entity and relation classification parts as described above, we use different parameters. This is intentional to investigate the impact of joint training vs. pipeline models. With the pipeline models, the entity classification part is trained first. Then, its predictions are used as additional input features for the last hidden layer of the relation classification model. With the jointly trained model, we use the same combined architecture but train both model parts at the same time. Thus, in both cases the input to the last hidden layer of the relation classification model becomes:

$$\boldsymbol{u'} = [\boldsymbol{u}; \boldsymbol{t}_{e_1}; \boldsymbol{t}_{e_2}] \tag{5.3}$$

with $\boldsymbol{t}_{e_k} \in \mathbb{R}^{H_T}$ being the entity type representation of entity e_k with dimensionality H_T. After applying the fully-connected hidden layer of size H_R, the context representation $\boldsymbol{h} \in \mathbb{R}^{H_R}$ is:

$$\boldsymbol{h} = \tanh(\boldsymbol{W}^r \boldsymbol{u'} + \boldsymbol{b}^r) \tag{5.4}$$

where $\boldsymbol{W}^r \in \mathbb{R}^{H_R \times (3 \cdot 3 \cdot m_{RE} + 2 \cdot H_T)}$ is a weight matrix and $\boldsymbol{b}^r \in \mathbb{R}^{H_R}$ a bias term. The models we describe in the following paragraphs only differ in their computation of $\boldsymbol{t}_{e_k}$. The general equation for $\boldsymbol{t}_{e_k}$ is:

$$\boldsymbol{t}_{e_k} = f(\boldsymbol{W}^t[P(t_1|e_k, c); \ldots; P(t_T|e_k, c)]) \tag{5.5}$$

with c being the context of e_k. The different approaches we propose in the following apply different choices for f and $\boldsymbol{W}^t$.

Pipeline Approaches.

- `PREDICTED-HIDDEN`: f is tanh and $\boldsymbol{W}^t \in \mathbb{R}^{H_T \times T}$ is a weight matrix learned during training. T is the number of types. Thus, a fully-connected hidden layer learns representations $\boldsymbol{t}_{e_k}$ based on the predictions $[P(t_1|e_k, c); \ldots; P(t_T|e_k, c)]$.
- `BINARY-HIDDEN`: f and $\boldsymbol{W}^t$ are defined as in `PREDICTED-HIDDEN` but the prediction probabilities $P(t_j|e_k, c), 1 \leq j \leq T$ are binarized:

$$B(t_j|e_k, c) = \begin{cases} 1, & P(t_j|e_k, c) \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

- `BINARY`: $\boldsymbol{t}_{e_k}$ is the binary vector itself, i.e., $B(t_j|e_k, c), 1 \leq j \leq T$, as in Equation 5.6. Thus, $H_T = T$, $\boldsymbol{W}^t \in \mathbb{R}^{T \times T}$ is the identity matrix and f is the identity function.

  Using entity classes as binary information is a standard approach in related work (see Section 5.5). Therefore, we use it as a baseline approach here.
- `WEIGHTED`: The columns of $\boldsymbol{W}^t \in \mathbb{R}^{H_T \times T}$ are distributional embeddings of types (with dimensionality H_T) trained on our corpus (see Section 5.2.2). They are pre-trained and stay fixed during classifier training. Thus, the `WEIGHTED` model computes a weighted average of entity class embeddings with entity class probabilities as weights. f is the identity function.

Joint Training. For jointly training the entity and relation classification model (`JOINT`), we use the architecture from `PREDICTED-HIDDEN` with the key difference that $P(t|e_k, c)$ and $P(r|e_i, e_j, c)$ are learned jointly. The objective function of `JOINT` is a weighted average of the task-specific loss functions:

$$\mathcal{L} = \mathcal{L}_{T_1} + \mathcal{L}_{T_2} + \gamma \cdot \mathcal{L}_R \tag{5.7}$$

with $\mathcal{L}_R$ being the loss of relation classification and $\mathcal{L}_{T_i}, i \in \{1, 2\}$, the loss of entity classification. For the entity classification part with its sigmoid output layer, we use the binary cross entropy between the predictions and the distantly supervised labels as in (Yaghoobzadeh and Schütze, 2017). For relation classification, we use the standard cross-entropy loss (see Section 2.2.4) based on Equation 5.2. The combination weight γ is tuned on the development set. As described before, we integrate multi-instance learning to the loss function (Zeng et al., 2015) to alleviate the distant supervision assumption.

Figure 5.2 depicts the resulting network for joint training. The source code is provided at `http://cistern.cis.lmu.de/noise-mitigation`.

5.2.2 Dataset and Evaluation Measure

For training the entity classification models, we use CF-FIGMENT,[1] a dataset based on distant supervision with Freebase and a version of ClueWeb[2] in which Freebase entities

[1] `http://cistern.cis.lmu.de/figment`.
[2] `http://lemurproject.org/clueweb12`.

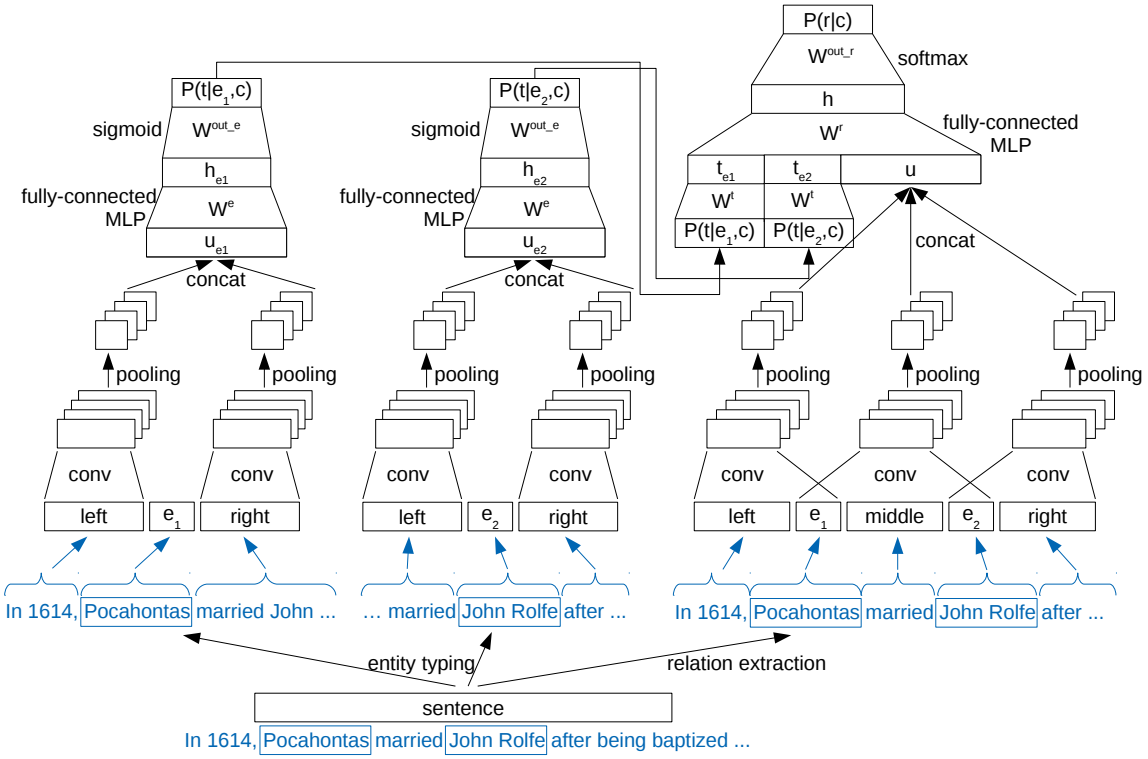


Figure 5.2: Architecture for joint entity typing and relation extraction.

`GOV.GOV_agency.jurisdiction`	`PPL.PER.children`
`GOV.us_president.vice_president`	`PPL.PER.nationality`
`PPL.deceased_PER.place_of_death`	`PPL.PER.religion`
`ORG.ORG.place_founded`	`PPL.PER.place_of_birth`
`ORG.ORG_founder.ORGs_founded`	`N` (no relation)
`LOC.LOC.containedby`	

Table 5.1: Selected relations for relation extraction; PPL = people, GOV = government.

are annotated using FACC1[3] (Gabrilovich et al., 2013). CF-FIGMENT contains 200,000 Freebase entities mapped to 102 FIGER types (Ling and Weld, 2012). Yaghoobzadeh and Schütze (2015) have published that mapping as well as a split into train (50%), dev (20%) and test (30%). When assigning contexts to the entities in those different sets, it is ensured that entities from the test set do not occur in the sentences of the train or dev set. Thus, a sentence from the train set includes only train entities. The resulting dataset is subsampled for each type in train and each entity in dev/test and contains 4,300,000 sentences (contexts).

For training the relation extraction part, we select the ten most frequent relations (plus `N` for no relation according to Freebase) of the entity pairs occurring in CF-FIGMENT with at least one context. This results in 5815, 3054 and 6889 unique entity pairs for

[3] `http://lemurproject.org/clueweb12/FACC1`.

train, dev and test using the same splits as in CF-FIGMENT.[4] This results in a dev set of 124,462 and a test set of 556,847 instances. As the train set, we take a subsample of 135,171 sentences.

Note that when training the entity classification models for the pipeline approaches, we use the whole CF-FIGMENT training set while for training the joint model, we only use the derived relation extraction dataset. Thus, the entity classification part of the joint model is trained with less examples than the entity classification part of the pipeline models. The larger training set is, in fact, an advantage for the pipeline models and increases the challenge of the joint model when evaluating all of them on the same test data.

Embeddings

Motivated by the challenge of domain mismatch for relation classification (see Section 3.5.2), we do not use Wikipedia embeddings for our experiments on ClueWeb data but embeddings trained on ClueWeb by Yaghoobzadeh and Schütze (2015) and Yaghoobzadeh and Schütze (2017). In particular, they use the FACC1 annotations (Gabrilovich et al., 2013) to extend ClueWeb as follows: Each sentence is replaced with three variants: (i) the sentence itself, (ii) the sentence with entities replaced by their Freebase ids, and (iii) the sentence with (train and dev) entities replaced by their notable type in Freebase. Then, `word2vec` (Mikolov et al., 2013) is applied to train embeddings on the extended corpus. We use the resulting word embeddings to represent the context words in our models, the entity embeddings as extended input for our relation classification models (as described in Section 5.2.1) and the entity type embeddings in our `WEIGHTED` pipeline architecture.

Evaluation with Precision-Recall Curves

The evaluation setup of this section is similar to general relation extraction. Therefore, we follow related work on relation extraction and use precision-recall (PR) curves for evaluating our models. To create a precision-recall curve, the model outputs are sorted in descending order according to the confidence of the model and then precision and recall values are computed for each position in the sorted list of results. For example, to compute precision and recall for the fifth item of the list, only the results of the five items with highest model confidence are considered. The resulting precision-recall pairs are then plotted. In order to compare results not only visually but also quantitatively by a number, we follow Ritter et al. (2013) and compute the area A under the PR curve as an additional evaluation measure.

Note that our PR curves are calculated on the corpus level and not on the sentence level, i.e., after aggregating the predictions for each entity pair following Zeng et al. (2015).

[4] We only assign those entity pairs to test (resp. dev, resp. train) for which both constituting entities are in the CF-FIGMENT test (resp. dev, resp. train) set.

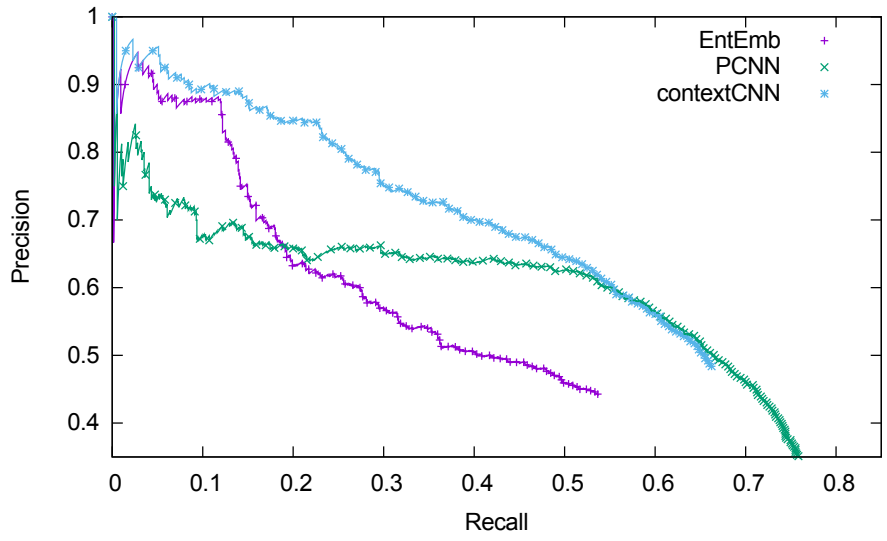


Figure 5.3: PR curves: relation extraction models.

5.2.3 Results

Comparison with Baselines

First, we again compare `contextCNN` (see Section 3.3.4) against the state-of-the-art piecewise CNN (`PCNN`) (Zeng et al., 2015), similar to Section 3.5.2. In contrast to before, we use multi-instance learning for both models with the loss function proposed by Zeng et al. (2015) to mitigate the noise from distant supervision. Another baseline we use is a model that does not learn context features but only uses the embeddings of the relation arguments (as described in Section 5.2.2) as input for a fully-connected feed-forward layer (`EntEmb`). The precision-recall (PR) curves in Figure 5.3 show that `contextCNN` outperforms all baseline models. The results of the `EntEmb` baseline model show that the pre-trained entity embeddings contain information relevant for relation extraction. However, adding contextual information (with the `PCNN` or the `contextCNN`) clearly improves the performance, especially the recall. While the `PCNN` model has a better recall than the `contextCNN` when evaluated on the whole dataset, the curve of the `contextCNN` is much more smooth and has a considerably higher precision, especially in the first part of the curve. The areas under the PR curves confirm these performance differences: `EntEmb`: $A = 0.34$, `PCNN`: $A = 0.48$, `contextCNN`: $A = 0.49$. The area of `PCNN` is close to the area of `contextCNN` because of the higher overall recall of the first.

Type-aware Relation Extraction Results

For the type-aware relation extraction models, we use `contextCNN` and integrate the entity class representations as described in Section 5.2.1. Figure 5.4 shows that the relation extraction performance increases when entity class information is integrated. The main

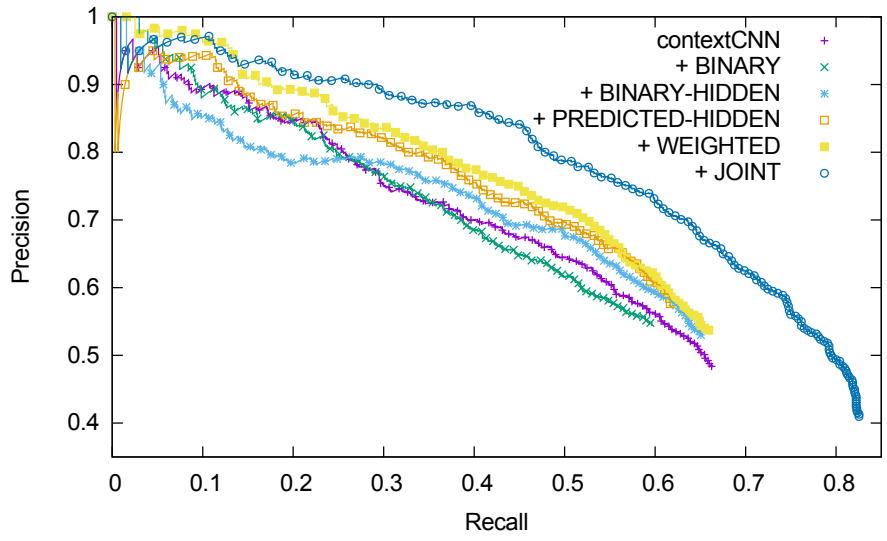


Figure 5.4: PR curves: type-aware relation extraction models.

trend of the PR curves and the areas under them show the following order of model performances: `JOINT` ($A = 0.66$) > `WEIGHTED` ($A = 0.53$) > `PREDICTED-HIDDEN` ($A = 0.49$) > `BINARY-HIDDEN` ($A = 0.49$) > `BINARY` ($A = 0.46$).

`BINARY` and `BINARY-HIDDEN` can be seen as baseline entity class integration models since adding entity classes as binary information is a standard approach in related work (e.g., (Ling and Weld, 2012), see also related work in Section 5.5). However, the performance increases when using the predictions directly (without binarization), as in `PREDICTED-HIDDEN`. Although the areas under the PR curves of `BINARY-HIDDEN` and `PREDICTED-HIDDEN` are almost the same, the actual curves show that `BINARY-HIDDEN` only outperforms `PREDICTED-HIDDEN` for very low recall values. When considering the remaining curve, `PREDICTED-HIDDEN` is clearly superior to `BINARY-HIDDEN`. This suggests that probabilistic predictions of an entity classification system can be a valuable resource for relation extraction. We assume that the main reason is information loss due to binarization: With binary types, it is not possible to tell whether one of the selected types had a higher probability than another or whether a type whose binary value is 0 just barely missed the threshold. Probabilistic representations, on the other hand, preserve this information. Thus, by using probabilistic representations, the relation extraction model can learn to compensate possible noise in entity class predictions.

`WEIGHTED` has access to entity class embeddings learned from a large corpus in an unsupervised way. Integrating a weighted average of those embeddings, i.e., combining the entity class predictions with another resource works better than learning embedding-like weights during training (as in `PREDICTED-HIDDEN`).

Joint training, finally, performs better than all pipeline models. This result is even stronger since the entity classification part of the pipelines has been trained on more data than the joint model (see Section 5.2.2). A possible reason is the mutual dependency of

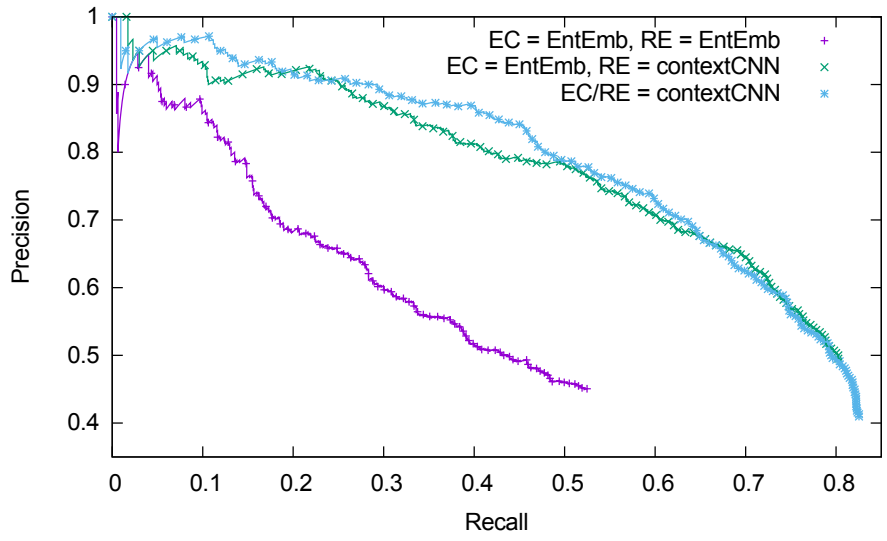


Figure 5.5: Variants of joint training.

the two tasks from which a joint model can benefit. Note that we do not report entity classification results since we consider it as an additional input to help relation extraction here. However, joint training also improves entity classification: On the joint dataset, the mean average precision scores for entity classification increase by about 20%. Another possibility of jointly modeling EC and RE is described in Section 5.3.

5.2.4 Analysis

In the following analyses, we investigate the impact of context modeling on joint training and present relation-wise results of the different models.

Impact of Context Modeling on Joint Training

In this paragraph, we investigate joint training in more detail. In particular, we are interested in the impact of the context representations on the final performance. In Figure 5.3, we have already shown that relation extraction benefits from contextual information. In this section, we compare the results when using `EntEmb` as entity classification model to convolving the contexts for entity classification.

Figure 5.5 shows that the results of combining entity classification with `EntEmb` and relation classification with `EntEmb` are the worst. This emphasizes the importance of modeling the context for relation classification (cf., Figure 5.3), even in combination with joint training. The performance of using context representations from convolutional layers for relation extraction but only entity embeddings for entity classification comes close but is still worse than the performance of the model presented in Section 5.2.1, which uses convolution for both model parts.

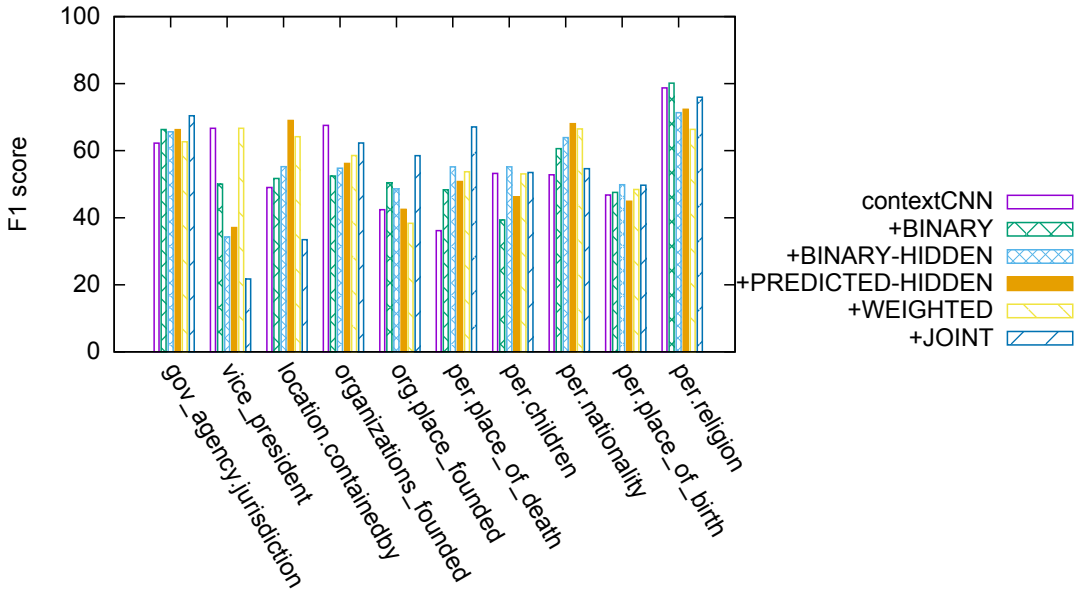
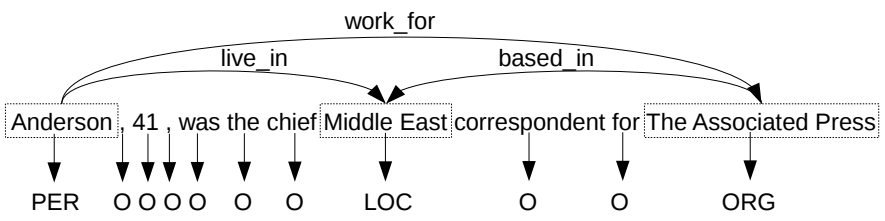


Figure 5.6: Relation-wise comparison of type-aware models.

Thus, it is important which models are trained together in joint training. The areas under the PR curves show the following model trends: `JOINT` original with EC/RE=`contextCNN` ($A = 0.66$) > `JOINT` with EC=`EntEmb` and RE=`contextCNN` ($A = 0.64$) > `JOINT` with EC=`EntEmb` and RE=`EntEmb` ($A = 0.35$).

Relation-wise Results

We identify the relations for which the performance of the model is improved the most when entity class information is added. For this, we compare the relation specific F1 scores of `contextCNN` with the scores of `WEIGHTED` (the best pipeline model) and `JOINT`. With `WEIGHTED`, the relations `PPL.deceased_PER.place_of_death` and `LOC.LOC.containedby` are improved the most (from 36.13 to 53.73 and 49.04 to 64.19 F1, resp.). `JOINT` has the most positive impact on `PPL.deceased_PER.place_of_death`, `GOV.GOV_agency.jurisdiction` and `ORG.ORG.place_founded` (those relations are improved from 36.13 to 67.10, 62.26 to 70.41 and 42.38 to 58.51, resp.). Figure 5.6 gives a graphical overview of a relation-wise comparison of `contextCNN`, the different pipeline models and the jointly trained model. The corresponding numbers are provided in the appendix in Table C.2 and Table C.3. The analysis shows that the improvements by joint training are not consistent across relations and again confirms that different relations pose different challenges to the models. Recall that the performance trends of different models on the slot filling relations have not been consistent, either (see Table 3.6). It might also be a side effect of training the entity classification part of the joint model on less data than in the pipeline setting, which might



Model inputs (query entity pairs) ⇒ Model outputs:
(“Anderson”, “,”) ⇒ PER – N – O
(“Anderson”, “41”) ⇒ PER – N – O
...
(“Anderson”, “chief”) ⇒ PER – N – O
(“Anderson”, “Middle East”) ⇒ PER – live_in – LOC
...
(“was”, “for”) ⇒ O – N – O
...
(“for”, “The Associated Press”) ⇒ O – N – ORG

Figure 5.7: Examples of the task and model inputs/outputs.

result in more confident entity class predictions within the pipeline models.

5.3 Neural Structured Prediction Model

Most studies on joint training and multitask learning optimize a weighted average of task-wise costs. In Section 5.2, we have proposed and investigated a joint model for entity and relation classification, which follows this line of research. In this section, we introduce a joint classification layer based on structured prediction, which optimizes a sequence of outputs from both tasks instead. Specifically, this model treats both entity and relation classes as output classes instead of using the entity classification results as an additional input for relation classification, as presented in Section 5.2. Thus, there is no hierarchy between entity and relation classification and both tasks can benefit from each other by design.

5.3.1 Model

First, we identify candidate entities (either by prior knowledge about entity boundaries or by treating each token as a potential entity) and then predict their classes (coarse-grained types) as well as the relation between them by creating a length-three prediction sequence: class of the first entity, relation between the entities, class of the second entity.

Figure 5.7 shows an example of how we model the task: Each sentence can contain

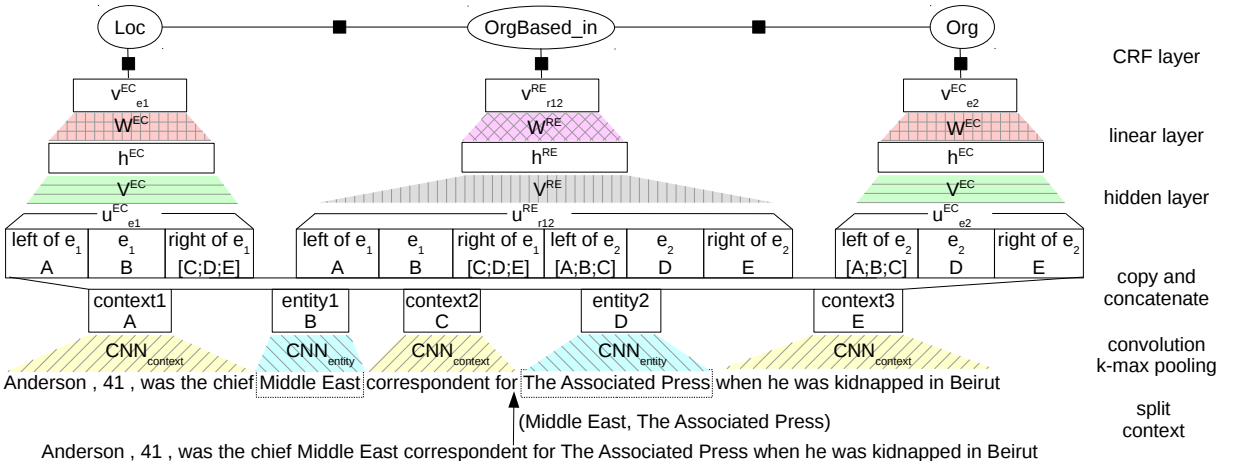


Figure 5.8: Model overview; the colors/shades show which model parts share parameters.

multiple named entities as well as multiple relations between them. In order to identify all possible entity classes and relations, we compute the set of all possible combinations of candidate entities. Each of those combinations (called query entity pair in the remainder of this section) then forms an input to our model, which predicts the output sequence of entity and relation classes.

Figure 5.8 illustrates our model. It is based on CNNs for creating context and entity representations and a CRF (Lafferty et al., 2001) output layer which finds the best output sequence of entity and relation classes. Its source code is available at `http://cistern.cis.lmu.de/globalNormalization`.

Sentence Representation

As shown in Figure 5.7, the inputs to our model are the sentence and two query entities (tokens or phrases of the sentence) for which the classes and relation should be identified. The tokens of the input sentence are represented by word embeddings. In this study, we use the Wikipedia `word2vec` embeddings again (see Section 2.2.3). The sentence is then split into different parts at the positions of the query entities, building on our positive results with the `contextCNN` described in Section 3.3.4 and Section 5.2.1. For identifying the class of an entity e_k, the model uses three parts: the context left of e_k, the words constituting e_k and the context to its right. For classifying the relation between two query entities e_i and e_j, the sentence is split into six parts: left of e_i, e_i, right of e_i, left of e_j, e_j, right of e_j. For the example sentence in Figure 5.7 and the query entity pair ("Anderson", "chief"), the context split is: [] [Anderson] [, 41 , was the chief Middle ...] [Anderson , 41 , was the] [chief] [Middle East correspondent for ...] Note that the dataset we use (see Section 5.3.2) provides boundaries for entities to concentrate on the classification task (Roth and Yih, 2004). In two of our evaluation setups, which are described in Section 5.3.3, we use these boundaries, thus, the query entities can be constituted by several tokens. In our third

evaluation setup, we assume no prior knowledge about the boundaries and, therefore, all query entities are single tokens.

For representing the different parts of the input sentence, we use convolutional neural networks. In particular, we train one CNN layer for convolving the entities and one for the contexts. Using two CNN layers instead of one gives our model more flexibility. Since entities are usually shorter than contexts, the filter width for entities can be smaller than for contexts. Furthermore, Yaghoobzadeh and Schütze (2017) show that character-based entity representations add useful information to the entity classification task. Our architecture simplifies changing the entity representation from words to characters in future work.

After convolution, we apply k-max pooling for both the entities and the contexts and concatenate the results. The concatenated vector $\boldsymbol{u}^z \in \mathbb{R}^{U_z}$ with $z \in \{EC, RE\}$ is forwarded to a task-specific hidden layer of size H_z, which can learn patterns across the different input parts:

$$\boldsymbol{h}^z = \tanh(\boldsymbol{V}^z \boldsymbol{u}^z + \boldsymbol{b}^z) \tag{5.8}$$

with weights $\boldsymbol{V}^z \in \mathbb{R}^{H_z \times U_z}$ and bias $\boldsymbol{b}^z \in \mathbb{R}^{H_z}$.

To compute scores $\boldsymbol{v}^z$ for the different entity or relation classes, we apply a linear mapping from the context representations $\boldsymbol{h}^z \in \mathbb{R}^{H_z}$ as follows:

$$\boldsymbol{v}^z = \boldsymbol{W}^z \boldsymbol{h}^z \tag{5.9}$$

with $\boldsymbol{W}^z \in \mathbb{R}^{(N_{EC}+N_{RE}) \times H_z}$ and N_{EC} and N_{RE} being the number of entity and relation classes, respectively.

Structured Prediction Layer

We propose to model the joint entity and relation classification task with the following sequence of scores:

$$\boldsymbol{q} = [\boldsymbol{v}^{EC}_{e_1}; \boldsymbol{v}^{RE}_{r_{12}}; \boldsymbol{v}^{EC}_{e_2}] \tag{5.10}$$

with $\boldsymbol{v}^{EC}_{e_k}, k \in 1, 2$ being the score for the class of entity e_k and $\boldsymbol{v}^{RE}_{r_{ij}}$ being the score for the relation between e_i und e_j. This is also shown in Figure 5.8. We use a linear-chain CRF layer to model this sequence, thus, we approximate the joint probability of entity types T_{e_1}, T_{e_2} and relation $R_{e_1e_2}$ as follows:

$$\begin{aligned} &P(T_{e_1}, R_{e_1e_2}, T_{e_2}) \\ \approx &P(T_{e_1}) \cdot P(R_{e_1e_2}|T_{e_1}) \cdot P(T_{e_2}|R_{e_1e_2}) \end{aligned} \tag{5.11}$$

Our intuition is that the dependency between relation and entities is stronger than the dependency between the two entities and that, therefore, this approximation makes sense. To the best of our knowledge, this is the first work to model the joint entity and relation classification task with a sequence of predictions. Before, CRF layers for neural networks have been mainly used for token-labeling tasks like NER or part-of-speech tagging (Collobert et al., 2011; Andor et al., 2016; Lample et al., 2016).

For implementation, we adopt the linear-chain CRF layer by Lample et al. (2016).[5] The input sequence from Equation 5.10 is padded with begin and end tags to a sequence of length $n + 2$ ($n = 3$ in our case) and fed into a CRF layer as defined in Equation 2.28 and Equation 2.29 of Section 2.2.3. The matrix of transition scores $\boldsymbol{T} \in \mathbb{R}^{(N+2)\times(N+2)}$ with $N = N_{EC} + N_{RE}$ is learned during training. In order to compute the probability of a label sequence as in Equation 2.29, all possible label sequences need to be calculated. To avoid redundant computations, the forward algorithm, a dynamic programming technique, is applied for this (Rabiner, 1989). At each step of the label sequence, it sums over the scores of all possible previous label sub-sequences. When predicting the best label sequence (without its probability) during testing, the Viterbi algorithm is used (Rabiner, 1989). It uses a similar dynamic programming technique as the forward algorithm but computes and stores the maximum scores (instead of the sum) of the previous label sub-sequences. Even for length-three sequences, dynamic programming reduces the number of necessary computations considerably. For longer sequences, as they might be necessary for other applications, computations without this technique might not be feasible.

Relationship to Other Joint Models

The components of the model described in this section are similar to the components of the jointly trained model described in Section 5.2.1 (context splitting, convolution and pooling, etc.). However, since the outputs of entity and relation classification are designed to be on the same level now, the entity classes cannot be used as features for relation classification any more. While the joint model in Section 5.2.1 consists of two different model parts (one for entity and one for relation classification) which interact only through the entity type features for relation classification, the model in this section aims at exchanging information between the two tasks already at the level of context representation. Therefore, the sentence representation layer is slightly changed with the goal of sharing as much parameters as possible between the two tasks.

Instead of applying a structured prediction output layer on the sequence of entity and relation classes, joint inference would also be possible by using, for example, an RNN decoder network with a local softmax layer for each item of the output sequence. However, this would introduce additional parameters depending on the size of the RNN hidden layer. Moreover, it would be prone to the label bias problem, a challenge which is solved by global normalization in CRFs (Lafferty et al., 2001; Andor et al., 2016).

5.3.2 Dataset and Evaluation Measure

The "entity and relation recognition" (ERR) dataset from (Roth and Yih, 2004)[6] provides 5925 sentences from newspaper articles from TREC,[7] which have been manually annotated with entity classes (`Peop`, `Org`, `Loc`, `Other`, `O`) and relations between entities (`Located_in`,

[5] `https://github.com/glample/tagger`.

[6] `https://cogcomp.org/page/resource_view/43`.

[7] `http://trec.nist.gov`.

Work_for, OrgBased_in, Live_in, Kill). It, therefore, provides an interesting evaluation option for our models which is different and complementary to the automatically created (Section 3.4.3) and distantly supervised datasets (Section 3.4.2, Section 5.2.2) we use in the other chapters of this thesis.

Following previous work, we focus on the 1441 sentences of the ERR dataset with at least one relation between entities. We use the train-test split published by Gupta et al. (2016) to be able to compare with their results. For parameter tuning, we further split train into a core training set and a dev set. Table 5.2 provides statistics of the data composition in our different setups, which are described in Section 5.3.3. Similar to the O class for candidate entities which do not belong to one of the named entity classes, we use the label N for entity pairs without one of the pre-defined relations. Note that we subsample the N class in the training and development set of setup 2 and setup 3 to speed up training and to avoid that the model learns to only predict this dominant class. The three evaluation setups are described in Section 5.3.3.

	train	dev	test
# sentences	911	242	288
Peop	1146	224	321
Org	596	189	198
Loc	1204	335	427
Other	427	110	125
O	20338	5261	6313
Located_in	243	66	94
Work_for	243	82	76
OrgBased_in	239	106	105
Live_in	342	79	100
Kill	203	18	47
N (setup 1)	10742	2614	3344
N (setup 2/3)	123453	30757	120716

Table 5.2: Dataset statistics. The number of N differs for our different experimental setups.

For evaluation, we follow previous work on the ERR dataset and compute F1 of the individual classes for EC and RE, as well as a task-wise macro F1 score. We also report the average of scores across tasks (Avg EC+RE).

5.3.3 Experimental Setups

In this section, we describe the three setups we consider in our experiments. The first and last setups follow two different lines of related work on the ERR dataset. The second setup is designed to provide an intermediate step and better analyze the additional difficulties setup 3 poses in contrast to setup 1.

Setup 1: Entity Pair Relations. In the first setup, relation classification is applied only to named entity pairs. Moreover, the entity boundary information is used for entity and relation classification. This setup is used by, e.g., Roth and Yih (2004), Roth and Yih (2007) and Kate and Mooney (2010). Note that when we only input named entity pairs

Anderson	Peop	N	N	N	N	N	N	live	N	N	work
,		O	N	N	N	N	N	N	N	N	N
41			O	N	N	N	N	N	N	N	N
,				O	N	N	N	N	N	N	N
was					O	N	N	N	N	N	N
the						O	N	N	N	N	N
chief							O	N	N	N	N
Middle East								Loc	N	N	based
correspondent									O	N	N
for										O	N
The Associated Press											Org
	Anderson	,	41	,	was	the	chief	Middle East	correspondent	for	The Associated Press

Figure 5.9: Entity-relation table.

as query entity pairs into our model, the entity classification part is simplified because the model can learn to omit the `O` class. This is different to related work which regard entity and relation classification in different models and use them in a pipeline or combine their results.

Setup 2: Table Filling. Miwa and Sasaki (2014) introduce a novel way to model the joint task of EC and RE: as a table filling problem. For each sentence with length m, a quadratic table is created, as shown in Figure 5.9 for the example sentence from Figure 5.7. The diagonal cells (k, k) contain the classes of the corresponding entities. The non-diagonal cells (i, j) contain the relation between word i and word j (`N` for no relation). Following previous work, we only predict classes for half of the table, i.e., for $m(m+1)/2$ cells. When using this setup with our model, each cell (i, j) forms a separate input query with the query entities being formed by the row i and the column j of the cell. As described in Section 5.3.1, our model predicts not only the relation r_{ij} but also the classes of entities e_i and e_j. Thus, the class of entity e_i, for example, is predicted $m-1$ times from different perspectives. To fill the corresponding entity class cell (i, i), we aggregate the results from all those predictions using majority vote. This is a common strategy for combining different predictions, for example in the context of model ensembling (Breiman, 1996; Vu et al., 2016). In Section 5.3.5, we verify that the individual predictions agree with the majority vote in almost all cases.

Setup 3: Table Filling Without Entity Boundaries. The table from setup 2 includes one row/column per multi-token entity, utilizing the given entity boundaries of the ERR dataset. The table filling setup used in related work (Miwa and Sasaki, 2014; Gupta et al., 2016) does not include this prior information but assigns a table row/column to each token of the input sentence. We follow this approach in our third setup. For evaluation, we follow Gupta et al. (2016) and score a multi-token entity as correct if at least one of its comprising cells has been classified correctly.

	setup 1		setup 2		setup 3	
	softmax	CRF	softmax	CRF	softmax	CRF
Peop	**95.24**	94.95	93.99	**94.47**	91.46	**92.21**
Org	**88.94**	87.56	78.95	**79.37**	67.29	**67.91**
Loc	93.25	**93.63**	90.69	**90.80**	85.99	**86.20**
Other	**90.38**	89.54	73.78	**73.97**	**62.67**	61.19
Avg EC	**91.95**	91.42	84.35	**84.65**	76.85	**76.88**
Located_in	55.03	**57.72**	51.03	**55.13**	44.96	**52.29**
Work_for	**71.23**	70.67	52.89	**61.42**	52.63	**65.31**
OrgBased_in	53.25	**59.38**	56.96	**59.12**	46.15	**57.65**
Live_in	**59.57**	58.94	**64.29**	60.12	**64.09**	61.45
Kill	74.70	**79.55**	69.14	**74.73**	**82.93**	75.86
Avg RE	62.76	**65.25**	58.86	**62.10**	58.15	**62.51**
Avg EC+RE	77.36	**78.33**	71.61	**73.38**	67.50	**69.69**

Table 5.3: F1 results for entity classification and relation extraction in the three setups.

Comparison. There are two important differences between setup 1 and setup 2/3: First, when only entering combinations of named entities in setup 1, our model can basically omit the `O` class of entity classification which simplifies setup 1. Second, the number of entity pairs with no relation (`N`) is significantly different in setup 1 and the table filling setups 2/3: In the test set, there are about 3k entity pairs with no relation in setup 1 while there are about 121k entity pairs with no relation in setup 2/3. This makes the table filling setups 2/3 considerably more challenging. To better cope with this, we randomly subsample negative instances in the training set of setup 2 and 3. Finally, setup 3 is the most challenging setup since it splits multi-token entities into their comprising tokens and, thus, considers the most query entity/word pairs in total. On the other hand, setup 3 is also the most realistic scenario of the three setups since in most cases, entity boundaries cannot be assumed to be given. Setup 1 or 2 can only be applied to datasets without entity boundaries if a pre-processing step, such as entity boundary detection or chunking is applied first. Setup 3, however, is directly applicable to any other dataset. When comparing the results of setup 2 and setup 3, it is possible to investigate the impact of the prior knowledge of entity boundaries on the classification results.

5.3.4 Results

For all three experimental setups, we compare our CNN+CRF model as described in Section 5.3.1 with a CNN-based model that uses softmax layers instead of a CRF layer for entity and relation classification. Thus, it does not learn transition scores between the classes and optimizes the entity and relation classes independent from each other instead of their sequence. For training, the losses of entity classification and relation classification are averaged, as in joint training (see Section 5.2). Table 5.3 shows the results.

The CRF layer performs comparable or better than the softmax layer across experimental setups. Especially for the more challenging table filling setups (setup 2 and 3), the improvements are apparent. When comparing setup 2 and setup 3, it is visible that

model	S	feats	EC	RE	EC+RE
R & Y 2007	1	yes	85.8	58.1	72.0
K & M 2010	1	yes	91.7	62.2	77.0
Ours (CNN+CRF)	1	no	**92.1**	**65.3**	**78.7**
Ours (CNN+CRF)	2	no	**88.2**	**62.1**	**75.2**
M & S 2014	3	yes	92.3	**71.0**	**81.7**
G et al. 2016 (1)	3	yes	**92.4**	69.9	81.2
G et al. 2016 (2)	3	no	**88.8**	58.3	**73.6**
Ours (CNN+CRF)	3	no	82.1	**62.5**	72.3

Table 5.4: Comparison to state of the art (S: setup, feats shows whether the models use additional hand-crafted features).

entity classification suffers when no entity boundaries are given (in setup 3). One reason for that might be that the convolutional layer cannot extract features from multi-token entity surface forms any more since it only gets single tokens as potential entities in setup 3 (context B and D in Figure 5.8). This might be mitigated by using character-level entity representations instead. Interestingly, the relation classification performance is not affected from the missing entity boundaries: The average relation classification results are comparable in setup 2 and setup 3. This shows that the model part for relation classification can internally account for potentially wrong entity classification results due to missing entity boundaries.

To sum up, the CRF layer leads to better overall results (Avg EC+RE) in all three setups. Thus, joint EC and RE benefits from structured prediction and our way of creating the input sequence for the CRF layer for joint EC and RE is effective.

Comparison to State of the Art

In Table 5.4, we set our results in the context of state-of-the-art results: (Roth and Yih, 2007), (Kate and Mooney, 2010), (Miwa and Sasaki, 2014), (Gupta et al., 2016).[8] Since all reported prior results are based on different setups and different train-test splits, they can only give a hint about model rankings but are not directly comparable. Our results are best comparable with (Gupta et al., 2016) since we use the same setup and train-test splits. The model presented in (Gupta et al., 2016) is quite complicated and uses hand-crafted features and various iterations of modeling dependencies among entity and relation classes. In contrast, we only use pre-trained word embeddings and train our model end-to-end with only one iteration for both entity and relation classification per entity pair and epoch. When we compare with their model without additional features (G et al. 2016 (2)), our model performs worse for EC but better for RE and comparable for Avg EC+RE.

[8]We only show results of single models, no ensembles. Following previous studies, we omit the entity class "Other" when computing the EC score.

	N	Based_in	Live_in	Kill	Located_in	Work_for
O						
Other						
Peop						
Org						
Loc						

Table 5.5: Heat map of positive correlations between entity types and relations according to scores in CRF transition matrix: the darker the cell, the higher is the transition score.

5.3.5 Analysis

Analysis of Entity Type Aggregation

As described in Section 5.3.3, we aggregate the EC results by majority vote. Therefore, we analyze the disagreement in order to assess how stable these aggregated results are. For the structured prediction model of setup 2, there are only nine entities (0.12%) with disagreement in the test data. For those, the max, min and median disagreement with the majority label is 36%, 2%, and 8%, respectively. Thus, the disagreement is negligibly small and we can consider the entity classification results stable, i.e., we would get approximately the same results with a different aggregation scheme.

Analysis of CRF Transition Matrix

In this paragraph, we analyze the CRF layer. In contrast to the task-individual softmax layers, it learns transition scores $\boldsymbol{T}$ between the entity and relation classes (cf., Equation 2.28). In Table 5.5, we show which transitions have positive scores, i.e., which entity and relation classes are positively correlated with each other. The color intensity visualizes the value of the score. It is obvious that the layer has learned correct correlations between entity types and relations, such as the relation `Work_for` is a relation between a person (`Peop`) and an organization (`Org`) or a location (`Loc`) can be an argument of the relations `Based_in`, `Live_in` and `Located_in`.

5.4 Application to Slot Filling

In standard slot filling systems, a pipeline approach of entity and relation classification is applied: Based on named entity classes (obtained with a NER tool), possible filler candidates are identified which are then classified by relation classification models. As a result, the inputs of our binary models (see Section 3.5.2) all have named entity types corresponding to the expected types of the slots. The model for the relation `per:date_of_birth`, for example, only classifies sentences with one relation argument being a `PERSON` and the other relation argument being a `DATE`. When preparing our training data, we also ensure this constraint, by only extracting entities with the correct type as negative examples (see Section 3.2.1). When using multiclass models (as in Section 3.5.3), the relation classifica-

tion models do not know about the input types. Although the filler candidate extraction process is still based on named entity classes, the slot filler classification module uses the same multiclass model for all slots. This complicates the relation classification task for the model. A context for the relation `per:date_of_birth`, for example, might be similar to a context for the relation `per:location_of_birth`. Although it is possible to only consider the output probabilities for relations which are consistent with the predicted named entity classes, we do not want to apply such a hard constraint as this would suffer from error propagation. Instead, our approaches of jointly modeling entity and relation classification allow our model to compensate for errors in the named entity classification of the candidate extraction module. While some other systems use (binary) entity types as additional input features for slot filler classification, e.g., (Angeli et al., 2014a; Rahman et al., 2016; Zhang et al., 2016c), we are not aware of a system using neural models for joint entity and relation classification.

5.4.1 Model

In this section, we describe how we integrate jointly trained models (Section 5.2) and neural structured prediction models (Section 5.3) into the slot filling pipeline. For all models, we use the same coarse-grained types we use in the slot filling system: `PERSON, ORGANIZATION, LOCATION, DATE, NUMBER, O`.[9]

Pipeline Approach and Joint Training

First, we investigate two different settings for augmenting the input of the multiclass model with named entity types: a pipeline approach and a jointly trained model. The architecture of the model we use for this is similar to the architecture illustrated in Figure 5.2.

We input the scores for the types (either binary or probabilistic scores) as a vector $\boldsymbol{p}_{e_k} \in \mathbb{R}^T$ of the size of the type vocabulary T and create type embeddings $\boldsymbol{t}_{e_k} \in \mathbb{R}^{H_T}$ with a hidden layer of size H_T:

$$\boldsymbol{t}_{e_k} = \tanh(\boldsymbol{V}\boldsymbol{p}_{e_k} + \boldsymbol{d}) \tag{5.12}$$

with $\boldsymbol{V} \in \mathbb{R}^{H_T \times T}$ being the weight matrix and $\boldsymbol{d} \in \mathbb{R}^{H_T}$ the bias of the hidden layer.

Then, the type embeddings $\boldsymbol{t}_{e_k}$ are concatenated with the three context representations of the slot filling CNN (see Figure 3.8). Thus, the sentence representation $s \in \mathbb{R}^H$ is now computed as follows:

$$s = tanh(\boldsymbol{W}^1\boldsymbol{u} + \boldsymbol{W}^2\boldsymbol{t}_{e_1} + \boldsymbol{W}^3\boldsymbol{t}_{e_2} + \boldsymbol{b}) \tag{5.13}$$

with $\boldsymbol{t}_{e_1}$ being the embedding for the type of the first relation argument and $\boldsymbol{t}_{e_2}$ being the embedding for the type of the second relation argument. Figure 5.10 depicts this.

For obtaining type scores $\boldsymbol{p}_{e_k}$, we evaluate two different settings: `slotNER` and `jointNER`.

[9] We omit `MISC` since there is no slot correlated with that entity class.

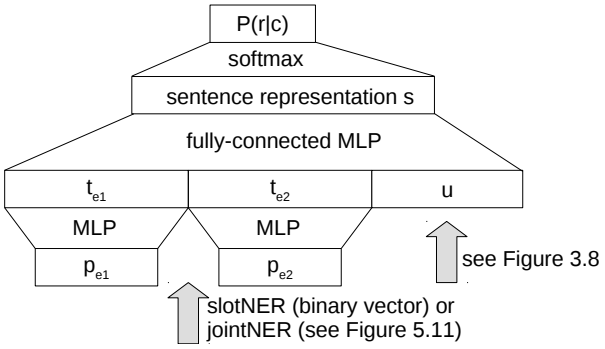


Figure 5.10: Integration of entity type information into multiclass CNN.

In `slotNER`, we create binary type scores based on the slot of the input sentence. For the slot `per:employee_or_member_of`, for example, the type score vector for the first relation argument would consist of only one 1 at the position of `PERSON` and 0 otherwise (like a one-hot vector). The type score vector for the second relation argument would consist of a 1 at the position of `ORGANIZATION` and a 1 at the position of `LOCATION` since a person can be employed by either an organization or a geo-political entity. The model `slotNER`, thus, applies a pipeline by using the predictions of the named entity recognition system in the filler candidate extraction component as features for slot filler classification.

In `jointNER`, we predict probabilities $P(t|e_k, c)$ for the different types using a CNN over the left and right contexts of the relation arguments, similar to the entity classification model from Section 5.2.1. This is depicted in Figure 5.11. We then use the predicted probabilities as type scores $\boldsymbol{p}_{e_k}$. The CNN for entity classification is trained jointly with the CNN for slot filler classification. Similar to Equation 5.7, the objective function is a weighted average of the task-specific losses:

$$\mathcal{L} = (1 - \alpha) \cdot \mathcal{L}_R + \frac{\alpha}{2} \cdot \mathcal{L}_{T_1} + \frac{\alpha}{2} \cdot \mathcal{L}_{T_2} \tag{5.14}$$

The weight α controls the ratio between the relation classification loss and the entity type classification loss and is tuned on dev.

Neural Structured Prediction Model

Second, we describe how we apply the structured prediction output layer to slot filling. Given a sentence with the query entity and the slot filler candidate, we adapt the model structure from Figure 5.8 to the architecture we have developed for slot filling relation classification. In particular, we do not represent the relation arguments but only the three contexts and integrate a flag to the sentence representation for relation classification, which indicates the order of the relation arguments. As a result, the context representation used to classify the first relation argument is the left and middle context: $\boldsymbol{u}_{e_1}^{EC} = [A; C]$ (using the variable names from Figure 5.8), the context representation for classifying the relation is the left, middle and right context as well as the flag: $\boldsymbol{u}_{r_{12}}^{RE} = [A; C; E; v]$ and

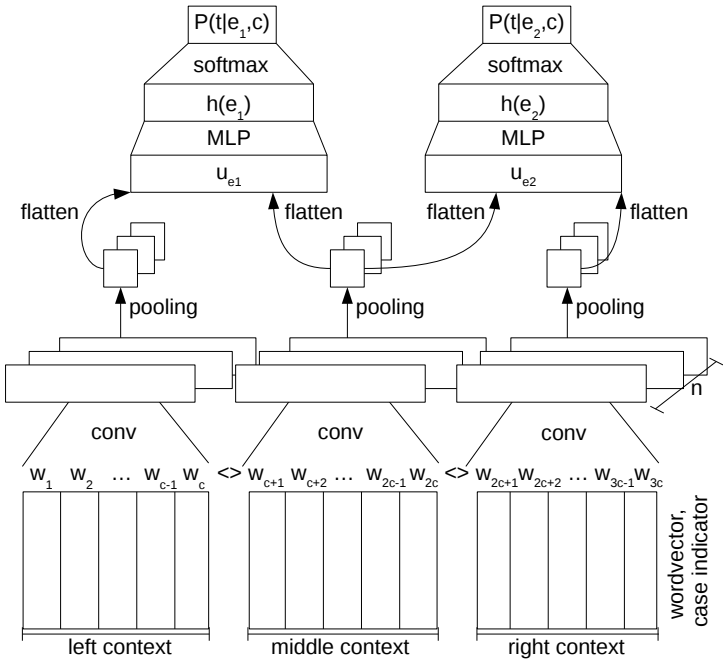


Figure 5.11: Convolutional neural network for entity type classification.

the representation for identifying the class of the second relation argument is the middle and right context: $\boldsymbol{u}_{e_2}^{EC} = [C; E]$. The remaining layers including the structured prediction CRF layer are left unchanged. For computing probabilities for the different relation classes (slots), we apply the forward-backward algorithm (Rabiner, 1989).

5.4.2 Results on Slot Filling Benchmark

Table 5.6 provides slot-wise results for the different CNN setups on the slot filling benchmark dataset for relation classification (see Section 3.4.3): binary CNNs (from Table 3.6), a multiclass CNN without entity type information (`noNER`, cf., Section 3.5.3), a multiclass CNN with slot-based entity types (`slotNER`) following a pipeline approach, a multiclass CNN with entity type probabilities jointly trained with the relation classification CNN (`jointNER`) and a multiclass CNN with a structured prediction output layer, which is globally normalized on joint entity and relation classification (`global`).

Binary CNNs provide the best results, even when using entity type information in the multiclass models. Adding binary entity types based on the slot type does not improve the macro F1 score of the multiclass CNN. Joint training and structured prediction lead to slightly better macro F1 results on the development set. The structured prediction model has a better generalization ability to an unseen evaluation set than the other multiclass models: It outperforms `slotNER` and `jointNER` by 3 F1 points and, therefore, comes closer to the performance of the binary CNN. In terms of micro F1, the pipeline approach `slotNER` provides the best results of the multiclass models but still performs considerably worse than

	binary		multiclass							
	contCNN		noNER		slotNER		jointNER		global	
	dev	test	dev	test	dev	test	dev	test	dev	test
per:age	.83	**.76**	.70	.68	**.84**	.72	.71	.66	.75	.67
per:alternate_names	**.32**	**.04**	.22	.00	.25	.00	.10	.00	.04	.00
per:cause_of_death	**.77**	**.52**	.53	.11	**.77**	.29	.42	.00	.40	.06
per:children	**.82**	**.61**	.71	.48	.70	.44	.69	.44	.75	.36
per:date_of_birth	**1.0**	.77	.93	**.80**	**1.0**	**.80**	.98	.73	.90	.73
per:date_of_death	**.72**	.48	.64	.51	.69	.46	.68	.39	**.72**	**.59**
per:empl_memb_of	.41	**.37**	.37	.28	**.42**	.29	.34	.25	.36	.28
per:location_of_birth	.59	.23	.68	**.36**	.71	.20	**.74**	.34	.71	.35
per:loc_of_death	**.63**	**.28**	.62	**.28**	.54	.19	.60	.25	.61	.21
per:loc_of_residence	**.20**	.23	.06	.15	.04	.06	.15	.22	.16	**.25**
per:origin	**.43**	**.39**	.09	.11	.26	.30	.13	.13	.15	.17
per:schools_att	**.72**	.55	.65	.45	.59	.47	.70	.56	.65	**.68**
per:siblings	**.63**	.70	.58	**.73**	.58	.54	**.63**	.63	.60	.68
per:spouse	.67	.30	.66	.39	.64	**.49**	**.69**	.36	.64	.30
per:title	**.57**	.46	.51	.42	.50	.43	.53	.44	.53	**.48**
org:alternate_names	**.65**	**.66**	.55	.58	.54	.55	.47	.50	.60	.58
org:date_founded	**.64**	.71	.54	.63	.58	.65	.59	**.74**	.63	.69
org:founded_by	**.80**	.68	.62	.71	.34	.43	.70	**.74**	.65	.73
org:loc_of_headqu	**.43**	**.45**	.25	.24	.37	.42	.21	.21	.34	.34
org:members	.65	.04	.64	**.17**	.42	.07	.66	**.17**	**.72**	.11
org:parents	**.41**	**.16**	.37	.14	.17	.09	.38	.12	.32	.10
org:top_memb_empl	.43	.53	.48	.55	.39	.49	**.49**	**.58**	.45	**.58**
micro F1	**.52**	**.43**	.28	.22	.35	.28	.28	.22	.30	.23
macro F1	**.61**	**.45**	.52	.40	.52	.38	.53	.38	.53	.41

Table 5.6: F1 results on slot filling benchmark dataset for different CNN setups. `contCNN` is short for `contextCNN`.

the binary models.

Slots for which entity type information seems to help the most are, e.g., `per:date_of_birth` and `per:location_of_birth`, i.e., two slots with similar contexts. Although the binary CNNs have the best results on dev for almost all the slots, multiclass models with entity class information have a better generalization ability to unseen test data for some slots, such as `per:date_of_death`, `per:schools_attended` or `per:spouse`. Reasons could be long contexts mentioning different relations between different entities. In theses cases, the classes of the two given relation arguments can help to disambiguate which context words are relevant for identifying the relation between the two arguments.

5.4.3 Results of Slot Filling Pipeline

Finally, we evaluate the different multiclass setups in the context of the slot filling pipeline. Table 5.7 shows that the multiclass model which has been jointly trained with entity classification achieves the highest overall micro F1 scores. The scores are higher than using binary models and higher than using multiclass models without entity class information. This confirms that type-aware relation classification models are beneficial and that joint

	SVM	CNN	micro P	micro R	micro F1	macro F1
hop 0	binary skip	binary	31.79	28.23	29.91	**34.20**
hop 0	multi skip	multi, `noNER`	**34.42**	26.66	30.04	32.82
hop 0	multi skip	multi, `slotNER`	23.58	**28.55**	25.83	30.91
hop 0	multi skip	multi, `jointNER`	32.42	27.84	29.95	33.14
hop 0	multi skip	multi, `global`	33.33	27.68	**30.25**	33.98
hop 1	binary skip	binary	9.80	**7.00**	**8.17**	8.28
hop 1	multi skip	multi, `noNER`	12.59	3.89	5.94	7.78
hop 1	multi skip	multi, `slotNER`	6.62	3.00	4.13	4.66
hop 1	multi skip	multi, `jointNER`	**13.47**	5.00	7.29	8.15
hop 1	multi skip	multi, `global`	12.24	5.22	7.32	**9.24**
all	binary skip	binary	23.80	**19.42**	21.39	24.92
all	multi skip	multi, `noNER`	**29.60**	17.20	21.76	23.86
all	multi skip	multi, `slotNER`	20.02	17.94	18.92	21.51
all	multi skip	multi, `jointNER`	27.97	18.36	**22.17**	24.20
all	multi skip	multi, `global`	27.70	18.36	22.08	**25.12**

Table 5.7: Final results for type-aware relation classification models.

training works better than pipeline-based approaches, also in the context of slot filling. The neural structured prediction model performs comparable with the jointly trained model. When looking at macro F1 scores, the neural structured prediction model performs best. In total, Table 5.7 reveals that multiclass models outperform binary models in most evaluation measures.

5.5 Related Work

5.5.1 Type-aware Relation Extraction

While we have given an overview of related work on relation extraction with neural networks in general in Section 3.7.2, we now focus on type-aware relation extraction.

Models

Named entity tags are widely used as features for relation extraction. Table 5.8 provides references. Most works include coarse-grained entity types (as we do in Section 5.3) while others use fine-grained entity types (as in Section 5.2). Some studies include binary entity types (i.e., the predicted or given labels), others include the outputs (e.g., in the form of probabilities) from an entity classifier. In Section 5.2.3, we show that our model performs better when using probabilistic types, most probably because it can draw conclusions from the confidence of the entity classifier and learn to cope with its errors. Similar to our `WEIGHTED` scheme, which represents an entity using type embeddings which are weighted by the probabilities of each type belonging to that entity, Das et al. (2017) represent an entity by adding up embeddings of its Freebase types. In contrast to our work, they do not predict the types though and do not use a weighted sum but learn the type embeddings during

choice of named entity information	example studies
coarse-grained entity types	(Kambhatla, 2004; Roth and Yih, 2004; Zhou et al., 2005; Roth and Yih, 2007; Zhou et al., 2007; Mintz et al., 2009; Kate and Mooney, 2010; Riedel et al., 2010; Yu and Lam, 2010; Hoffmann et al., 2011; Singh et al., 2013a; Li and Ji, 2014; Miwa and Sasaki, 2014; Yu et al., 2015; Gupta et al., 2016; Kirschnick et al., 2016; Miwa and Bansal, 2016; Pawar et al., 2017; Zhang et al., 2017a)
fine-grained entity types	(Yao et al., 2010; Ling and Weld, 2012; Zhang et al., 2013; Liu et al., 2014; Augenstein et al., 2015; Du et al., 2015; Das et al., 2017; Ren et al., 2017)
binary entity types	(Kambhatla, 2004; Roth and Yih, 2004; Zhou et al., 2005; Roth and Yih, 2007; Zhou et al., 2007; Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Ling and Weld, 2012; Zhang et al., 2013; Liu et al., 2014; Vlachos and Clark, 2014; Yu et al., 2015; Gupta et al., 2016; Kirschnick et al., 2016; Miwa and Bansal, 2016; Das et al., 2017)
probabilities from entity classifier	(Kate and Mooney, 2010; Yao et al., 2010; Yu and Lam, 2010; Singh et al., 2013a; Li and Ji, 2014; Miwa and Sasaki, 2014; Pawar et al., 2017; Zhang et al., 2017a)

Table 5.8: Example studies for different choices of using named entity information as features for relation extraction.

training instead. As a result, they cannot represent entities which are not in Freebase while our approach could achieve that.

Especially early studies often apply pipeline approaches by first recognizing named entities and their classes and then extracting relations using the named entity tags as features. Pipeline approaches, however, have several disadvantages, such as accumulation of errors, no interaction between layers and no modeling of dependencies across tasks (cf., Li and Ji (2014)). Moreover, the information flow is uni-directional: when using entity classes as features for relation classification, relation classification can benefit from entity classification but not vice versa (cf., Singh et al. (2013a)). As a result, related work shows that joint entity and relation classification outperforms pipeline models, e.g., (Yao et al., 2010; Miwa and Sasaki, 2014). In Section 5.2.3, our results with neural models and a large number of fine-grained entity types are in line with those findings. Examples for work on joint entity and relation classification without neural networks use structured prediction, integer linear programming, card-pyramid parsing, probabilistic graphical models, imitation learning or Markov Logic. Table 5.9 provides references for example studies. However many of those works still train different models for entity and relation classification and only combine their outputs across tasks (second part of Table 5.9). Underlying task-specific models are, for example, support vector machines (Zhou et al., 2005; Kate and Mooney, 2010), `MultiR` (Liu et al., 2014), CRF (Ling and Weld, 2012) or perceptrons (Ling and Weld, 2012). Only a few studies use a single model for both tasks (last row of Table 5.9).

Recent studies consider joint entity and relation classification in neural models: Miwa and Bansal (2016) train bidirectional LSTM networks on dependency trees and text sequences and update their parameters using both the entity and relation classification loss.

model	example studies
structured prediction	(Yao et al., 2010; Li and Ji, 2014; Miwa and Sasaki, 2014)
integer linear programming	(Roth and Yih, 2004, 2007; Yang and Cardie, 2013)
card-pyramid parsing	(Kate and Mooney, 2010)
probabilistic graphical models	(Yu and Lam, 2010; Singh et al., 2013a)
imitation learning	(Vlachos and Clark, 2014; Augenstein et al., 2015)
Markov Logic	(Riedel et al., 2009)
different models for EC and RE	(Roth and Yih, 2004, 2007; Kate and Mooney, 2010; Yang and Cardie, 2013; Vlachos and Clark, 2014; Augenstein et al., 2015)
single model for EC and RE	(Riedel et al., 2009; Yao et al., 2010; Yu and Lam, 2010; Li and Ji, 2014; Miwa and Sasaki, 2014)

Table 5.9: Example studies for different modeling choices of joint entity and relation classification.

However, they only connect the entity predictions as input to the relation classifier (as we do in Section 5.2) but the entity classifier cannot directly benefit from the relation predictions (in contrast to our model in Section 5.3). Gupta et al. (2016) train recurrent neural networks for entity and relation classification. They iterate training the two tasks and use the outputs from one task as input features for the other task. In contrast, we use CNNs for representing the input and do not apply an iterative training scheme but train both tasks at the same time. Furthermore, we are the first to adopt neural structured prediction output layers to directly model the triples from a knowledge base. This layer allows the model to automatically learn typical relation-entity class correlations (see Section 5.3.5), information which has been used as prior hard constraints by other works, such as by Li and Ji (2014) and Kirschnick et al. (2016). Zhang et al. (2017a) also apply a neural network with a CRF layer but globally normalize the sequence of table filling predictions. Pawar et al. (2017) train an LSTM-based neural network for entity and relation extraction in a multi-task way and then combine the local task-specific outputs in a postprocessing step using a Markov Logic Network. Zheng et al. (2017) convert the relation extraction task into a sequence tagging problem by predicting not the entity classes for each entity but the relation it participates in. They also apply a CRF layer on this tagging task but do not get improvements. In a postprocessing step, they convert the predicted sequence into entity-relation triples. However, their tagging schema does not allow an entity to participate in more than one relation.

We use only word embeddings as input to our models in contrast to many previous works, which use a variety of linguistic features, such as part-of-speech tags or dependency path features, e.g., (Kambhatla, 2004; Roth and Yih, 2004; Zhou et al., 2005; Giuliano et al., 2007; Roth and Yih, 2007; Mintz et al., 2009; Kate and Mooney, 2010; Riedel et al., 2010; Yao et al., 2010; Hoffmann et al., 2011; Ling and Weld, 2012; Li and Ji, 2014; Liu et al., 2014; Miwa and Sasaki, 2014; Augenstein et al., 2015; Yu et al., 2015; Gupta et al., 2016; Kirschnick et al., 2016; Miwa and Bansal, 2016; Pawar et al., 2017; Ren et al., 2017; Zhang et al., 2017a).

dataset	example studies
ACE	(Zhou et al., 2005, 2007; Singh et al., 2013a; Li and Ji, 2014; Yu et al., 2015; Miwa and Bansal, 2016; Pawar et al., 2017; Zhang et al., 2017a)
ERR	(Roth and Yih, 2004; Giuliano et al., 2007; Roth and Yih, 2007; Kate and Mooney, 2010; Miwa and Sasaki, 2014; Gupta et al., 2016; Zhang et al., 2017a)
own datasets	(Yao et al., 2010; Ling and Weld, 2012; Liu et al., 2014; Augenstein et al., 2015; Ren et al., 2017)

Table 5.10: Datasets and example studies using them.

Datasets

There exist a few manually labeled benchmark datasets suitable for joint entity and relation extraction, for instance ACE 2004, ACE 2005 and ERR. Table 5.10 shows example studies using them.

The ACE datasets (Doddington et al., 2004; Walker et al., 2006), which have already been briefly introduced in Section 3.7.2, label the data with seven coarse-grained entity types (`person`, `organization`, `geo-political entity`, `location`, `facility`, `weapon` and `vehicle`) and seven vs. six (2004 vs. 2005 version) coarse-grained relation types ({`physical`, `person-social`, `employment-organization`, `agent-artifact`, `discourse`, `person/organization affiliation`, `geo-political-entity affiliation`} vs. {`physical`, `person-social`, `employment-organization`, `agent-artifact`, `geo-political-entity affiliation`, `part-whole`}).

The entity and relation recognition (ERR) dataset (Roth and Yih, 2004, 2007), which we use in Section 5.3, focuses on entity and relation classes which are more related to the slot filling task. Therefore, we use it in this thesis (for a dataset description and statistics, see Section 5.3.2). Modeling the joint entity and relation extraction task as a table filling problem is introduced by Miwa and Sasaki (2014) and mostly applied to the ERR dataset. Pawar et al. (2017) also apply it to the ACE dataset.

Other studies on joint entity and relation classification apply distant supervision for building their own datasets, as we do in Section 5.2.

5.5.2 CRF Layers for Neural Networks

Conditional random field models (Lafferty et al., 2001) have been proposed for sequence labeling, i.e., for assigning an output tag to each token of an input sequence. However, more general variants also exists, such as Semi Markov CRFs (Sarawagi and Cohen, 2005), which segment the input tokens and assign tags to each segment. Traditional CRF models are non-neural models with hand-crafted input features. Training neural networks based on a sequence of predictions was already proposed in the 1990s (Bottou et al., 1997; LeCun et al., 1998). Other works add non-linear or RNN layers to CRFs (Peng et al., 2009; Yao et al., 2014) or Markov networks (Do and Artieres, 2010).

Nowadays, CRF output layers are becoming popular in neural networks to obtain globally normalized output sequences. They are used for a variety of tasks as shown in Ta-

task	example studies
named entity tagging	(Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016)
part-of-speech tagging	(Huang et al., 2015; Andor et al., 2016; Ma and Hovy, 2016; Zhang et al., 2017b)
dependency parsing	(Zhou et al., 2015a; Andor et al., 2016; Wiseman and Rush, 2016; Cai et al., 2017)
CCG parsing	(Lee et al., 2016)
constituency parsing	(Durrett and Klein, 2015)
word segmentation	(Zhang et al., 2016b)
sentence compression	(Andor et al., 2016)
aspect-based sentiment analysis	(Wang et al., 2016b)
question answering	(Raiman and Miller, 2017)
community question answering	(Xiang et al., 2016)
semantic slot filling	(Xu and Sarikaya, 2013; Yao et al., 2014)
word ordering	(Wiseman and Rush, 2016)
machine translation	(Wiseman and Rush, 2016)

Table 5.11: Example studies for using CRF output layers for different tasks.

neural network type	example studies
feed-forward network	(Durrett and Klein, 2015; Zhou et al., 2015a; Andor et al., 2016)
LSTM	(Lample et al., 2016; Lee et al., 2016; Ma and Hovy, 2016; Xiang et al., 2016; Zhang et al., 2016b; Raiman and Miller, 2017; Zhang et al., 2017b)
RNN	(Yao et al., 2014)
encoder-decoder models	(Sountsov and Sarawagi, 2016; Wiseman and Rush, 2016)
recursive neural networks	(Wang et al., 2016b)
CNN	(Ma and Hovy, 2016; Xiang et al., 2016)

Table 5.12: Example studies for integrating CRF output layers into different neural network types.

ble 5.11. In most cases, those are sequence labeling tasks or tasks for which models typically output a sequence of tags. In this study, we apply a CRF output layer to joint entity and relation extraction, both sentence classification tasks which output a single class. Neural and non-neural CRFs have been applied to information extraction tasks before with the task being cast as a token-labeling problem (Sarawagi and Cohen, 2005; Zhu et al., 2005; Culotta et al., 2006; Peng and McCallum, 2006; Sutton and McCallum, 2007; Zheng et al., 2017). In contrast, we propose a simpler linear-chain CRF model which directly connects entity and relation classes instead of assigning a label to each token of the input sequence. This is more similar to the table filling optimization by Zhang et al. (2017a) or the factor graphs by Yao et al. (2010) and Singh et al. (2013a) but computationally simpler. Since our tag set is rather small and the sequence we optimize has only length three, we do not need to apply beam search with early updates (Collins and Roark, 2004) to make the model more efficient as in other neural networks with CRF layers, such as (Zhou et al., 2015a; Andor et al., 2016; Sountsov and Sarawagi, 2016; Wiseman and Rush, 2016; Zhang et al., 2016b; Raiman and Miller, 2017; Zhang et al., 2017a).

CRF output layers are integrated into a variety of neural networks, such as feed-forward

networks, LSTMs, RNNs, encoder-decoder models, recursive neural networks or CNNs. Table 5.12 provides references to example studies. CRF layers are also used in the encoder of neural autoencoders (Zhang et al., 2017b) or in the attention layer of a network (Kim et al., 2017). In this thesis, we add them to CNN-based neural networks, building on the promising results of CNNs in the previous chapters.

5.6 Summary of Contributions

In this section, we proposed different approaches of type-aware relation extraction models: pipeline approaches, joint training and neural structured prediction models. Pipeline approaches use entity classes as input features for relation extraction. In joint training, entity and relation classification are learned together by combining their loss functions as in multitask learning. This allows the entity classification to guide the relation classification. Structured prediction optimizes a triple of entity class and relation predictions, allowing both tasks to benefit from each other.

We evaluated our models on two different datasets: a large-scale distantly supervised dataset consisting of web texts, and a medium-scale manually labeled dataset. For pipeline models, we found that using the probabilities of the entity classifier directly outperformed binarizing its output, most probably because it preserves the confidence of the entity classifier and can better learn to cope with noisy predictions. Joint training outperformed pipeline models. This finding has been shown before, especially for non-neural models. Our results show that it also holds for neural networks and a large number of fine-grained entity types.

Finally, neural structured prediction models outperformed jointly trained models with shared parameters on the manually labeled ERR dataset. We proposed a way to apply a CRF layer for a sentence classification task without converting it into a token-level tagging problem. Instead, we designed the input to the CRF layer as a triple consisting of the classes of two entities and the relation between them. This is beneficial because it allows to learn transitions between entity and relation classes while avoiding higher-order connections between the different items of the CRF output sequence (which would be necessary in token-level tagging scenarios).

The code for the pipeline and joint training architectures is provided at `http://cistern.cis.lmu.de/noise-mitigation`, the code for the neural structured prediction models is available at `http://cistern.cis.lmu.de/globalNormalization`.

Finally, we added entity class information to our multiclass slot filler classification models. We showed that joint training of multiclass relation classification and entity classification led to the best micro F1 results and neural structured prediction models achieved the best macro F1 scores.

Chapter 6

Conclusion and Future Work

In this thesis, we explored neural networks for automatically populating knowledge bases from text. We addressed the tasks slot filling, uncertainty detection and type-aware relation extraction and contributed to state-of-the-art research with novel neural architectures for those tasks and a benchmark dataset for slot filling.

In Chapter 3, we developed a state-of-the-art slot filling system. Slot filling aims at extracting information about named entities from a large text corpus. We addressed this challenge by implementing a modular system of various natural language processing components. We made it publicly available at `http://cistern.cis.lmu.de/CIS_SlotFilling`. The core of the system is the slot filler classification component which directly influences the output of the system by scoring slot filler candidates based on the textual contexts they appear in. We proposed `contextCNN`, a convolutional neural network especially designed for relation classification. It splits the context at the relation arguments and computes features for each of the context parts. To avoid redundant training and make more effective use of available training data for inverse relations, such as `per:children` and `per:parents`, it models both relations with only one output label and includes a flag stating which relation argument appears first in the sentence. Its code is also provided at `http://cistern.cis.lmu.de/CIS_SlotFilling`. Our experimental results and analysis showed that `contextCNN` is able to extract n-grams relevant to the different relations and improves the results when combining it with patterns and support vector machines, which are traditional models for slot filling. Using `contextCNN` in the slot filling pipeline improves the performance by 5.0% micro F1 and 2.9% macro F1 (absolute values). Since the slot filling task does not provide an official large-scale training dataset, we extracted training data using distant supervision and presented a data selection strategy based on self-training in order to reduce noisy labels. With the manual assessments of the slot filling evaluations, it is only possible to evaluate outputs of the whole slot filling pipeline. However, it is necessary to assess and compare the performance of individual pipeline components as well. Therefore, we proposed a way to create a development and test set for the slot filler classification component. In particular, we automatically extracted a labeled relation classification benchmark dataset from the manually assessed system outputs from 2012–2014. Since the labels are based on manual annotations, they are less noisy than

the labels from datasets created with distant supervision. Moreover, the dataset is closely related to the actual slot filling evaluations because the sentences have been directly extracted from system responses. We showed that results on this benchmark dataset are correlated with the slot filling pipeline results. Therefore, the dataset will be helpful for other shared task participants and slot filling researchers in the future to develop their systems and compare their slot filler classification modules without the need of assessing the whole slot filling pipeline. We published the script for reproducing the dataset at `http://cistern.cis.lmu.de/SFbenchmark`. Similar to the slot filling pipeline, the best result on the benchmark dataset was achieved by a combination of patterns, support vector machines and `contextCNN`. The micro F1 result of the combination is 51%, the macro F1 score is 53% on test. In our experiments, we compared binary models to multiclass models and also investigated the effect of different domains in the source corpus on the slot filler classification component. While binary models outperformed multiclass models on the slot filling benchmark for relation classification, multiclass models led to a slightly better micro F1 score of the slot filling pipeline. We performed several analyses to assess the impact of the different components of the slot filling system. Our recall analysis revealed that information retrieval, coreference resolution and the alias component are crucial parts for obtaining high recall. Since coreference resolution is expensive and slows down the system considerably, we prepared a resource of coreference chains for the slot filling source corpus and made it available to other shared task participants at `http://cistern.cis.lmu.de/corefresources`. In a manual error analysis, we showed that the candidate extraction component and the slot filler classification component are responsible for most of the false positive extractions of the system. Our ablation study finally showed the positive impact of entity linking, coreference resolution and `contextCNN` on the slot filling results.

One possible future work is the extension of the slot filling system to other languages. Since 2016, the slot filling shared task includes evaluations for Chinese and Spanish texts besides the traditional evaluation for English texts. This might be helpful for entities for which more information and resources exist in other languages. Another research direction for future work is to move from models that have a strong pipeline character to models that incorporate more joint training. A recent trend in natural language processing is end-to-end training. For example, the different models of statistical machine translation have been superseded by an encoder-decoder neural network, which is trained end to end (Sutskever et al., 2014; Bahdanau et al., 2015). Examples for other tasks for which end-to-end systems are applied instead of traditionally used pipelines are speech recognition (Graves and Jaitly, 2014; Amodei et al., 2016), question answering (Weston et al., 2015; Kumar et al., 2016), coreference resolution (Lee et al., 2017) and dialogue systems (Serban et al., 2016). For slot filling, replacing the pipeline with an end-to-end system is challenging since the system would need to process a large number of input documents in order to extract answers to the queries. This might lead to long training and processing times. Moreover, the limited capacity of the hidden layers of neural networks might require the usage of external memory components (Weston et al., 2015). A first step towards this direction might be an end-to-end modeling of a single document. Training several pipeline

components as a single system could be achieved by introducing confidence scores for the intermediate components and using backpropagation to measure their error and update them accordingly (Rumelhart et al., 1986).

Chapter 4 described our work on uncertainty detection. Uncertainty detection addresses the challenge of identifying a sentence or a phrase as a fact (certain) vs. as an unspecific, speculative, subjective or ambiguous utterance (uncertain). This distinction is important for knowledge base population since statements derived from uncertain information should be either labeled as such or entirely excluded from a knowledge base. For uncertainty detection, we developed several attention-based methods. Attention is a promising approach since uncertainty is mostly expressed by one or more hedge cues, which can be distributed over the sentence. In particular, we made a first attempt to systematize the design space of attention. We identified the following axes: weighted vs. unweighted selection, sequence-agnostic vs. sequence-preserving selection, and internal vs. external attention. We proposed different novel attention mechanisms along those axes: k-max average attention, sequence-preserving attention and external attention. K-max average attention only considers the vectors with the k maximum attention weights for computing the result of the attention layer. It, thus, alleviates the need of the network to produce sharp attention weight distributions in order to reduce noise from unimportant input vectors. Sequence-preserving attention keeps order information from the input vectors, an information which is typically lost in standard average attention but might be important for several natural language processing tasks. External attention allows the network to use external knowledge for computing the attention weights. For uncertainty detection, we used vector representations of uncertainty cues as external knowledge. However, all of our proposed attention mechanisms can be used for other tasks as well. For sentiment analysis, for example, one could use vectors representing sentiment words as a resource for external attention. We conducted our experiments on the dataset of the CoNLL 2010 shared task on learning to detect hedges. To the best of our knowledge, we are the first to explore convolutional and recurrent neural networks on that dataset. Our networks improved the state-of-the-art results on the Wikipedia domain by 3.6% F1 absolute without using any hand-crafted features. On Wikipedia, our best results were 67.52%. On the Biomedical domain, our models performed comparably to state of the art with 85.57%. The best results were obtained with external attention on the input word embeddings with (k-max) average attention. In an analysis, we compared different selection strategies: pooling, internal attention and external attention and showed that external attention provided the network with the clearest focus on the relevant parts of the sentence. Furthermore, we compared recurrent neural networks with convolutional neural networks and showed that convolutional neural networks outperformed recurrent neural networks especially for long sentences. They also seemed to be slightly more robust in the presence of many out-of-vocabulary words. We published the source code of our models at `http://cistern.cis.lmu.de/attentionUncertainty`. For the first time, we integrated an uncertainty detection component into the slot filling pipeline. The component improved precision by 1.4% and the final micro F1 by 0.4% absolute.

In future work, the detection of uncertainty for knowledge base population can be ex-

tended in several ways: First, it is possible to also identify the degree of uncertainty and include corresponding labels as additional information for facts in a knowledge base. A benchmark dataset which can be used for this investigation is FactBank (Saurí and Pustejovsky, 2009). Second, there are other characteristics of information which we have not included in our investigation so far. Two examples are contribution and temporality. Contribution identifies the source of information, e.g., the author of a statement. Depending on his/her credibility, a statement might be considered as true or possibly wrong (Saurí and Pustejovsky, 2009). Thus, future work could also extract the author of a statement (and if possible an assessment of his or her credibility, for example, as in (De Marneffe et al., 2012)) to include it in a knowledge base as well. Temporality accounts for the fact that some characteristics or relations of entities are only true for a certain amount of time. Employment of a person, for example, or the place of residence, can change over time. For those relations, time stamps could also be extracted and entered in a knowledge base. When including an uncertainty component in a slot filling system, the scope of the uncertainty cues is highly relevant. Thus, another future research direction is scope identification in addition to uncertainty cue detection.

In Chapter 5, we presented our investigations of type-aware relation extraction. In the first part of the chapter (Section 5.2), we described pipeline-based and jointly trained models in which entity types or entity type predictions are used as input features for relation classification. As entity and relation classifiers, we applied slightly adapted versions of `contextCNN`. We explored different pipeline approaches: `PREDICTED-HIDDEN`, which computes entity class features by feeding the entity class probabilities into a fully-connected feed-forward layer, `BINARY-HIDDEN`, which uses the predicted entity classes instead of their probabilities as input of a fully-connected hidden layer, `BINARY`, which uses the predicted entity classes directly as features (without a hidden layer), and `WEIGHTED`, which uses the predicted entity class probabilities to weight pre-trained entity class embeddings. Our experiments on a large-scale distantly supervised dataset showed that `WEIGHTED` and `PREDICTED-HIDDEN`, i.e., using the probabilities of the entity class predictions, outperformed deriving features from binary decisions. For joint training of entity and relation classes, we applied the architecture from `PREDICTED-HIDDEN` but trained both the entity and the relation classifier jointly. This approach outperformed all pipeline models by a large margin. Its area under the precision-recall curve was 0.66 while the best pipeline model achieved only 0.53. Our analysis revealed that deriving features from the textual context with `contextCNN` had a high impact on the results, especially for relation classification. The code for our models is provided at `http://cistern.cis.lmu.de/noise-mitigation`. In the second part of the chapter (Section 5.3), we proposed a model which is trained on the sequence of entity and relation class outputs. The probability of a sequence of predictions is obtained by globally normalizing the score of the sequence over the scores of all possible sequences. We propose to apply a linear-chain conditional random field output layer to be able to learn transition scores indicating which entity classes and relation classes often occur together. Again, we used context splitting and convolutional layers to derive features from the input word embeddings. The input of the model is a sentence together with a pair of entities (or words). The output is a globally normalized sequence

consisting of the class of the first entity, the relation between the two entities and the class of the second entity. We conducted our experiments on a medium-scale manually labeled dataset for entity and relation recognition and obtained results comparable to other state-of-the-art models. We evaluated three setups with increasing difficulty: (1) predicting relations only between named entity pairs, (2) table filling with entity boundaries, (3) table filling without entity boundaries. In all setups, the conditional random field output layer improved results upon task-specific softmax output layers by up to 4.4% F1 on average. Our analysis of the transition scores of the linear-chain conditional random field layer showed that the model has learned reasonable correlations between entity and relation classes. We made the code for the neural structured prediction models available at `http://cistern.cis.lmu.de/globalNormalization`. Finally, we integrated joint entity and relation classifiers into the slot filling pipeline. On the slot filling benchmark, the structured prediction models outperformed other type-aware relation classification models in terms of macro F1, especially on unseen test data. A pipeline model achieved the best micro F1 scores of all multiclass models though. When applied in the slot filling pipeline, the neural structured prediction models led to the best macro F1 scores with 25.12% while the jointly trained models had the highest micro F1 scores with 22.17%. These results were better than both using binary models and using multiclass models without entity class information. To the best of our knowledge, this work is the first to use neural networks for joint entity and relation classification in a slot filling pipeline.

Training a model on joint entity and relation classification can be seen as a form of multi-task learning (Caruana, 1998; Collobert and Weston, 2008). Multi-task learning is applied in natural language processing to create more robust input representations or guide the network during training (Collobert and Weston, 2008; Collobert et al., 2011; Klerke et al., 2016; Martínez Alonso and Plank, 2017). For relation classification, not only entity classification but also other tasks, such as semantic role labeling, may be useful. Therefore, a direction for future work is the investigation of other possible additional tasks.

Appendix A

Additional Material for Slot Filling

A.1 Slot List

slot	filler cardinality	inverse
per:age	1	-
per:alternate_names	≥ 1	-
per:cause_of_death	1	-
per:charges	≥ 1	-
per:children	≥ 1	per:parents
per:cities_of_residence	≥ 1	gpe:residents_of_city
per:city_of_birth	1	gpe:births_in_city
per:city_of_death	1	gpe:deaths_in_city
per:countries_of_residence	≥ 1	gpe:residents_of_country
per:country_of_birth	1	gpe:births_in_country
per:country_of_death	1	gpe:deaths_in_country
per:date_of_birth	1	-
per:date_of_death	1	-
per:employee_or_member_of	≥ 1	{org,gpe}:employees_or_members
per:origin	≥ 1	-
per:other_family	≥ 1	per:other_family
per:religion	1	-
per:schools_attended	≥ 1	org:students
per:siblings	≥ 1	per:siblings
per:spouse	≥ 1	per:spouse
per:stateorprovince_of_birth	1	gpe:births_in_stateorprovince
per:stateorprovince_of_death	1	gpe:deaths_in_stateorprovince
per:statesorprovinces_of_residence	≥ 1	gpe:residents_of_stateorprovince
per:title	≥ 1	-
org:alternate_names	≥ 1	-
org:city_of_headquarters	1	gpe:headquarters_in_city
org:country_of_headquarters	1	gpe:headquarters_in_country
org:date_dissolved	1	-
org:date_founded	1	-
org:founded_by	≥ 1	{per,org,gpe}:organizations_founded
org:members	≥ 1	{org,gpe}:member_of
org:number_of_employees_members	1	-
org:parents	≥ 1	{org,gpe}:subsidiaries
org:political_religious_affiliation	≥ 1	-
org:shareholders	≥ 1	{per,org,gpe}:holds_shares_in
org:stateorprovince_of_headquarters	1	gpe:headquarters_in_stateorprovince
org:top_members_employees	≥ 1	per:top_member_employee_of
org:website	1	-

Table A.1: TAC KBP slot filling slots and their inverse. Filler cardinality shows whether the output is a single filler or a list of multiple fillers.

Table A.1 provides the different relations (slots) from the slot filling task as introduced in Section 3.1.

A.2 Distantly Supervised Patterns

slot	number of patterns
per:age	497
per:alternate_names	3250
per:cause_of_death	3661
per:charges	1714
per:children	2143
per:locations_of_residence	10688
per:location_of_birth	7431
per:location_of_death	6767
per:date_of_birth	3228
per:date_of_death	8104
per:employee_or_member_of	25446
per:origin	8240
per:other_family	1430
per:parents	5966
per:religion	3716
per:schools_attended	3161
per:siblings	5452
per:spouse	4074
per:title	11104
org:alternate_names	2053
org:location_of_headquarters	53612
org:date_dissolved	1297
org:date_founded	1931
org:founded_by	3151
org:member_of	4638
org:members	378
org:number_of_employees_members	4183
org:parents	7262
org:political_religious_affiliation	4280
org:shareholders	86
org:subsidiaries	10434
org:top_members_employees	5933
org:website	161

Table A.2: Statistics of distantly supervised patterns from Roth et al. (2013) with (city, state or province, country)-slots merged to one location-of slot.

Table A.2 provides statistics of the distantly supervised pattern set as introduced in Section 3.3.2.

A.3 Dataset Statistics

slot	+	−	ratio
per:age	4151	32254	0.13
per:alternate_names	287	752	0.38
per:cause_of_death	173	994	0.17
per:charges	5	1008	0.00
per:children	61826	138456	0.45
per:date_of_birth	38413	34182	1.12
per:date_of_death	3039	9404	0.32
per:employee_or_member_of	283434	585997	0.48
per:location_of_birth	94565	187957	0.50
per:location_of_death	9053	57259	0.16
per:locations_of_residence	247312	551846	0.45
per:origin	165777	414065	0.40
per:other_family	46	957	0.05
per:religion	5	1004	0.00
per:schools_attended	9618	297964	0.03
per:siblings	5186	20939	0.25
per:spouse	70636	201608	0.35
per:title	11175	211475	0.05
org:alternate_names	1691	5879	0.29
org:date_dissolved	94	570	0.16
org:date_founded	8079	53809	0.15
org:founded_by	42885	280808	0.15
org:location_of_headquarters	354491	736513	0.48
org:members	70611	177586	0.40
org:number_of_employees_members	7	648	0.01
org:parents	48490	111628	0.43
org:shareholders	47	1325	0.04
org:top_members_employees	194726	412137	0.47
total	1725822	4529024	0.38

Table A.3: Statistics of the training data set.

Table A.3 provides statistics of the slot filling training data set which has been created with distant supervision. It is introduced and described in Section 3.4.2.

Table A.4 and Table A.5 show statistics of the slot filling benchmark dataset as described in Section 3.4.3. In Table A.4, the data is split into a development and evaluation set according to the slot filling assessment years. In Table A.5, the data is split according to genre and randomly assigned to development and evaluation sets.

	dev			eval		
slot	+	–	ratio	+	–	ratio
per:age	185	119	1.55	129	130	0.99
per:alternate_names	53	329	0.16	3	357	0.01
per:cause_of_death	148	103	1.44	34	69	0.49
per:charges	42	148	0.28	66	118	0.56
per:children	214	451	0.47	26	452	0.06
per:cities_of_residence	126	979	0.13	86	446	0.19
per:city_of_birth	42	149	0.28	14	151	0.09
per:city_of_death	108	224	0.48	12	161	0.07
per:countries_of_residence	100	488	0.20	62	260	0.24
per:country_of_birth	11	116	0.09	7	139	0.05
per:country_of_death	22	159	0.14	8	88	0.09
per:date_of_birth	55	100	0.55	5	85	0.06
per:date_of_death	119	178	0.67	15	102	0.15
per:employee_or_member_of	426	1873	0.23	146	732	0.2
per:origin	132	552	0.24	89	465	0.19
per:other_family	26	309	0.08	14	353	0.04
per:parents	71	431	0.16	53	388	0.14
per:religion	15	108	0.14	17	40	0.42
per:schools_attended	97	397	0.24	21	239	0.09
per:siblings	49	172	0.28	19	320	0.06
per:spouse	157	478	0.33	31	470	0.07
per:stateorprovince_of_birth	29	116	0.25	7	116	0.06
per:stateorprovince_of_death	59	145	0.41	7	109	0.06
per:statesorprovinces_of_residence	90	558	0.16	44	214	0.21
per:title	879	2463	0.36	331	1136	0.29
org:alternate_names	152	458	0.33	69	180	0.38
org:city_of_headquarters	60	279	0.22	38	124	0.31
org:country_of_headquarters	99	243	0.41	37	78	0.47
org:date_dissolved	4	92	0.04	2	54	0.04
org:date_founded	35	167	0.21	29	79	0.37
org:founded_by	48	904	0.05	47	306	0.15
org:member_of	9	307	0.03	0	177	0.0
org:members	65	334	0.19	5	162	0.03
org:number_of_employees_members	26	87	0.30	9	50	0.18
org:parents	50	524	0.10	10	246	0.04
org:political_religious_affiliation	15	196	0.08	5	31	0.16
org:shareholders	24	874	0.03	3	203	0.01
org:stateorprovince_of_headquarters	45	192	0.23	23	70	0.33
org:subsidiaries	90	762	0.12	15	138	0.11
org:top_members_employees	430	1627	0.26	169	412	0.41
org:website	66	62	1.06	16	5	3.2
total	4473	18253	0.25	1723	9455	0.18

Table A.4: Statistics of the slot filling benchmark dataset.

	news						web					
	dev			eval			dev			eval		
slot	+	−	ratio	+	−	ratio	+	−	ratio	+	−	ratio
org:alternate_names	85	209	0.41	86	204	0.42	23	105	0.22	21	99	0.21
org:city_of_headquarters	40	162	0.25	43	167	0.26	9	36	0.25	5	32	0.16
org:country_of_headquarters	65	139	0.47	59	128	0.46	5	23	0.22	7	24	0.29
org:date_dissolved	3	63	0.05	3	63	0.05	0	7	0.0	0	7	0.0
org:date_founded	30	98	0.31	30	96	0.31	2	21	0.1	1	20	0.05
org:founded_by	41	498	0.08	40	499	0.08	5	95	0.05	4	98	0.04
org:member_of	3	203	0.01	3	207	0.01	2	29	0.07	1	28	0.04
org:members	33	210	0.16	32	208	0.15	3	35	0.09	2	39	0.05
org:number_of_employees_members	17	55	0.31	16	55	0.29	1	12	0.08	1	12	0.08
org:parents	28	287	0.1	27	288	0.09	3	91	0.03	2	89	0.02
org:political_religious_affiliation	10	98	0.1	9	94	0.1	0	16	0.0	0	15	0.0
org:shareholders	14	462	0.03	13	466	0.03	0	63	0.0	0	60	0.0
org:stateorprovince_of_headquarters	24	103	0.23	30	103	0.29	6	24	0.25	7	26	0.27
org:subsidiaries	47	335	0.14	47	342	0.14	6	96	0.06	5	102	0.05
org:top_members_employees	264	837	0.32	268	844	0.32	27	151	0.18	26	156	0.17
org:website	37	23	1.61	36	23	1.57	3	10	0.3	4	9	0.44
per:age	168	102	1.65	139	95	1.46	6	16	0.38	6	16	0.38
per:alternate_names	29	254	0.11	22	245	0.09	2	47	0.04	2	45	0.04
per:cause_of_death	90	76	1.18	88	72	1.22	3	9	0.33	2	8	0.25
per:charges	54	113	0.48	47	111	0.42	3	17	0.18	1	14	0.07
per:children	118	362	0.33	115	348	0.33	6	35	0.17	2	35	0.06
per:cities_of_residence	111	622	0.18	93	586	0.16	4	84	0.05	5	74	0.07
per:city_of_birth	36	165	0.22	22	91	0.24	0	12	0.0	0	13	0.0
per:city_of_death	88	215	0.41	29	126	0.23	0	17	0.0	3	14	0.21
per:countries_of_residence	79	309	0.26	76	319	0.24	5	36	0.14	3	40	0.07
per:country_of_birth	10	116	0.09	8	94	0.09	0	16	0.0	0	15	0.0
per:country_of_death	16	123	0.13	14	94	0.15	0	12	0.0	0	9	0.0
per:date_of_birth	36	72	0.5	25	65	0.38	0	18	0.0	0	16	0.0
per:date_of_death	69	123	0.56	65	119	0.55	2	15	0.13	2	15	0.13
per:employee_or_member_of	272	1017	0.27	248	965	0.26	22	241	0.09	19	232	0.08
per:origin	111	424	0.26	95	392	0.24	8	75	0.11	6	69	0.09
per:other_family	18	264	0.07	16	261	0.06	2	18	0.11	2	17	0.12
per:parents	70	332	0.21	54	329	0.16	2	29	0.07	1	32	0.03
per:religion	14	58	0.24	12	57	0.21	2	12	0.17	2	11	0.18
per:schools_attended	67	242	0.28	48	243	0.2	2	51	0.04	2	48	0.04
per:siblings	37	184	0.2	34	185	0.18	0	39	0.0	0	38	0.0
per:spouse	103	379	0.27	79	364	0.22	4	44	0.09	3	45	0.07
per:stateorprovince_of_birth	20	100	0.2	17	103	0.17	0	13	0.0	0	9	0.0
per:stateorprovince_of_death	32	122	0.26	30	107	0.28	3	8	0.38	1	10	0.1
per:statesorprovinces_of_residence	64	334	0.19	56	326	0.17	7	35	0.2	4	36	0.11
per:title	576	1597	0.36	501	1498	0.33	54	162	0.33	51	145	0.35
total	3029	11487	0.26	2675	10982	0.24	232	1875	0.12	203	1822	0.11

Table A.5: Statistics of the slot filling benchmark dataset, genre split.

A.4 Slot Filler Classification Model Choices

	PATuschema		PATdist		SVMbow		SVMskip	
	dev	eval	dev	eval	dev	eval	dev	eval
per:age	.65	.77	**.69**	**.80**	.84	**.75**	**.86**	.74
per:alternate_names	.40	.18	**.50**	**.50**	.35	**.02**	**.35**	**.02**
per:cause_of_death	.40	.00	**.44**	**.11**	.80	**.36**	**.82**	.32
per:children	**.11**	**.07**	.10	**.07**	.71	.43	**.81**	**.68**
per:date_of_birth	.64	**.57**	**.67**	**.57**	.99	**.67**	**1.0**	**.67**
per:date_of_death	.24	.10	**.30**	**.32**	.78	.50	**.79**	**.54**
per:empl_memb_of	**.32**	**.24**	.24	.22	**.43**	.34	.42	**.36**
per:location_of_birth	.22	**.39**	**.30**	.30	**.62**	**.29**	.59	.27
per:loc_of_death	.08	**.00**	**.13**	**.00**	**.64**	.27	**.64**	**.34**
per:loc_of_residence	**.26**	**.32**	.10	.03	.28	.27	**.31**	**.33**
per:origin	**.26**	**.19**	.13	.11	**.68**	.59	.65	**.64**
per:parents	.26	.36	**.27**	**.38**	.51	.63	**.65**	**.79**
per:schools_att	.25	**.26**	**.27**	**.26**	**.78**	.67	**.78**	**.71**
per:siblings	**.17**	.48	.14	**.50**	**.60**	.64	**.60**	**.68**
per:spouse	.39	.49	**.40**	**.53**	.63	.29	**.67**	**.32**
per:title	**.51**	**.45**	.48	.42	.36	.30	**.54**	**.48**
org:alternate_names	.65	**.74**	**.70**	.71	.60	**.63**	**.62**	.62
org:date_founded	.44	**.53**	**.47**	.40	**.61**	.59	.57	**.70**
org:founded_by	.22	.51	**.39**	**.62**	**.85**	**.76**	.77	.74
org:loc_of_headqu	**.43**	**.38**	.39	.30	**.43**	**.44**	**.43**	.42
org:members	**.03**	**.29**	**.03**	**.29**	.49	**.16**	**.70**	.13
org:parents	**.31**	.10	**.31**	**.18**	.28	.10	**.37**	**.20**
org:subsidiaries	.20	.13	**.32**	**.56**	.33	.24	**.38**	**.37**
org:top_memb_empl	.45	.38	**.53**	**.46**	**.51**	**.57**	.43	.55
macro F1	.33	.33	**.35**	**.36**	.59	.44	**.62**	**.48**

Table A.6: F1 results of different patterns and SVM feature choices on slot filling benchmark dataset.

Table A.6 and Table A.7 provide slot-wise results for the different model choices which are described in Section 3.5.2.

	PCNN		PCNNext		contextCNN	
	dev	test	dev	test	dev	test
per:age	.85	.74	**.86**	**.76**	.83	**.76**
per:alternate_names	.21	**.04**	.27	**.04**	**.32**	**.04**
per:cause_of_death	.00	.00	.67	.46	**.77**	**.52**
per:children	.73	.49	.77	.56	**.82**	**.61**
per:date_of_birth	.99	.67	.99	**.77**	**1.0**	**.77**
per:date_of_death	**.77**	**.48**	.76	.41	.72	**.48**
per:empl_memb_of	.39	.30	.40	.26	**.41**	**.37**
per:location_of_birth	**.59**	.25	**.59**	**.26**	**.59**	.23
per:loc_of_death	.63	.32	**.64**	**.34**	.63	.28
per:loc_of_residence	.19	.21	.18	.21	**.20**	**.23**
per:origin	.38	.30	.38	.36	**.43**	**.39**
per:parents	.50	.48	.44	.34	**.65**	**.78**
per:schools_att	.71	.42	.70	.45	**.72**	**.55**
per:siblings	.57	.65	**.64**	.68	.63	**.70**
per:spouse	.66	.34	.66	**.39**	**.67**	.30
per:title	.50	.44	.53	.45	**.57**	**.46**
org:alternate_names	.58	.59	.54	.56	**.65**	**.66**
org:date_founded	**.65**	.62	**.65**	.69	.64	**.71**
org:founded_by	.76	.60	.74	.65	**.80**	**.68**
org:loc_of_headqu	.39	.42	.40	.40	**.43**	**.45**
org:members	.46	**.06**	.56	**.06**	**.65**	.04
org:parents	.25	.10	.27	.11	**.41**	**.16**
org:subsidiaries	.18	.40	.15	.39	**.36**	**.44**
org:top_memb_empl	**.44**	.48	.43	.52	.43	**.53**
macro F1	.52	.39	.55	.42	**.60**	**.46**

Table A.7: F1 results of different CNN architecture choices on slot filling benchmark dataset.

A.5 Hyperparameters

A.5.1 Hyperparameters of Support Vector Machines

	SVMbow	SVMskip		
	genre-independent	genre-independent	trained on news	trained on web
slot	C	C	C	C
per:age	0.03	0.03	0.01	0.01
per:alternate_names	10.00	0.03	-	-
per:cause_of_death	0.01	1.00	-	-
per:children	0.03	0.01	0.03	0.03
per:date_of_birth	0.01	0.01	0.10	0.01
per:date_of_death	0.01	0.01	0.10	0.01
per:empl_memb_of	0.03	0.01	0.03	0.03
per:location_of_birth	0.01	0.01	0.10	0.01
per:loc_of_death	1.00	0.01	0.10	0.01
per:loc_of_residence	3.00	0.01	10.00	0.01
per:origin	3.00	0.10	30.00	0.03
per:schools_att	3.00	0.10	0.30	0.10
per:siblings	1.00	0.01	0.01	0.01
per:spouse	0.01	0.10	30.00	0.01
per:title	30.00	10.00	1.00	0.10
org:alternate_names	0.01	10.00	0.10	0.01
org:date_founded	0.01	0.01	0.30	0.01
org:founded_by	0.01	0.01	0.03	0.03
org:loc_of_headqu	0.03	0.01	10.00	0.10
org:members	0.01	3.00	1.00	0.03
org:parents	0.03	0.01	3.00	0.03
org:top_memb_empl	0.01	0.30	0.01	0.01

Table A.8: Hyperparameter (penalty parameter C of error term) of SVMs for slot filling. No training of genre-specific models if no genre-specific training data is available.

Table A.8 shows the penalty values C of the error term of the SVM, which we tune on the development set. The SVMs are described in Section 3.3.3. For all other parameters of the SVM, we use the default values of LinearSVC:[1] For regularization, for example, l_2 penalty is applied, the loss function is squared hinge loss and the training is run for a maximum of 1000 iterations.

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`.

A.5.2 Hyperparameters of contextCNN

	genre-independent			trained on news			trained on web		
slot	filter width	# conv filters	# hidden units	filter width	# conv filters	# hidden units	filter width	# conv filters	# hidden units
per:age	3	300	300	3	300	100	3	300	100
per:alternate_names	5	300	300	-	-	-	-	-	-
per:cause_of_death	5	300	1000	-	-	-	-	-	-
per:children	3	300	100	5	300	300	3	300	100
per:date_of_birth	5	300	1000	3	300	100	3	300	100
per:date_of_death	5	300	100	5	300	1000	3	300	100
per:empl_memb_of	3	300	100	3	300	1000	3	300	1000
per:location_of_birth	3	300	300	3	300	1000	3	300	100
per:loc_of_death	3	300	300	5	300	1000	5	300	100
per:loc_of_residence	3	300	100	3	300	300	5	300	300
per:origin	5	300	100	5	300	300	5	300	1000
per:schools_att	3	300	300	5	300	1000	3	300	100
per:siblings	3	300	1000	3	300	100	3	300	100
per:spouse	3	300	300	5	300	1000	3	300	300
per:title	3	300	100	3	300	1000	3	300	1000
org:alternate_names	5	300	1000	5	300	300	5	300	1000
org:date_founded	5	300	100	3	300	100	3	300	100
org:founded_by	5	300	300	3	300	100	3	300	100
org:loc_of_headqu	3	300	100	3	300	100	5	300	300
org:members	5	300	100	5	300	300	5	300	100
org:parents	5	300	300	3	300	100	5	300	100
org:top_memb_empl	3	300	100	5	300	300	5	300	300

Table A.9: Hyperparameters of contextCNN for slot filling, optimized on dev set of slot filling benchmark with genre-independent split (according to years) and genre-dependent splits. No training of genre-specific models if no genre-specific training data is available.

Table A.9 shows the hyperparameters of contextCNN which lead to the best performance on the development set. The contextCNN is introduced in Section 3.3.4. All networks (contextCNN, PCNN and PCNNext) are trained with gradient descent using a batchsize of 10 and an initial learning rate of 0.1. The learning rate is halved whenever the performance on the development set decreases. For regularization, we apply l_2 regularization with a weight of 1e-5.

A.5.3 Hyperparameters of PCNN and PCNNext

The PCNN, which we use as a baseline model, is described in Section 3.5.2 and Section 3.7.2. For training PCNN, we use the same hyperparameters for all slots: a contextsize of 120, 300 convolutional filters with filter width 5 and no fully-connected hidden layer. However, we still add the flag v denoting which relation argument appears first in the sentence to the input of the softmax layer.

For training PCNNext, we use the hyperparameters and the training schedule from contextCNN. Thus, the only difference between contextCNN and PCNNext is the time of context splitting (before or after convolution). We also apply 3-max pooling and add the flag v denoting which relation argument appears first in the sentence.

A.5.4 Hyperparameters of Multiclass Models

parameter	#N ≈ #non-N	#N ≈ 2· #non-N	#N ≈ 4· #non-N
SVMskip: C	0.3	0.1	0.01

Table A.10: Hyperparameter (penalty parameter C of error term) of multiclass SVM for slot filling.

Table A.10 provides the results of tuning the penalty parameter C of the multiclass SVM for slot filling. Its results are provided in Section 3.5.3. We additionally apply class weights in order to account for imbalanced distributions of classes in the dataset. They are multiplied to the error term of the SVM. The weights are determined automatically to be inversely proportional to the frequency of the classes: $w_c = \frac{N}{C \cdot f_c}$ with w_c being the weight for class c, C the total number of classes, N the total number of instances and f_c the frequency of class c.

	parameter	#N ≈ #non-N	#N ≈ 2· #non-N	#N ≈ 4· #non-N
noNER	filter width	3	3	3
	# conv filters	3000	3000	3000
	# hidden units	100	100	300
	learning rate	0.1	0.1	0.1

Table A.11: Hyperparameters of multiclass version of contextCNN.

model	filter width	# conv filters	# hidden units RE	# hidden units EC	learning rate
slotNER	3	300	100	100	0.10
jointNER	3	300	100	25	0.10
global	3	3000	100	100	0.03

Table A.12: Hyperparameters of type-aware derivations of contextCNN.

Table A.11 and Table A.12 provide hyperparameters of the multiclass versions of contextCNN with and without entity type information. They are presented in Section 3.5.3 and Section 5.4. All neural multiclass models are trained with a fixed learning rate, which is tuned on the development set. As before, we use minibatch training with a batchsize of 10. The size of the individual contexts is limited to 40. For regularization, we apply l_2 regularization with a weight of 1e-5.

Note that we tune the number of negative examples first using noNER (Table A.11) and then keep it fixed for the type-aware models (Table A.12) in order to use the same data for all of them.

A.6 Genre-specific Results

		train on NEWS				train on WEB			
		SVM		CNN		SVM		CNN	
		dev	test	dev	test	dev	test	dev	test
test on news	per:age	.79	.80	**.88**	**.87**	.78	.76	.85	.83
	per:alternate_names	-	-	-	-	-	-	-	-
	per:cause_of_death	-	-	-	-	-	-	-	-
	per:children	**.85**	**.86**	.78	.78	**.75**	**.80**	.00	.07
	per:date_of_birth	**.98**	**.99**	.00	.00	.95	.91	.00	.00
	per:date_of_death	**.72**	**.78**	.56	.63	.31	.48	.00	.00
	per:employee_or_member_of	.45	.44	.44	**.46**	**.47**	.45	.40	.44
	per:location_of_birth	.57	.50	.54	.43	**.59**	**.51**	.57	.48
	per:location_of_death	.65	**.67**	.62	.60	.65	.62	**.66**	.64
	per:locations_of_residence	.26	.25	.14	.10	**.39**	**.39**	.29	.28
	per:origin	.47	.37	.36	.27	**.55**	**.51**	.36	.39
	per:schools_attended	.80	**.82**	**.81**	.79	.64	.70	.63	.69
	per:siblings	**.81**	**.67**	.79	.61	**.81**	**.67**	.71	.55
	per:spouse	.74	.64	.76	**.71**	**.77**	.65	.73	.67
	per:title	.46	.42	**.49**	**.50**	.28	.23	.29	.33
	org:alternate_names	.22	.32	**.69**	.67	.65	**.70**	.66	.66
	org:date_founded	**.25**	.00	.00	.00	.00	.00	.00	.00
	org:founded_by	.84	**.85**	**.88**	.76	.85	.78	.86	.65
	org:location_of_headquarters	.51	.50	**.53**	.51	.51	**.53**	**.53**	.50
	org:members	**.60**	**.60**	.59	.51	.50	.52	.46	.51
	org:parents	**.30**	.32	.29	**.34**	.26	.33	**.30**	**.34**
	org:top_members_employees	**.56**	**.55**	**.56**	.54	.54	.53	.53	.54

Table A.13: Genre specific F1 scores. Genre specific training data (of the same sizes). Slots with no results had either no genre-specific training or evaluation data. News results.

Table A.13 and Table A.14 show slot-wise results. The most important findings are summarized in Section 3.5.2. The models are evaluated on the data presented in Table A.5. For training, the data from Table A.3 is split into genre-specific sets according to the sources of the sentences (see Section 3.4.2).

		train on NEWS				train on WEB			
		SVM		CNN		SVM		CNN	
		dev	test	dev	test	dev	test	dev	test
test on web	per:age	.33	.73	**.57**	**.83**	.00	.67	**.57**	**.83**
	per:alternate_names	-	-	-	-	-	-	-	-
	per:cause_of_death	-	-	-	-	-	-	-	-
	per:children	.59	.33	**.70**	.33	.63	**.57**	.00	.00
	per:date_of_birth	-	-	-	-	-	-	-	-
	per:date_of_death	**.67**	**.86**	**.67**	.75	.00	**.86**	.00	.00
	per:employee_or_member_of	.29	.19	**.35**	**.25**	.30	.19	.30	.20
	per:location_of_birth	-	-	-	-	-	-	-	-
	per:location_of_death	.40	.36	.33	.42	**.43**	**.47**	.31	.18
	per:locations_of_residence	.23	.07	.00	.08	**.34**	**.27**	.32	.14
	per:origin	**.59**	.40	.40	.22	**.59**	**.41**	.41	.23
	per:schools_attended	.50	.67	.36	.57	**.80**	**.80**	.00	.67
	per:siblings	-	-	-	-	-	-	-	-
	per:spouse	.52	.50	.60	.57	.56	.57	**.67**	**.62**
	per:title	.47	**.43**	**.52**	**.43**	.33	.13	.41	.31
	org:alternate_names	.27	.19	.51	.37	**.60**	**.49**	.56	.38
	org:date_founded	.00	.00	.00	.00	.00	.00	.00	.00
	org:founded_by	.46	.86	.46	.75	.60	**.89**	**.67**	.75
	org:location_of_headquarters	.39	.46	.43	.44	**.44**	**.48**	.36	.47
	org:members	.40	.44	.14	.36	**.44**	**.67**	.15	.57
	org:parents	.09	**.08**	.11	.07	.10	**.08**	**.15**	**.08**
	org:top_members_employees	.44	**.40**	.39	.34	**.47**	.36	.45	.35

Table A.14: Genre specific F1 scores. Genre specific training data (of the same sizes). Slots with no results had either no genre-specific training or evaluation data. Web results.

A.7 Slot-wise Results for Multiclass Models

	SVM						CNN					
	#N ≈ #non-N		#N ≈ 2· #non-N		#N ≈ 4· #non-N		#N ≈ #non-N		#N ≈ 2· #non-N		#N ≈ 4· #non-N	
	dev	eval	dev	eval	dev	eval	dev	eval	dev	eval	dev	eval
per:age	.77	.70	.76	.69	.77	.68	.70	.68	.70	.65	.67	.66
per:alternate_names	.0	.0	.0	.0	.0	.0	.22	.0	.11	.0	.4	.0
per:cause_of_death	.50	.0	.56	.0	.48	.0	.53	.11	.59	.20	.44	.17
per:children	.73	.39	.68	.27	.59	.31	.71	.48	.76	.38	.65	.43
per:date_of_birth	.76	.33	.33	.0	.20	.0	.93	.80	.95	.75	.82	.80
per:date_of_death	.68	.60	.59	.51	.54	.54	.64	.51	.65	.53	.57	.59
per:employee_or_member_of	.25	.21	.11	.13	.4	.3	.37	.28	.39	.27	.31	.22
per:location_of_birth	.69	.31	.50	.12	.37	.7	.68	.36	.73	.34	.71	.21
per:location_of_death	.65	.32	.66	.31	.61	.36	.62	.28	.62	.28	.60	.30
per:locations_of_residence	.6	.11	.4	.4	.1	.0	.6	.15	.6	.2	.0	.1
per:origin	.9	.4	.4	.0	.1	.0	.9	.11	.7	.0	.3	.0
per:schools_attended	.70	.57	.73	.64	.73	.65	.65	.45	.64	.43	.67	.52
per:siblings	.56	.71	.56	.67	.59	.67	.57	.73	.62	.70	.66	.63
per:spouse	.73	.45	.75	.50	.76	.57	.66	.39	.69	.46	.70	.50
per:title	.51	.51	.48	.46	.40	.41	.51	.42	.23	.19	.33	.23
org:alternate_names	.57	.59	.48	.41	.38	.42	.55	.57	.18	.31	.38	.42
org:date_founded	.61	.60	.57	.55	.55	.56	.54	.63	.55	.74	.49	.62
org:founded_by	.70	.70	.70	.69	.76	.70	.62	.71	.70	.77	.64	.69
org:location_of_headquarters	.14	.14	.5	.4	.2	.4	.25	.24	.17	.19	.5	.7
org:members	.71	.15	.69	.36	.71	.44	.64	.17	.55	.25	.52	.33
org:parents	.37	.10	.32	.21	.21	.0	.37	.14	.34	.27	.13	.0
org:top_members_employees	.45	.54	.40	.39	.26	.20	.48	.55	.53	.61	.52	.57
micro F1	.50	.43	.47	.37	.41	.34	.27	.21	.24	.18	.22	.18
macro F1	.51	.37	.46	.32	.41	.30	.52	.40	.49	.38	.45	.36

Table A.15: Slot-wise F1 results for multiclass models with different amounts of N samples.

Table A.15 shows slot-wise results for the multiclass models. Balancing the number of positive and negative samples leads to the best results on average. The results are summarized in Section 3.5.3.

A.8 Slot-wise Results of Slot Filling System

	hop 0			hop 1			all hops		
Slot	P	R	F1	P	R	F1	P	R	F1
gpe:births_in_city	13.04	8.82	10.53	100.0	0.0	0.0	13.04	7.32	9.38
gpe:births_in_country	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
gpe:births_in_stateorprovince	4.17	2.17	2.86	100.0	0.0	0.0	4.17	1.22	1.89
gpe:deaths_in_city	46.15	13.04	20.34	0.0	0.0	0.0	35.29	10.91	16.67
gpe:deaths_in_country	37.5	37.5	37.5	40.0	18.18	25.0	38.1	29.63	33.33
gpe:deaths_in_stateorprovince	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
gpe:employees_or_members	32.65	20.25	25.0	0.0	0.0	0.0	31.37	17.39	22.38
gpe:headquarters_in_city	20.59	20.0	20.29	18.75	11.54	14.29	19.7	14.94	16.99
gpe:headquarters_in_country	18.18	13.95	15.79	0.0	0.0	0.0	17.65	6.12	9.09
gpe:headquarters_in_stateorprovince	6.67	33.33	11.11	0.0	0.0	0.0	2.27	0.98	1.37
gpe:holds_shares_in	100.0	0.0	0.0	-	-	-	100.0	0.0	0.0
gpe:member_of	20.83	55.56	30.3	17.65	23.08	20.0	20.0	41.94	27.08
gpe:organizations_founded	13.33	50.0	21.05	100.0	0.0	0.0	13.33	50.0	21.05
gpe:residents_of_city	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
gpe:residents_of_country	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
gpe:residents_of_stateorprovince	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
gpe:subsidiaries	33.33	12.5	18.18	0.0	0.0	0.0	16.67	6.45	9.3
org:alternate_names	40.0	19.51	26.23	0.0	0.0	0.0	30.77	17.78	22.54
org:city_of_headquarters	45.45	50.0	47.62	0.0	0.0	0.0	16.67	33.33	22.22
org:country_of_headquarters	44.44	50.0	47.06	0.0	0.0	0.0	21.05	21.05	21.05
org:date_dissolved	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0
org:date_founded	81.82	64.29	72.0	16.67	12.5	14.29	58.82	45.45	51.28
org:employees_or_members	29.41	34.72	31.85	25.93	18.42	21.54	28.57	29.09	28.83
org:founded_by	48.65	78.26	60.0	0.0	0.0	0.0	45.0	62.07	52.17
org:holds_shares_in	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
org:member_of	5.41	12.5	7.55	0.0	0.0	0.0	5.0	12.5	7.14
org:members	20.63	16.46	18.31	8.27	21.57	11.96	12.24	18.46	14.72
org:number_of_employees_members	60.0	50.0	54.55	0.0	0.0	0.0	50.0	42.86	46.15
org:organizations_founded	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0
org:parents	7.69	33.33	12.5	0.0	0.0	0.0	6.67	28.57	10.81
org:political_religious_affiliation	50.0	33.33	40.0	100.0	0.0	0.0	50.0	33.33	40.0
org:shareholders	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
org:stateorprovince_of_headquarters	63.64	70.0	66.67	0.0	0.0	0.0	50.0	58.33	53.85
org:students	35.71	10.0	15.63	20.83	10.87	14.29	26.32	10.42	14.93
org:subsidiaries	22.5	22.5	22.5	21.43	100.0	35.29	22.22	27.91	24.74
org:top_members_employees	27.78	52.63	36.36	100.0	0.0	0.0	27.78	37.04	31.75
org:website	100.0	33.33	50.0	-	-	-	100.0	33.33	50.0

Table A.16: Slot-wise CSLDC max scores from best CIS entry (baseline + CNN + EL) in 2015 slot filling assessments; part 1: gpe and org slots.

Table A.16 and Table A.17 show slot-wise results of the slot filling pipeline using the best setup from Section 3.5.4. Aggregated results and the most important findings are given in Section 3.5.4.

	hop 0			hop 1			all hops		
Slot	P	R	F1	P	R	F1	P	R	F1
per:age	100.0	70.0	82.35	75.0	12.5	21.43	90.91	29.41	44.44
per:alternate_names	100.0	23.81	38.46	100.0	0.0	0.0	100.0	20.0	33.33
per:cause_of_death	33.33	50.0	40.0	100.0	0.0	0.0	33.33	33.33	33.33
per:charges	50.0	7.69	13.33	50.0	12.5	20.0	50.0	9.52	16.0
per:children	58.33	100.0	73.68	0.0	0.0	0.0	36.84	46.67	41.18
per:cities_of_residence	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
per:city_of_birth	75.0	42.86	54.55	100.0	0.0	0.0	75.0	18.75	30.0
per:city_of_death	100.0	0.0	0.0	100.0	100.0	100.0	100.0	16.67	28.57
per:countries_of_residence	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
per:country_of_birth	50.0	50.0	50.0	-	-	-	50.0	50.0	50.0
per:country_of_death	100.0	100.0	100.0	0.0	0.0	0.0	57.14	80.0	66.67
per:date_of_birth	83.33	62.5	71.43	100.0	0.0	0.0	83.33	55.56	66.67
per:date_of_death	50.0	11.11	18.18	100.0	0.0	0.0	50.0	6.67	11.76
per:employee_or_member_of	21.25	34.69	26.36	0.0	0.0	0.0	20.48	29.82	24.29
per:holds_shares_in	100.0	0.0	0.0	-	-	-	100.0	0.0	0.0
per:organizations_founded	60.0	50.0	54.55	100.0	100.0	100.0	63.64	53.85	58.33
per:origin	23.08	18.75	20.69	0.0	0.0	0.0	3.06	10.34	4.72
per:other_family	60.0	75.0	66.67	100.0	0.0	0.0	60.0	75.0	66.67
per:parents	50.0	66.67	57.14	0.0	0.0	0.0	46.15	66.67	54.55
per:religion	100.0	25.0	40.0	100.0	0.0	0.0	100.0	16.67	28.57
per:schools_attended	60.0	69.23	64.29	-	-	-	60.0	69.23	64.29
per:siblings	43.75	58.33	50.0	100.0	0.0	0.0	43.75	35.0	38.89
per:spouse	50.0	18.75	27.27	-	-	-	50.0	18.75	27.27
per:stateorprovince_of_birth	50.0	50.0	50.0	100.0	0.0	0.0	50.0	50.0	50.0
per:stateorprovince_of_death	66.67	57.14	61.54	100.0	0.0	0.0	66.67	50.0	57.14
per:statesorprovinces_of_residence	100.0	0.0	0.0	100.0	0.0	0.0	100.0	0.0	0.0
per:title	50.0	34.04	40.51	26.03	21.35	23.46	33.33	25.74	29.05
per:top_member_employee_of	35.71	55.56	43.48	100.0	0.0	0.0	35.71	55.56	43.48

Table A.17: Slot-wise CSLDC max scores from best CIS entry (baseline + CNN + EL) in 2015 slot filling assessments; part 2: per slots.

Appendix B

Additional Material for Uncertainty Detection

B.1 Hyperparameters

		CNN				GRU		
		# conv filters	filter width	# hidden units	# att hidden units	# gru hidden units	# hidden units	# att hidden units
Wikipedia	(3)/(2)	200	3	200	-	10	100	-
	(6)/(4)	100	3	500	-	10	100	-
	(7)/(5)	200	3	200	-	10	200	-
	(11)/(8)	200	3	200	-	10	100	-
	(12)/(9)	200	3	200	200	30	200	200
	(13)/(10)	100	3	200	200	10	200	100
Biomedical	(3)/(2)	200	3	500	-	10	500	-
	(6)/(4)	100	3	200	-	10	500	-
	(7)/(5)	100	3	500	-	10	50	-
	(11)/(8)	200	3	200	-	10	50	-
	(12)/(9)	200	3	500	100	30	100	200
	(13)/(10)	200	3	50	100	10	50	200

Table B.1: Result of hyperparameter tuning for CNN and GRU models.

Table B.1 provides the hyperparameters of the models we train and evaluate on uncertainty detection. They are described in Section 4.2. The numbers in the first column correspond to the model numbers in Section 4.4.

For training the CNNs, we use a fixed learning rate of 0.03. For training the GRUs, we apply Adagrad (see Section 2.2.4) with an initial learning rate of 0.1. In all cases, we use a batchsize of 10 and apply l_2 regularization with a weight of 1e-5. The number of training epochs is determined by peak performances on the development set.

B.2 Analysis of Selection Mechanisms

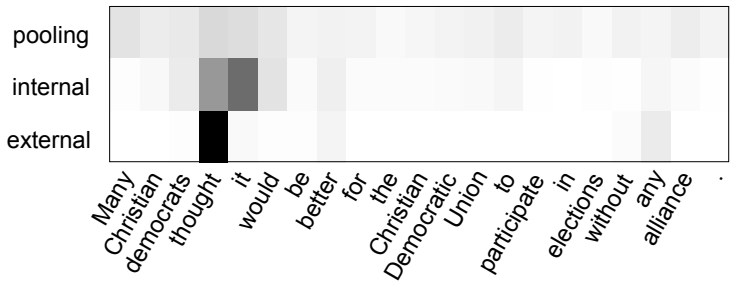


Figure B.1: Pooling vs. internal attention vs. external attention.

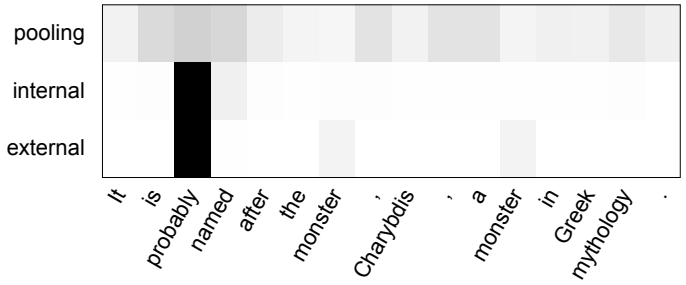


Figure B.2: Pooling vs. internal attention vs. external attention.

Figure B.1 and Figure B.2 compare pooling, internal attention and external attention for randomly picked examples from the test set. As in Section 4.5.1, pooling extracts values from all over the sentence while internal and external attention learn to focus on words which can indicate uncertainty (e.g., “thought” or “probably”).

Appendix C

Additional Material for Type-aware Relation Extraction

C.1 Hyperparameters of Pipeline and Jointly Trained Models

model	# epochs	# conv filters	hidden units	EC hidden units
`contextCNN` (no types)	8	100	100	–
+ `BINARY`	8	100	100	–
+ `BINARY-HIDDEN`	11	100	100	80
+ `PREDICTED-HIDDEN`	11	100	100	80
+ `WEIGHTED`	11	100	100	100
`JOINT`	11	100	100	80

Table C.1: Hyperparameters of type-aware models for relation extraction on ClueWeb.

Table C.1 provides the hyperparameters of the pipeline and joint models for relation extraction. The models are presented in Section 5.2.1.

For the three contexts for relation extraction, we use a contextsize of 25 and a filter width of 3. For the entity classification parts, we use a contextsize of 10 and 300 convolutional filters of widths 1–4. The hidden unit size is 600.

All models are trained with a learning rate of 0.03 and a batchsize of 50. To avoid overfitting, we apply l_2 regularization with a weight of 1e-5.

C.2 Relation-specific Results of Pipeline and Jointly Trained Models

relation	contextCNN	BINARY	BINARY-HIDDEN
government.government_agency.jurisdiction	62.26	66.33	65.63
government.us_president.vice_president	66.67	50.00	34.29
location.location.containedby	49.04	51.73	55.21
organization.organization_founder.organizations_founded	67.54	52.46	54.79
organization.organization.place_founded	42.38	50.42	48.63
people.deceased_person.place_of_death	36.13	48.35	55.17
people.person.children	53.20	39.34	55.17
people.person.nationality	52.82	60.59	63.91
people.person.place_of_birth	46.76	47.56	49.82
people.person.religion	78.73	80.13	71.34

Table C.2: Relation-wise results of type-aware models on ClueWeb (part 1).

relation	PREDICTED-HIDDEN	WEIGHTED	JOINT
government.government_agency.jurisdiction	66.25	62.65	70.41
government.us_president.vice_president	37.04	66.67	21.74
location.location.containedby	68.99	64.19	33.45
organization.organization_founder.organizations_founded	56.18	58.54	62.30
organization.organization.place_founded	42.48	38.34	58.51
people.deceased_person.place_of_death	50.78	53.73	67.10
people.person.children	46.25	53.09	53.48
people.person.nationality	68.06	66.50	54.65
people.person.place_of_birth	44.90	48.44	49.70
people.person.religion	72.34	66.34	75.94

Table C.3: Relation-wise results of type-aware models on ClueWeb (part 2).

Table C.2 and Table C.3 show relation-wise results of the type-aware models. The results are visualized in Figure 5.6 and discussed in Section 5.2.4.

C.3 Hyperparameters of Neural Structured Prediction Models

	setup 1		setup 2		setup 3	
	softmax	CRF	softmax	CRF	softmax	CRF
filter width $\text{CNN}_{context}$	3	3	3	3	3	3
filter width CNN_{entity}	2	2	2	2	2	2
# filters $\text{CNN}_{context}$	500	200	500	500	500	500
# filters $\text{CNN}_{entities}$	100	50	100	100	100	100
# hidden units RE	100	100	100	200	100	100
# hidden units EC	50	50	50	50	50	50

Table C.4: Hyperparameters of globally normalized models.

Table C.4 provides hyperparameters of the neural structured prediction models presented in Section 5.3.1. The different setups are described in Section 5.3.3.

We use a contextsize of 120 and an entity size of 20. All models are trained with a learning rate of 0.1 and a batchsize of 10. To avoid overfitting, we apply l_2 regularization with a weight of 1e-4. The number of epochs is determined via early stopping on the development set.

Bibliography

Heike Adel and Hinrich Schütze. TAC KBP 2014 slot filling shared task: Baseline system for investigating coreference. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Heike Adel and Hinrich Schütze. CIS at TAC cold start 2015: Neural networks and coreference resolution for slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Heike Adel and Hinrich Schütze. Impact of coreference resolution on slot filling. *Computing Research Repository (arXiv.org), arXiv:1710.09753*, 2017a.

Heike Adel and Hinrich Schütze. Global normalization of convolutional neural networks for joint entity and relation classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 1724–1730. Association for Computational Linguistics, 2017b.

Heike Adel and Hinrich Schütze. Exploring different dimensions of attention for uncertainty detection. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 22–34. Association for Computational Linguistics, 2017c.

Heike Adel, Benjamin Roth, and Hinrich Schütze. Comparing convolutional neural networks to traditional models for slot filling. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 828–838. Association for Computational Linguistics, 2016.

Alicia Ageno, Pere R. Comas, Ali M. Naderi, Horacio Rodríguez, and Jordi Turmo. The TALP participation at TAC-KBP 2013. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning*, pages 2091–2100. International Machine Learning Society, 2016.

Aurore De Amaral. Ontology-guided extraction for KBP 2014 slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Vaino Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning*, pages 173–182. International Machine Learning Society, 2016.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452. Association for Computational Linguistics, 2016.

Gabor Angeli, Arun Chaganty, Angel Chang, Kevin Reschke, Julie Tibshirani, Jean Y. Wu, Osbert Bastani, Keith Siilats, and Christopher D. Manning. Stanford's 2013 KBP system. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D. Manning, Christopher Re, Julie Tibshirani, Jean Y. Wu, Sen Wu, and Ce Zhang. Stanford's 2014 slot filling systems. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014a.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D. Manning. Combining distant and partial supervision for relation extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 1556–1567. Association for Computational Linguistics, 2014b.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D. Manning. Bootstrapped self training for knowledge base population. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735. Springer, 2007.

Isabelle Augenstein, Andreas Vlachos, and Diana Maynard. Extracting relations between non-standard entities using distant supervision and imitation learning. In *Conference*

on Empirical Methods in Natural Language Processing, pages 747–757. Association for Computational Linguistics, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

Kathryn Baker, Michael Bloodgood, Bonnie J. Dorr, Chris Callison-Burch, Nathaniel W. Filardo, Christine Piatko, Lori Levin, and Scott Miller. Use of modality and negation in semantically-informed syntactic MT. In *Computational Linguistics*, volume 38(2), pages 411–438. MIT Press, 2012.

Zuyi Bao, Yao Lu, Tian Tian, Dongyun Liang, Yinge Zhao, Zhichao Li, Pengda Qin, and Weiran Xu. DoughnutPRIS at TAC KBP 2016. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Annual Meeting of the Association for Computational Linguistics*, pages 238–247. Association for Computational Linguistics, 2014.

Farah Benamara, Baptiste Chardon, Yannick Mathieu, Vladimir Popescu, and Nicholas Asher. How do negation and modality impact on opinions? In *Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, pages 10–18. Association for Computational Linguistics, 2012.

Yoshua Bengio. Learning deep architectures for AI. In *Foundations and Trends® in Machine Learning*, volume 2(1), pages 1–127. Now Publishers, Inc., 2009.

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In *Large-scale Kernel Machines*, volume 34(5), pages 1–41. MIT Press, 2007.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. In *Journal of Machine Learning Research*, volume 3, pages 1137–1155. MIT Press, 2000.

Yinon Bentor, Amelia Harrison, Shruti Bhosale, and Raymond Mooney. University of Texas at Austin KBP 2013 slot filling system: Bayesian logic programs for textual inference. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Yinon Bentor, Vidhoon Viswanathan, and Raymond Mooney. University of Texas at Austin KBP 2014 slot filling system: Bayesian logic programs for textual inference. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A CPU and GPU math expression compiler. In *Python for Scientific Computing Conference*, pages 3 – 10, 2010.

Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Conference on Computational Learning Theory*, pages 92–100. Association for Computational Linguistics, 1998.

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

Léon Bottou, Yoshua Bengio, and Yann Le Cun. Global training of document processing systems using graph transformer networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 489–494. Institute of Electrical and Electronics Engineers, 1997.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. In *Computational Linguistics*, volume 18(4), pages 467–479. MIT Press, 1992.

Razvan Bunescu and Raymond Mooney. Learning to extract relations from the web using minimal supervision. In *Annual Meeting of the Association for Computational Linguistics*, pages 576–583. Association for Computational Linguistics, 2007.

Lorna Byrne and John Dunnion. UCD IIRG at TAC 2012. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Lorna Byrne, Caroline Fenlon, and John Dunnion. UCD IIRG at TAC KBP 2013. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Lorna Byrne, Caroline Fenlon, and John Dunnion. UCD IIRG at TAC KBP 2014. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Jiong Cai, Yong Jiang, and Kewei Tu. CRF autoencoder for unsupervised dependency parsing. In *Conference on Empirical Methods in Natural Language Processing*, pages 1639–1644. Association for Computational Linguistics, 2017.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. Bidirectional recurrent convolutional neural network for relation classification. In *Annual Meeting of the Association for Computational Linguistics*, pages 756–765. Association for Computational Linguistics, 2016.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI Conference on Artificial Intelligence*, pages 1306–1313. Association for the Advancement of Artificial Intelligence, 2010.

Rich Caruana. Multitask learning. In *Learning to Learn*, pages 95–133. Springer, 1998.

Arun T. Chaganty, Ashwin P. Paranjape, Percy Liang, and Christopher D. Manning. Importance sampling for unbiased on-demand evaluation of knowledge base population. In *Conference on Empirical Methods in Natural Language Processing*, pages 1049–1059. Association for Computational Linguistics, 2017.

Hans Chalupsky. English slot filling with the knowledge resolver system. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Hans Chalupsky. English slot filling with the knowledge resolver system for TAC-KBP 2014. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Haw-Shiuan Chang, Abdurrahman Munir, Ao Liu, Johnny Tian-Zheng Wei, Aaron Traylor, Ajay Nagesh, Nicholas Monath, Patrick Verga, Emma Strubell, and Andrew McCallum. Extracting multilingual relations under limited resources: TAC 2016 cold-start KB construction and slot-filling using compositional universal schema. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. ABC-CNN: An attention based convolutional neural network for visual question answering. *Computing Research Repository (arXiv.org), arXiv:1511.05960*, 2015.

Liwei Chen, Yansong Feng, Songfang Huang, Yong Qin, and Dongyan Zhao. Encoding relation requirements for relation extraction via joint inference. In *Annual Meeting of the Association for Computational Linguistics*, pages 818–827. Association for Computational Linguistics, 2014.

Jason Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. In *Transactions of the Association for Computational Linguistics*, volume 4, pages 357–370. Association for Computational Linguistics, 2016.

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

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Computing Research Repository (arXiv.org), arXiv:1412.3555*, 2014.

David Clausen. HedgeHunter: A system for hedge detection and uncertainty classification. In *Computational Natural Language Learning*, pages 120–125. Association for Computational Linguistics, 2010.

Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Annual Meeting of the Association for Computational Linguistics*, pages 111–118. Association for Computational Linguistics, 2004.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*, pages 160–167. International Machine Learning Society, 2008.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. In *Journal of Machine Learning Research*, volume 12, pages 2493–2537. Microtome Publishing, 2011.

Mike Conway, Son Doan, and Nigel Collier. Using hedges to enhance a disease outbreak report text mining system. In *BioNLP Workshop*, pages 142–143. Association for Computational Linguistics, 2009.

Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Intelligent Systems for Molecular Biology*, volume 1999, pages 77–86. International Society for Computational Biology, 1999.

Noa P. Cruz, Maite Taboada, and Ruslan Mitkov. A machine-learning approach to negation and speculation detection for sentiment analysis. In *Journal of the Association for Information Science and Technology*, volume 67(9), pages 2118–2136. Wiley-Blackwell, 2016.

Aron Culotta, Andrew McCallum, and Jonathan Betz. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 296–303. Association for Computational Linguistics, 2006.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 132–141. Association for Computational Linguistics, 2017.

DBpedia. Statistics of DBpedia, 2015. `http://wiki.dbpedia.org/services-resources/datasets/dataset-2015-10/dataset-2015-10-statistics`.

Marie-Catherine De Marneffe, Christopher D. Manning, and Christopher Potts. Did it happen? The pragmatic complexity of veridicality assessment. In *Computational Linguistics*, volume 38(2), pages 335–367. MIT Press, 2012.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. In *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38. JSTOR, 1977.

Noa P. Cruz Díaz, Maite Taboada, and Ruslan Mitkov. A machine-learning approach to negation and speculation detection for sentiment analysis. In *Journal of the Association for Information Science and Technology*, volume 67(9), pages 2118–2136. Wiley-Blackwell, 2016.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. In *Artificial Intelligence*, volume 89(1), pages 31–71. Elsevier, 1997.

Trinh–Minh–Tri Do and Thierry Artieres. Neural conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 177–184. Proceedings of Machine Learning Research, 2010.

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. The automatic content extraction (ACE) program — tasks, data, and evaluation. In *International Conference on Language Resources and Evaluation*, pages 837–840. European Language Resources Association, 2004.

Pedro Domingos. A few useful things to know about machine learning. In *Communications of the ACM*, volume 55(10), pages 78–87. Association for Computing Machinery, 2012.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 626–634. Association for Computational Linguistics, 2015.

Lan Du, Anish Kumar, Mark Johnson, and Massimiliano Ciaramita. Using entity information from a knowledge base to improve relation extraction. In *ALTA Workshop*, pages 31–38. Australasian Language Technology Association, 2015.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*, volume 12, pages 2121–2159. Microtome Publishing, 2011.

Greg Durrett and Dan Klein. Neural CRF parsing. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 302–312. Association for Computational Linguistics, 2015.

Javid Ebrahimi and Dejing Dou. Chain based RNN for relation classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249. Association for Computational Linguistics, 2015.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? In *Journal of Machine Learning Research*, volume 11, pages 625–660. Microtome Publishing, 2010.

Katrin Erk. Vector space models of word meaning and phrase meaning: A survey. In *Language and Linguistic Compass*, volume 6(10), pages 635–653. Wiley Online Library, 2012.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y. Chang. Distant supervision for relation extraction with matrix completion. In *Annual Meeting of the Association for Computational Linguistics*, pages 839–849. Association for Computational Linguistics, 2014.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. In *Journal of Machine Learning Research*, volume 9, pages 1871–1874. Microtome Publishing, 2008.

Richárd Farkas and György Szarvas. Automatic construction of rule-based ICD-9-CM coding systems. In *BMC Bioinformatics*, volume 9(3). BioMed Central, 2008.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. The CoNLL-2010 shared task: Learning to detect hedges and their scope in natural language text. In *Computational Natural Language Learning*, pages 1–12. Association for Computational Linguistics, 2010.

Oliver Ferschke, Torsten Zesch, and Iryna Gurevych. Wikipedia revision toolkit: Efficiently accessing Wikipedia's edit history. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: System Demonstrations*, pages 97–102. Association for Computational Linguistics, 2011.

Matthias Feys, Lucas Sterckx, Laurent Mertens, Johannes Deleu, Thomas Demeester, and Chris Develder. Ghent University-IBCN participation in TAC-KBP 2014 slot filling and cold start tasks. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Annual Meeting of the Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

John R. Firth. A synopsis of linguistic theory, 1930–1955. In *Studies in Linguistic Analysis*. Basil Blackwell, 1957.

Carol Friedman, Philip O. Alderson, John H.M. Austin, James J. Cimino, and Stephen B. Johnson. A general natural-language text processor for clinical radiology. In *Journal of the American Medical Informatics Association*, volume 1(2), pages 161–174. BMJ Group BMA House, 1994.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, 2013.

Viola Ganter and Michael Strube. Finding hedges by chasing weasels: Hedge detection using Wikipedia tags and shallow linguistic features. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 173–176. Association for Computational Linguistics, 2009.

Matthew Gardner. *Reading and reasoning with knowledge graphs*. PhD thesis, Carnegie Mellon University, 2015.

Guillermo Garrido, Anselmo Peñas, and Bernardo Cabaleiro. UNED slot filling and temporal slot filling systems at TAC KBP 2013. System description. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Maria Georgescul. A hedgehop over a max-margin framework using hedge cues. In *Computational Natural Language Learning*, pages 26–31. Association for Computational Linguistics, 2010.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. In *Neural Computation*, volume 12(10), pages 2451–2471. MIT Press, 2000.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2003.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. Multilingual language processing from bytes. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306. Association for Computational Linguistics, 2016.

Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Relation extraction and the influence of automatic named-entity recognition. In *ACM Transactions on Speech and Language Processing*, volume 5(1), pages 2:1–2:26. Association for Computing Machinery, 2007.

Michael Glass and Alfio Gliozzo. A dataset for web-scale knowledge base population. In *The Semantic Web*, pages 256–271. Springer, 2018.

Yoav Goldberg and Graeme Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

Edgar Gonzàlez, Horacio Rodríguez, Jordi Turmo, Pere R. Comas, Ali M. Naderi, Alicia Ageno, Emili Sapena, Marta Vila, and M. Antònia Martí. The TALP participation at TAC-KBP 2012. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Matthew R. Gormley, Mo Yu, and Mark Dredze. Improved relation extraction with feature-rich compositional embedding models. In *Conference on Empirical Methods in Natural Language Processing*, pages 1774–1784. Association for Computational Linguistics, 2015.

Édouard Grave. A convex relaxation for weakly supervised relation extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 1580–1590. Association for Computational Linguistics, 2014.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772. International Machine Learning Society, 2014.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *International Conference on Computational Linguistics*, pages 2537–2547. International Committee on Computational Linguistics, 2016.

Zellig S. Harris. Distributional structure. In *Word*, volume 10(23), pages 146–162. International Linguistic Association, 1954.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. Simple customization of recursive neural networks for semantic relation classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376. Association for Computational Linguistics, 2013.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. Task-oriented learning of word embeddings for semantic relation classification. In *Computational Natural Language Learning*, pages 268–278. Association for Computational Linguistics, 2015.

Erik Hatcher and Otis Gospodnetic. *Lucene in action*. Manning Publications, 2004.

Xiaodong He and David Golub. Character-level question answering with attention. In *Conference on Empirical Methods in Natural Language Processing*, pages 1598–1607. Association for Computational Linguistics, 2016.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *International Workshop on Semantic Evaluation*, pages 33–38. Association for Computational Linguistics, 2010.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701. Neural Information Processing Systems Foundation, Inc., Curran Associates, Inc., 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural Computation*, volume 9(8), pages 1735–1780. MIT Press, 1997.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. In *Artificial Intelligence*, volume 194, pages 28–61. Elsevier, 2013.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550. Association for Computational Linguistics, 2011.

Yu Hong, Xiaobin Wang, Yadong Chen, Jian Wang, Tongtao Zhang, Jin Zheng, Dian Yu, Qi Li, Boliang Zhang, Han Wang, Xiaoman Pan, and Heng Ji. RPI_BLENDER TAC-KBP2014 knowledge base population system. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Yu Hong, Yingying Qiu, Zengzhuang Xu, Wenxuan Zhou, Jian Tang, Xiaobin Wang, Liang Yao, and Jianmin Yao. SoochowNLP system description for 2016 KBP slot filling and nugget detection tasks. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. In *Technical Report*. National Taiwan University, 2003.

Lifu Huang, Avirup Sil, Heng Ji, and Radu Florian. Improving slot filling performance with attentive neural networks on dependency structures. In *Conference on Empirical Methods in Natural Language Processing*, pages 2578–2587. Association for Computational Linguistics, 2017.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *Computing Research Repository (arXiv.org), arXiv:1508.01991*, 2015.

Pierre-Antoine Jean, Sébastien Harispe, Sylvie Ranwez, Patrice Bellot, and Jacky Montmain. Uncertainty detection in natural language: A probabilistic model. In *International*

Conference on Web Intelligence, Mining and Semantics. Association for Computing Machinery, 2016.

Feng Ji, Xipeng Qiu, and Xuanjing Huang. Detecting hedge cues and their scopes with average perceptron. In *Computational Natural Language Learning*, pages 32–39. Association for Computational Linguistics, 2010.

Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1148–1158. Association for Computational Linguistics, 2011.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. Overview of the TAC 2011 knowledge base population track. In *Text Analysis Conference*. National Institute of Standards and Technology, 2011.

Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. Relation extraction with multi-instance multi-label convolutional neural networks. In *International Conference on Computational Linguistics*, pages 1471–1480. International Committee on Computational Linguistics, 2016.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Annual Meeting of the Association for Computational Linguistics*, pages 655–665. Association for Computational Linguistics, 2014.

Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, page 22. Association for Computational Linguistics, 2004.

Lauri Karttunen. Presuppositions of compound sentences. In *Linguistic Inquiry*, volume 4(2), pages 169–196. JSTOR, 1973.

Lauri Karttunen and Annie Zaenen. Veridicity. In *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), 2005.

Rohit J. Kate and Raymond Mooney. Joint entity and relation extraction using card-pyramid parsing. In *Computational Natural Language Learning*, pages 203–212. Association for Computational Linguistics, 2010.

Halil Kilicoglu and Sabine Bergler. Recognizing speculative language in biomedical research articles: A linguistically motivated perspective. In *Workshop on Current Trends in Biomedical Natural Language Processing*, pages 46–53. Association for Computational Linguistics, 2008.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of BioNLP'09 shared task on event extraction. In *Workshop on Current Trends in Biomedical Natural Language Processing*, pages 1–9. Association for Computational Linguistics, 2009.

Yoon Kim. Convolutional neural networks for sentence classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics, 2014.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

Paul Kiparsky and Carol Kiparsky. Fact. In *Progress in Linguistics*, pages 143–173. The Hague: Mouton, 1970.

Johannes Kirschnick, Holmer Hemsen, and Volker Markl. JEDI: Joint entity and relation detection using type inference. In *Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66. Association for Computational Linguistics, 2016.

Bryan Kisiel, Justin Betteridge, Matt Gardner, Jayant Krishnamurthy, Ndapa Nakashole, Mehdi Samadi, Partha Talukdar, Derry Wijaya, and Tom Mitchell. CMUML system for KBP 2013 slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Bryan Kisiel, Justin Betteridge, Matt Gardner, Jayant Krishnamurthy, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Abulhair Saparov, Partha Talukdar, Derry Wijaya, and Tom Mitchell. CMUML system for KBP 2014 slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Bryan Kisiel, Bill McDowell, Matt Gardner, Ndapandula Nakashole, Emmanouil A. Platanios, Abulhair Saparov, Shashank Srivastava, Derry Wijaya, and Tom Mitchell. CMUML system for KBP 2015 cold start slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. Improving sentence compression by learning to predict gaze. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1528–1533. Association for Computational Linguistics, 2016.

Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957. Neural Information Processing Systems Foundation, Inc., Morgan Kaufmann Publishers, 1992.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic

memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387. International Machine Learning Society, 2016.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289. International Machine Learning Society, 2001.

George Lakoff. Hedges: A study in meaning criteria and the logic of fuzzy concepts. In *Contemporary Research in Philosophical Logic and Linguistic Semantics*, pages 221–271. Springer, 1975.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics, 2016.

Yann LeCun. Generalization and network design strategies. In *Connectionism in Perspective*, pages 143–155. Elsevier, 1989.

Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, volume 1(4), pages 541–551. MIT Press, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86(11), pages 2278–2324. Institute of Electrical and Electronics Engineers, 1998.

Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–48. Springer, 2012.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Global neural CCG parsing with optimality guarantees. In *Conference on Empirical Methods in Natural Language Processing*, pages 2366–2376. Association for Computational Linguistics, 2016.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics, 2017.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10(8), pages 707–710. MAIK Nauka/Interperiodica and Springer, 1966.

Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *Annual Meeting of the Association for Computational Linguistics*, pages 402–412. Association for Computational Linguistics, 2014.

Xinxin Li, Jianping Shen, Xiang Gao, and Xuan Wang. Exploiting rich features for detecting hedges and their scope. In *Computational Natural Language Learning*, pages 78–83. Association for Computational Linguistics, 2010.

Xiujun Li, Wei Gao, and Jude W. Shavlik. Detecting semantic uncertainty by learning hedge cues in sentences using an HMM. In *SIGIR Workshop on Semantic Matching in Information Retrieval*. Special Interest Group on Information Retrieval, 2014.

Yan Li, Sijia Chen, Zhihua Zhou, Jie Yin, Hao Luo, Liyin Hong, Weiran Xu, Guang Chen, and Jun Guo. PRIS at TAC2012 KBP track. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Yan Li, Yichang Zhang, Doyu Li, Xin Tong, Jianlong Wang, Naiche Zuo, Ying Wang, Weiran Xu, Guang Chen, and Jun Guo. PRIS at knowledge base population 2013. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Marc Light, Xin Ying Qiu, and Padmini Srinivasan. The language of bioscience: Facts, speculations, and statements in between. In *BioLINK: Linking Biological Literature, Ontologies and Databases*, pages 17–24. Association for Computational Linguistics, 2004.

Seongwoo Lim, Jiyeon Han, Kyowoon Lee, and Jaesik Cho. SAIL (UNIST) team participation in KBP 2016 cold start slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016.

Hailun Lin, Zeya Zhao, Yantao Jia, Yuanzhuo Wang, Jinhua Xiong, Xiaojing Li, Yuxiao Chang, Manling Li, Hongbo Xu, and Xueqi Cheng. OpenKN at TAC KBP 2014. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Annual Meeting of the Association for Computational Linguistics*, pages 2124–2133. Association for Computational Linguistics, 2016.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*, 2017.

Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *AAAI Conference on Artificial Intelligence*, pages 94–100. Association for the Advancement of Artificial Intelligence, 2012.

Fang Liu and Jun Zhao. Sweat2012: Pattern based English slot filling system for knowledge base population at TAC 2012. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *International Conference on Computational Linguistics*, pages 2107–2116. International Committee on Computational Linguistics, 2014.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. A dependency-based neural network for relation classification. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 285–290. Association for Computational Linguistics, 2015.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Annual Meeting of the Association for Computational Linguistics*, pages 1064–1074. Association for Computational Linguistics, 2016.

Andrew MacKinlay, David Martinez, and Timothy Baldwin. Biomedical event annotation with CRFs and precision grammars. In *Workshop on Current Trends in Biomedical Natural Language Processing*, pages 77–85. Association for Computational Linguistics, 2009.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. YAGO3: A knowledge base from multilingual Wikipedias. In *Conference on Innovative Data Systems Research*. CIDR Conference, 2015.

Christopher Malon, Bing Bai, and Kazi Saidul Hasan. Slot-filling by substring extraction at TAC KBP 2012 (Team Papelo). In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics, 2014.

Héctor Martínez Alonso and Barbara Plank. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 44–53. Association for Computational Linguistics, 2017.

David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 152–159. Association for Computational Linguistics, 2006a.

David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *International Conference on Computational Linguistics*, pages 337–344. International Committee on Computational Linguistics, 2006b.

Pablo N. Mendes, Max Jakob, and Christian Bizer. DBpedia: A multilingual cross-domain knowledge base. In *International Conference on Language Resources and Evaluation*, pages 1813–1817. European Language Resources Association, 2012.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. Encoding source language with convolutional neural network for machine translation. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 20–30. Association for Computational Linguistics, 2015.

Rada Mihalcea. Co-training and self-training for word sense disambiguation. In *Computational Natural Language Learning*, pages 33–40. Association for Computational Linguistics, 2004.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Annual Conference of the International Speech Communication Association*, pages 1045–1048. International Speech Communication Association, 2010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Workshop at International Conference on Learning Representations*, 2013.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409. Association for Computational Linguistics, 2016.

George A. Miller and Walter G. Charles. Contextual correlates of semantic similarity. In *Language and Cognitive Processes*, volume 6(1), pages 1–28. Taylor & Francis, 1991.

Bonan Min and Ralph Grishman. Challenges in the knowledge base population slot filling task. In *International Conference on Language Resources and Evaluation*, pages 1137–1142. European Language Resources Association, 2012.

Bonan Min, Xiang Li, Ralph Grishman, and Ang Sun. New York University 2012 system for KBP slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782. Association for Computational Linguistics, 2013.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1003–1011. Association for Computational Linguistics, 2009.

Makoto Miwa and Mohit Bansal. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116. Association for Computational Linguistics, 2016.

Makoto Miwa and Yutaka Sasaki. Modeling joint entity and relation extraction with table representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1858–1869. Association for Computational Linguistics, 2014.

Roser Morante and Walter Daelemans. Learning the scope of hedge cues in biomedical texts. In *Workshop on Current Trends in Biomedical Natural Language Processing*, pages 28–36. Association for Computational Linguistics, 2009.

Ajay Nagesh, Gholamreza Haffari, and Ganesh Ramakrishnan. Noisy or-based model for relation extraction using distant supervision. In *Conference on Empirical Methods in Natural Language Processing*, pages 1937–1941. Association for Computational Linguistics, 2014.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning*, pages 807–814. International Machine Learning Society, 2010.

Thien Huu Nguyen and Ralph Grishman. Relation extraction: Perspective from convolutional neural networks. In *Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48. Association for Computational Linguistics, 2015.

Thien Huu Nguyen and Ralph Grishman. Combining neural networks and log-linear models to improve relation extraction. In *Workshop on Deep Learning for Artificial Intelligence*. IJCAI Conference Division, 2016.

Thien Huu Nguyen, Yifan He, Maria Pershina, Xiang Li, and Ralph Grishman. New York University 2014 knowledge base population systems. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *ACM International Conference on Information and Knowledge Management*, pages 86–93. Association for Computing Machinery, 2000.

Steven J. Nowlan and Geoffrey E. Hinton. Simplifying neural networks by soft weight-sharing. In *Neural Computation*, volume 4(4), pages 473–493. MIT Press, 1992.

Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Christina Lioma. Terrier: A high performance and scalable information retrieval platform. In *SIGIR Workshop on Open Source Information Retrieval*, pages 18–24. Association for Computing Machinery, 2006.

Arzucan Özgür and Dragomir R. Radev. Detecting speculations and their scopes in scientific text. In *Conference on Empirical Methods in Natural Language Processing*, pages 1398–1407. Association for Computational Linguistics, 2009.

Frank Robert Palmer. *Mood and modality*. Cambridge University Press, 2001.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Annual Meeting of the Association for Computational Linguistics*, pages 271–278. Association for Computational Linguistics, 2004.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318. International Machine Learning Society, 2013.

Sachin Pawar, Pushpak Bhattacharyya, and Girish Palshikar. End-to-end relation extraction using neural networks and Markov logic networks. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 818–827. Association for Computational Linguistics, 2017.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From Freebase to Wikidata: The great migration. In *International Conference on World Wide Web*, pages 1419–1428. International World Wide Web Conference Committee, 2016.

Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. Towards neural network–based reasoning. In *Computing Research Repository (arXiv.org), arXiv:1508.05508*, 2015.

Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. In *Information Processing & Management*, volume 42(4), pages 963–979. Elsevier, 2006.

Jian Peng, Liefeng Bo, and Jinbo Xu. Conditional neural fields. In *Advances in Neural Information Processing Systems*, pages 1419–1427. Neural Information Processing Systems Foundation, Inc., Curran Associates, Inc., 2009.

Maria Pershina, Bonan Min, Wei Xu, and Ralph Grishman. Infusion of labeled data into distant supervision for relation extraction. In *Annual Meeting of the Association for Computational Linguistics*, pages 732–738. Association for Computational Linguistics, 2014.

Francesco Piccinno and Paolo Ferragina. From TagME to WAT: A new entity annotator. In *International Workshop on Entity Recognition & Disambiguation*, pages 55–62. Association for Computing Machinery, 2014.

Glen Pink and James R. Curran. SYDNEY at TAC 2014. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Glen Pink, Joel Nothman, and James R. Curran. Analysing recall loss in named entity slot filling. In *Conference on Empirical Methods in Natural Language Processing*, pages 820–830. Association for Computational Linguistics, 2014.

Lutz Prechelt. Early stopping — but when? In *Neural Networks: Tricks of the Trade*, pages 553–553. Springer, 1998.

Pengda Qin, Chaoyi Ma, Yidong Jia, Wei Wang, Zhengkuan Zhang, Zuyi Bao, Weiran Xu, and Jun Guo. BUPT_PRIS at TAC KBP 2015. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Xin Ying Qiu, Xiaoting Li, Weijian Mo, Manli Zheng, and Zhuhe Zheng. GDUFS at slot filling TAC-KBP 2012. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–286. Institute of Electrical and Electronics Engineers, 1989.

Rashedur Rahman, Brigitte Grau, Sophie Rosset, Yoann Dupont, Jérémy Guillemot, Olivier Mesnard, Christian Lautier, and Wilson Fred. TAC KBP 2016 cold start slot filling and slot filler validation systems by IRT SystemX. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016.

Jonathan Raiman and John Miller. Globally normalized reader. In *Conference on Empirical Methods in Natural Language Processing*, pages 1070–1080. Association for Computational Linguistics, 2017.

Arpana Rawal, Ani Thomas, M K Kowar, and Sanjay Sharma. ARPANI@BIT_DURG: KBP English slot-filling task challenge. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Marek Rei and Ted Briscoe. Combining manual rules and supervised learning for hedge cue and scope detection. In *Computational Natural Language Learning*, pages 56–63. Association for Computational Linguistics, 2010.

Uwe Reichel and Piroska Lendvai. Veracity computing from lexical cues and perceived certainty trends. In *Workshop on Noisy User-generated Text*, pages 33–42. International Committee on Computational Linguistics, 2016.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, Tarek F. Abdelzaher, and Jiawei Han. CoType: Joint extraction of typed entities and relations with knowledge bases. In *International Conference on World Wide Web*, pages 1015–1024. International World Wide Web Conference Committee, 2017.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. A Markov logic approach to bio-molecular event extraction. In *Workshop on Current Trends in Biomedical Natural Language Processing*, pages 41–49. Association for Computational Linguistics, 2009.

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

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84. Association for Computational Linguistics, 2013.

Bryan Rink and Sanda Harabagiu. UTD: Classifying semantic relations by combining lexical and semantic resources. In *International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics, 2010.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. Modeling missing data in distant supervision for information extraction. In *Transactions of the Association for Computational Linguistics*, volume 1, pages 367–378. Association for Computational Linguistics, 2013.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129. Association for Computational Linguistics, 2015.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. In *International Conference on Learning Representations*, 2016.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *IEEE Workshops on Application of Computer Vision*, pages 29–36. Institute of Electrical and Electronics Engineers Computer Society, 2005.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. In *Psychological Review*, volume 65(6), pages 386–408. American Psychological Association, 1958.

Benjamin Roth and Dietrich Klakow. Combining generative and discriminative model scores for distant supervision. In *Conference on Empirical Methods in Natural Language Processing*, pages 24–29. Association for Computational Linguistics, 2013.

Benjamin Roth, Grzegorz Chrupała, Michael Wiegand, Mittul Singh, and Dietrich Klakow. Generalizing from freebase and patterns using cluster-based distant supervision for KBP slot-filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. Effective slot filling based on shallow distant supervision methods. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Benjamin Roth, Tassilo Barth, Grzegorz Chrupała, Martin Gropp, and Dietrich Klakow. RelationFactory: A fast, modular and effective system for knowledge base population. In *Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations*, pages 89–92. Association for Computational Linguistics, 2014a.

Benjamin Roth, Emma Strubell, John Sullivan, Lakshmi Vikraman, Kate Silverstein, and Andrew McCallum. Universal schema for slot-filling, cold-start KBP and event argument extraction: UMass IESL at TAC KBP 2014. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014b.

Benjamin Roth, Nicholas Monath, David Belanger, Emma Strubell, Patrick Verga, and Andrew McCallum. Building knowledge bases with universal schema: Cold start and slot-filling approaches. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. In *Computational Natural Language Learning*, pages 1–8. Association for Computational Linguistics, 2004.

Dan Roth and Wen-tau Yih. Global inference for entity and relation identification via a linear programming formulation. In *Introduction to Statistical Relational Learning*, pages 553–580. MIT Press, 2007.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. In *Nature*, volume 323(6088), pages 533–536. Nature Publishing Group, 1986.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics, 2015.

Evan Sandhaus. The New York Times annotated corpus. *Linguistic Data Consortium*, 6(12), 2008.

Sunita Sarawagi and William W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, pages 1185–1192. Neural Information Processing Systems Foundation, Inc., Curran Associates, Inc., 2005.

Roser Sauri. *A factuality profiler for eventualities in text*. PhD thesis, Brandeis University, 2008.

Roser Saurí and James Pustejovsky. FactBank: A corpus annotated with event factuality. In *International Conference on Language Resources and Evaluation*, volume 43(3), pages 227–268. European Language Resources Association, 2009.

Roser Saurí and James Pustejovsky. Are you sure that this happened? Assessing the factuality degree of events in text. In *Computational Linguistics*, volume 38(2), pages 261–299. MIT Press, 2012.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI Conference on Artificial Intelligence*, pages 3776–3784. Association for the Advancement of Artificial Intelligence, 2016.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. Joint inference of entities, relations, and coreference. In *Workshop on Automated Knowledge Base Construction*, pages 1–6. Association for Computing Machinery, 2013a.

Sameer Singh, Limin Yao, David Belanger, Ari Kobren, Sam Anzaroot, Michael Wick, Alexandre Passos, Harshal Pandya, Jinho Choi, Brian Martin, and Andrew McCallum. Universal schema for slot filling and cold start: UMass IESL at TACKBP 2013. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013b.

Amit Singhal. Introducing the knowledge graph: Things, not strings. *Official Google Blog*, 2012.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

Stephen Soderland, John Gilmer, Robert Bart, Oren Etzioni, and Daniel S. Weld. Open information extraction to KBP relations in 3 hours. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Stephen Soderland, Natalie Hawkins, John Gilmer, and Daniel S. Weld. Combining open IE and distant supervision for KBP slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. In *Information Processing & Management*, volume 45(4), pages 427–437. Elsevier, 2009.

Pavel Sountsov and Sunita Sarawagi. Length bias in encoder decoder models and a case for global conditioning. In *Conference on Empirical Methods in Natural Language Processing*, pages 1516–1525. Association for Computational Linguistics, 2016.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, volume 15(1), pages 1929–1958. Microtome Publishing, 2014.

Sanja Štajner, Nicole Baerg, Simone Paolo Ponzetto, and Heiner Stuckenschmidt. Automatic detection of speculation in policy statements. In *Workshops on Natural Language Processing and Computational Social Science*. Association for Computing Machinery, 2016.

Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. Ghent University - IBCN participation in the TAC KBP 2015 cold start slot filling task. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Marijn F. Stollenga, Jonathan Masci, Faustino Gomez, and Jürgen Schmidhuber. Deep networks with internal selective attention through feedback connections. In *Advances in Neural Information Processing Systems*, pages 3545–3553. Neural Information Processing Systems Foundation, Inc., Curran Associates, Inc., 2014.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge. In *International Conference on World Wide Web*, pages 697–706. International World Wide Web Conference Committee, 2007.

Kodai Sudo, Akihiko Kato, Van-Thuy Phi, Hiroyuki Shindo, and Yuji Matsumoto. NAIST participation in the TAC KBP 2016 cold start slot filling task: Combining CNN-based and bootstrapping-based methods. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016.

Mihai Surdeanu. Overview of the TAC2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013.

Mihai Surdeanu and Heng Ji. Overview of the English slot filling track at the TAC2014 knowledge base population evaluation. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Joint Conference on Empirical*

Methods in Natural Language Processing and Computational Natural Language Learning, pages 455–465. Association for Computational Linguistics, 2012.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112. Neural Information Processing Systems Foundation, Inc., Curran Associates, Inc., 2014.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*, pages 93–128. MIT Press, 2007.

György Szarvas, Veronika Vincze, Richárd Farkas, György Móra, and Iryna Gurevych. Cross-genre and cross-domain detection of semantic uncertainty. In *Computational Linguistics*, volume 38(2), pages 335–367. MIT Press, 2012.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Annual Meeting of the Association for Computational Linguistics*, pages 721–729. Association for Computational Linguistics, 2012.

Buzhou Tang, Xiaolong Wang, Xuan Wang, Bo Yuan, and Shixi Fan. A cascade method for detecting hedges and their scope in natural language text. In *Computational Natural Language Learning*, pages 13–17. Association for Computational Linguistics, 2010.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *Computing Research Repository (arXiv.org), arXiv:1605.02688*, 2016.

Özlem Uzuner, Xiaoran Zhang, and Tawanda Sibanda. Machine learning and rule-based approaches to assertion classification. In *Journal of the American Medical Informatics Association*, volume 16(1), pages 109–115. BMJ Group BMA House, 2009.

Sofie Van Landeghem, Yvan Saeys, Bernard De Baets, and Yves Van de Peer. Analyzing text in search of bio-molecular events: A high-precision machine learning framework. In *BioNLP Workshop*, pages 128–136. Association for Computational Linguistics, 2009.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer, 1995.

Erik Velldal. Detecting uncertainty in biomedical literature: A simple disambiguation approach using sparse random indexing. In *Symposium on Semantic Mining in Biomedicine*, pages 72–80. CEUR-WS.org, 2010.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. Speculation and negation: Rules, rankers, and the role of syntax. In *Computational Linguistics*, volume 38(2), pages 369–410. MIT Press, 2012.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. Multilingual relation extraction using compositional universal schema. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896. Association for Computational Linguistics, 2016.

Veronika Vincze. Uncertainty detection in Hungarian texts. In *International Conference on Computational Linguistics*, pages 1844–1853. International Committee on Computational Linguistics, 2014a.

Veronika Vincze. *Uncertainty detection in natural language texts*. PhD thesis, University of Szeged, 2014b.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. In *BMC Bioinformatics*, volume 9(11), pages 1–9. BioMed Central Ltd., 2008.

Dileep Viswanathan, Anurag Wazalwar, Ameet Soni, Jude Shavlik, and Sriraam Natarajan. TAC KBP 2015: English slot filling track — relational learning with expert advice. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015.

Andreas Vlachos and Stephen Clark. Application-driven relation extraction with limited distant supervision. In *AHA! Workshop on Information Discovery in Text*, pages 1–6. International Committee on Computational Linguistics, 2014.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. Combining recurrent and convolutional neural networks for relation classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539. Association for Computational Linguistics, 2016.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. In *IEEE Transactions on Acoustics, Speech, and Signal Processing*, volume 37(3), pages 328–339. Institute of Electrical and Electronics Engineers, 1989.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. ACE 2005 multilingual training corpus. *Linguistic Data Consortium*, 57, 2006.

Chang Wang, James Fan, Aditya Kalyanpur, and David Gondek. Relation extraction with relation topics. In *Conference on Empirical Methods in Natural Language Processing*, pages 1426–1436. Association for Computational Linguistics, 2011.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. Relation classification via multi-level attention CNNs. In *Annual Meeting of the Association for Computational Linguistics*, pages 1298–1307. Association for Computational Linguistics, 2016a.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing*, pages 616–626. Association for Computational Linguistics, 2016b.

Anurag Wazalwar, Tushar Khot, Ce Zhang, Chris Ré, Jude Shavlik, and Sriraam Natarajan. TAC KBP 2014 : English slot filling track DeepDive with expert advice. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Zhongyu Wei, Junwen Chen, Wei Gao, Binyang Li, Lanjun Zhou, Yulan He, and Kam-Fai Wong. An empirical study on uncertainty identification in social media context. In *Annual Meeting of the Association for Computational Linguistics*, pages 58–62. Association for Computational Linguistics, 2013.

Paul J. Werbos. Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78(10), pages 1550–1560. Institute of Electrical and Electronics Engineers, 1990.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *International Conference on World Wide Web*, pages 515–526. International World Wide Web Conference Committee, 2014.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics, 2005.

Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In *Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306. Association for Computational Linguistics, 2016.

Yi Wu, David Bamman, and Stuart Russell. Adversarial training for relation extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 1779–1784. Association for Computational Linguistics, 2017.

Yang Xiang, Xiaoqiang Zhou, Qingcai Chen, Zhihui Zheng, Buzhou Tang, Xiaolong Wang, and Yang Qin. Incorporating label dependency for answer quality tagging in community question answering via CNN-LSTM-CRF. In *International Conference on Computational Linguistics*, pages 1231–1241. International Committee on Computational Linguistics, 2016.

Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 842–850. Institute of Electrical and Electronics Engineers, 2015.

Jian Xu, Qin Lu, Jie Liu, and Ruifeng Xu. NLPComp in TAC 2012 entity linking and slot-filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2012.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Conference on Empirical Methods in Natural Language Processing*, pages 536–540. Association for Computational Linguistics, 2015a.

Puyang Xu and Ruhi Sarikaya. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 78–83. Institute of Electrical and Electronics Engineers, 2013.

Sheng Xu, Chunxia Zhang, Zhendong Niu, Rongyue Mei, Junpeng Chen, Junjiang Zhang, and Hongping Fu. BIT's slot-filling method for TAC-KBP 2013. In *Text Analysis Conference*. National Institute of Standards and Technology, 2013a.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. Filling knowledge base gaps for distant supervision of relation extraction. In *Annual Meeting of the Association for Computational Linguistics*, pages 665–670. Association for Computational Linguistics, 2013b.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying relations via long short term memory networks along shortest dependency paths. In *Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794. Association for Computational Linguistics, 2015b.

Yadollah Yaghoobzadeh and Hinrich Schütze. Corpus-level fine-grained entity typing using contextual information. In *Conference on Empirical Methods in Natural Language Processing*, pages 715–725. Association for Computational Linguistics, 2015.

Yadollah Yaghoobzadeh and Hinrich Schütze. Multi-level representations for fine-grained typing of knowledge base entities. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 578–589. Association for Computational Linguistics, 2017.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. Noise mitigation for neural entity typing and relation extraction. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 1183–1194. Association for Computational Linguistics, 2017.

Bishan Yang and Claire Cardie. Joint inference for fine-grained opinion extraction. In *Annual Meeting of the Association for Computational Linguistics*, pages 1640–1649. Association for Computational Linguistics, 2013.

Bishan Yang, Ndapandula Nakashole, Bryan Kisiel, Emmanouil A. Platanios, Abulhair Saparov, Shashank Srivastava, Derry Wijaya, and Tom Mitchell. CMUML micro-reader system for KBP 2016 cold start slot filling, event nugget detection, and event argument linking. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016a.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. A position encoding convolutional neural network based on dependency tree for relation classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 65–74. Association for Computational Linguistics, 2016b.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. Association for Computational Linguistics, 2016c.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong(Shiao-Long) Li, and Feng Gao. Recurrent conditional random field for language understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4077 – 4081. Institute of Electrical and Electronics Engineers, 2014.

Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023. Association for Computational Linguistics, 2010.

Limin Yao, Sebastian Riedel, and Andrew McCallum. Probabilistic databases of universal schema. In *Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 116–121. Association for Computational Linguistics, 2012.

Hai Ye, Wenhan Chao, Zhunchen Luo, and Zhoujun Li. Jointly extracting relations with class ties via effective deep ranking. In *Annual Meeting of the Association for Computational Linguistics*, pages 1810–1820. Association for Computational Linguistics, 2017.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. In *Transactions of the Association for Computational Linguistics*, volume 4, pages 259–272. Association for Computational Linguistics, 2016.

Dian Yu and Heng Ji. Unsupervised person slot filling based on graph mining. In *Annual Meeting of the Association for Computational Linguistics*, pages 44–53. Association for Computational Linguistics, 2016.

Dian Yu, Xiaoman Pan, Boliang Zhang, Lifu Huang, Di Lu, Spencer Whitehead, and Heng Ji. RPI_BLENDER TAC-KBP2016 system description. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016.

Mo Yu, Matthew R. Gormley, and Mark Dredze. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1374–1379. Association for Computational Linguistics, 2015.

Xiaofeng Yu and Wai Lam. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *International Conference on Computational Linguistics (Posters)*, pages 1399–1407. International Committee on Computational Linguistics, 2010.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *International Conference on Computational Linguistics*, pages 2335–2344. International Committee on Computational Linguistics, 2014.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762. Association for Computational Linguistics, 2015.

Chunyun Zhang, Weiran Xu, Pengda Qin, Aiqin Qi, Yuling Liu, Xingya Wang, and Qunluo. The PRIS systems at slot filling and entity linking 2014. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014.

Dongxu Zhang and Dong Wang. Relation classification via recurrent neural network. *Computing Research Repository (arXiv.org), arXiv:1508.01006*, 2015.

Jinjian Zhang, Huachun Ma, Siliang Tang, and Fei Wu. Combining MIML and distant supervision for KBP slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015a.

Jinjian Zhang, Siliang Tang, and Fei Wu. Sentences embedding for slot filling via convolutional neural networks. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016a.

Meishan Zhang, Yue Zhang, and Guohong Fu. Transition-based neural word segmentation. In *Annual Meeting of the Association for Computational Linguistics*, pages 421–431. Association for Computational Linguistics, 2016b.

Meishan Zhang, Yue Zhang, and Guohong Fu. End-to-end neural relation extraction with global optimization. In *Conference on Empirical Methods in Natural Language Processing*, pages 1731–1741. Association for Computational Linguistics, 2017a.

Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. Semi-supervised structured prediction with neural CRF autoencoder. In *Conference on Empirical Methods in Natural Language Processing*, pages 1702–1712. Association for Computational Linguistics, 2017b.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. Towards accurate distant supervision for relational facts extraction. In *Annual Meeting of the Association for Computational Linguistics*, pages 810–815. Association for Computational Linguistics, 2013.

Yangcheng Zhang, Hengsheng Liu, Gang Zhao, and Ji Wu. MSIIPL_THU's slot-filling method for TAC-KBP 2015. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015b.

Yuhao Zhang, Arun Chaganty, Ashwin Paranjape, Danqi Chen, Jason Bolton, Peng Qi, and Christopher D. Manning. Stanford at TAC KBP 2016: Sealing pipeline leaks and understanding chinese. In *Text Analysis Conference*. National Institute of Standards and Technology, 2016c.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Conference on Empirical Methods in Natural Language Processing*, pages 35–45. Association for Computational Linguistics, 2017c.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. Joint extraction of entities and relations based on a novel tagging scheme. In *Annual Meeting of the Association for Computational Linguistics*, pages 1227–1236. Association for Computational Linguistics, 2017.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Annual Meeting of the Association for Computational Linguistics*, pages 427–434. Association for Computational Linguistics, 2005.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. Tree kernel–based relation extraction with context-sensitive structured parse tree information. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 728–736. Association for Computational Linguistics, 2007.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Annual Meeting of the Association for Computational Linguistics*, pages 1213–1222. Association for Computational Linguistics, 2015a.

Huiwei Zhou, Huan Yang, Long Chen, Zhenwei Liu, Jianjun Ma, and Degen Huang. Combining feature-based and instance-based transfer learning approaches for cross-domain

hedge detection with multiple sources. In *Social Media Processing*, volume 568, pages 225–232. Springer, 2015b.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Annual Meeting of the Association for Computational Linguistics*, pages 207–212. Association for Computational Linguistics, 2016.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2D conditional random fields for web information extraction. In *International Conference on Machine Learning*, pages 1044–1051. International Machine Learning Society, 2005.

List of Publications Incorporated in this Thesis

Peer-reviewed International Conferences

- Heike Adel and Hinrich Schütze. Global normalization of convolutional neural networks for joint entity and relation classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 1724–1730. Association for Computational Linguistics, 2017b

- Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. Noise mitigation for neural entity typing and relation extraction. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 1183–1194. Association for Computational Linguistics, 2017

- Heike Adel and Hinrich Schütze. Exploring different dimensions of attention for uncertainty detection. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 22–34. Association for Computational Linguistics, 2017c

- Heike Adel, Benjamin Roth, and Hinrich Schütze. Comparing convolutional neural networks to traditional models for slot filling. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 828–838. Association for Computational Linguistics, 2016

International Workshop for Shared Task

- Heike Adel and Hinrich Schütze. CIS at TAC cold start 2015: Neural networks and coreference resolution for slot filling. In *Text Analysis Conference*. National Institute of Standards and Technology, 2015

- Heike Adel and Hinrich Schütze. TAC KBP 2014 slot filling shared task: Baseline system for investigating coreference. In *Text Analysis Conference*. National Institute of Standards and Technology, 2014

Pre-prints

- Heike Adel and Hinrich Schütze. Impact of coreference resolution on slot filling. *Computing Research Repository (arXiv.org), arXiv:1710.09753*, 2017a