Distributed Representations for Fine-grained Entity Typing

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

Yadollah Yaghoobzadeh

München 2017

Erstgutachter: Prof. Dr. Hinrich Schütze
Zweitgutachter: Prof. Dr. Chris Biemann
Drittgutachter: Lecturer Andreas Vlachos, PhD

Tag der Einreichung: 10. August 2017
Tag der mündlichen Prüfung: 29. September 2017

29. September 2017

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 29.09.2017

Yadollah Yaghoobzadeh

Abstract

Knowledge about entities is essential for natural language understanding. This knowledge includes several facts about entities such as their names, properties, relations and types. This data is usually stored in large scale structures called knowledge bases (KB) and therefore building and maintaining KBs is very important. Examples of such KBs are Wikipedia, Freebase and Google knowledge graph.

Incompleteness is unfortunately a reality for every KB, because the world is changing – new entities are emerging, and existing entities are getting new properties. Therefore, we always need to update KBs. To do so, we propose an information extraction method that processes large raw corpora in order to gather knowledge about entities. We focus on extraction of entity types and address the task of *fine-grained entity typing*: given a KB and a large corpus of text with mentions of entities in the KB, find all fine-grained types of the entities. For example given a large corpus and the entity "Barack Obama" we need to find all his types including PERSON, POLITICIAN, and AUTHOR.

Artificial neural networks (NNs) have shown promising results in different machine learning problems. Distributed representation (embedding) is an effective way of representing data for NNs. In this work, we introduce two models for fine-grained entity typing using NNs with distributed representations of language units: (i) A global model that predicts types of an entity based on its global representation learned from the entity's name and contexts. (ii) A context model that predicts types of an entity based on its context-level predictions.

Each of the two proposed models has some specific properties. For the global model, learning high quality entity representations is crucial because it is the only source used for the predictions. Therefore, we introduce representations using name and contexts of entities on three levels of entity, word, and character. We show each has complementary information and a multi-level representation is the best. For the context model, we need to use distant supervision since the context-level labels are not available for entities. Distant supervised labels are noisy and this harms the performance of models. Therefore, we introduce and apply new algorithms for noise mitigation using multi-instance learning.

Since the performance of both models is highly dependent on the quality of distributed representations of words and entities, we aim to find out which models are learning better representations. The common evaluations are mostly based on the human judgements about the overall similarity of embeddings. We show that these kinds of evaluation are problematic and a better evaluation is needed. We introduce new evaluation methods to investigate the features of embedding models better. This helps us to find out which embedding models we should use for our task.

The contributions we make in this work include the following: (i) We address fine-grained entity typing by using text corpora with the application in knowledge base completion. (ii) We build a dataset for this task from Freebase entities and their fine-grained types. (iii) We propose and implement two novel models for the task and show that each model has special features. (iv) We represent entities using novel distributed representations on three levels of entity, word and character. (v) We introduce new algorithms for multi-instance learning in neural networks and apply them for the first time to the task of fine-grained entity typing. (vi) We present a novel evaluation method for distributed representation of words.

Zusammenfassung

Wissen über Entitäten ist essentiell für das Verständnis natürlicher Sprachen (NLU). Dieses Wissen umfasst verschiedene Fakten über Entities, einschließlich ihrer Namen, Eigenschaften, Relationen und Typen. Da diese Informationen üblicherweise in großangelegten Strukturen, genannt *knowledge bases* (KB), gespeichert werden, ist das Erstellen und Pflegen der KBs sehr wichtig. Beispiele für KBs sind Wikipedia, Freebase oder Google knowledge graph.

Leider ist in der Realität jede KB unvollständig, da die Welt sich ständig verändert; neue Entitäten entstehen, und existierende Entitäten erhalten neue Eigenschaften. Daher müssen KBs ständig aktualisiert werden. Um dies zu erreichen, stellen wir eine Methode zur Informationsextraktion aus großen unannotierten Korpora vor, um Wissen über Entitäten zu sammeln. Wir konzentrieren uns auf die Extraktion von Entitätstypen und befassen uns mit dem Task *fine-grained entity typing*: dem Finden aller feinkörniger Typen von Entitäten mit einer gegebenen KB und einem großen Korpus, in dem die Entitäten der KB erwähnt werden. Ein Beispiel wäre, mit Hilfe eines großen Korpus alle Typen der gegebenen Entität "Barack Obama" zu finden, einschließlich PERSON, POLITIKER und AUTOR.

Künstliche neuronale Netze (NN) haben vielversprechende Ergebnisse für verschiedene Probleme des maschinellen Lernens erzielt. Verteilte Repräsentationen (Embeddings), sind ein effektiver Weg, um Daten neuronalen Netzen zugänglich zu machen. In dieser Arbeit stellen wir zwei Modelle für *fine-grained entity typing* mit neuronalen Netzen und verteilten Repräsentationen sprachlicher Einheiten vor: (i) ein globales Modell, das den Typ einer Entität anhand ihrer globalen Repräsentation, die aus ihrem Namen und ihrem Kontext gelernt wird, vorhersagt; (ii) ein Kontextmodell, das den Typ einer Entität basierend auf Vorhersagen auf dem Kontextlevel vorhersagt.

Jedes dieser Modelle hat spezielle Features. Für das globale Modell ist es entscheidend, Entitätsrepräsentationen von hoher Qualität zu lernen, da sie die einzigen Informationen sind, die für die Vorhersagen verwendet werden. Daher führen wir Repräsentationen ein, die den Namen und Kontext von Entitäten auf Entitäts-, Wort- und Buchstabenlevel verwenden. Wir zeigen, dass diese Repräsentationen komplementäre Informationen beinhalten und dass Multilevel-

repräsentationen die besten Ergebnisse erzielen. Für das Kontextmodell müssen wir distant supervision verwenden, da wir keine Kontextlevellabels für Entitäten zur Verfügung haben. Distant supervision Labels sind verrauscht, was die Performanz der Modelle reduziert. Daher stellen wir neue Algorithmen zur Verminderung von Rauschen mit Hilfe von Lernen mit multiplen Instanzen vor, und wenden diese an.

Da die Performanz beider Modelle stark von der Qualität der verteilten Repräsentationen der Wörter und Entitäten abhängt, wollen wir herausfinden, welche Modelle bessere Repräsentationen lernen. Übliche Evaluierungsmethoden basieren hauptsächlich auf menschlicher Beurteilung der allgemeinen Ähnlichkeit von Embeddings. Wir zeigen, dass diese Arten der Evaluierung problematisch sind und dass eine bessere Evaluierung benötigt wird. Wir stellen neue Evaluierungsmethoden vor, um die Features der Embeddingmodelle besser zu verstehen. Dies hilft uns, herauszufinden, welche Embeddingmodelle wir für unseren Task verwenden sollten.

Diese Arbeit beinhaltet die folgenden Beiträge: (i) Wir beschäftigen uns mit *fine-grained entity typing* unter Zuhilfename von Textkorpora für die Anwendung in der Vervollständigung von KBs. (ii) Wir erstellen ein Datenset für diesen Task unter Verwendung von Freebase-Entitäten und ihrer *fine-grained* Typen. (iii) Wir präsentieren und implementieren zwei neue Modelle für den Task und zeigen, dass jedes spezielle Features hat. (iv) Wir respräsentieren Entitäten mit neuen verteilten Respräsentationen auf Entitäts-, Wort- und Buchstabenlevel. (v) Wir führen neue Algorithmen für neuronale Netze zum Lernen mit multiplen Instanzen ein und sind die Ersten, die sie auf den Task des *fine-grained entity typing* anwenden. (vi) Wir stellen eine neue Evaluierungsmethode für verteilte Repräsentationen von Wörtern vor.

Acknowledgments

I would like to thank everyone who directly or indirectly helped me to finish this dissertation.

First, I appreciate all the support and help from Prof. Hinrich Schütze. I was very lucky to have him as my advisor. He tought me how to do research, focusing on interesting problems and solving them step by step.

I could not imagine finishing my thesis without countless discussions and support from my colleagues at CIS. Among all I would like to mention Wenpeng Yin, Heike Adel, David Kaummans, Sascha Rothe, Ehsan Asgari, Irina Sergienya, , Katharina Kann, Sebastian Ebert, Thang Vu, and Thomas Müller.

It is impossible to accomplish anything without the support from friends and family. I especially thank some of them, who I spent most of my time outside of academic life, koosha khajehmoogahi, Keyvan Kardel, Milad Khanibeig, Katharina Kann, Heike Adel, Sascha Rothe, Ehsan Asgari, Ramin Izadpanah, Hesam Moradi and Morteza Moosavi. And finally, I appreciate all the countless help I got from my family. My parents encouraged and supported me by any means they could and I dedicate my dissertation to them.

Contents

Publications and Declaration of Co-Authorship

Chapter 2

Chapter 2 corresponds to the following publication:

> Yadollah Yaghoobzadeh, Hinrich Schütze; **Corpus-level Fine-grained Entity Typing Using Contextual Information**; Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (Lisbon, Portugal, September, 2015), pages 715–725.

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.

Chapter 3

Chapter 3 corresponds to the following publication:

> Yadollah Yaghoobzadeh, Hinrich Schütze; **Intrinsic Subspace Evaluation of Word Embedding Representations**; Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Berlin, Germany, August, 2016) , pages 236–246.

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.

Chapter 4

Chapter 4 corresponds to the following publication:

Yadollah Yaghoobzadeh, Hinrich Schütze; **Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities**; Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Valencia, Spain, April, 2017), pages 578–589.

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.

Chapter 5

Chapter 5 corresponds to the following publication:

Yadollah Yaghoobzadeh, Heike Adel, Hinrich Schütze; **Noise Mitigation for Neural Entity Typing and Relation Extraction**; Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Valencia, Spain, April, 2017), pages 1183–1194.

This work is the result of a collaboration. Heike Adel and I contributed in equal parts. Heike Adel contributed those parts that are concerned with relation extraction, "the relation extraction part". I contributed those parts that are concerned with entity typing, "the entity typing part". I regularly discussed the entity typing part with my coauthors. Apart from these explicitly declared exceptions, I conceived of the original research contributions of the entity typing part and performed implementation and evaluation of the entity typing part. I wrote the initial draft of the entity typing part and did most of the subsequent corrections. My coauthors assisted me in improving the entity typing part.

München, 31.05.2017

Yadollah Yaghoobzadeh

Chapter 1

Introduction

Natural language understanding (NLU) is not possible without prior knowledge about the world. Many natural language processing (NLP) tasks, which must be addressed as part of NLU, need world knowledge; e.g., many coreference ambiguities can only be resolved based on world knowledge. Also, most NLU applications combine a variety of information sources that include both text sources and knowledge bases (KBs); e.g., question answering systems need access to knowledge bases like gazetteers. Thus, high-quality KBs, as resources to keep and query world knowledge, are critical for successful NLU.

Unfortunately, most large scale KBs like Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and Google knowledge graph are incomplete. The effort required to create KBs is considerable and since the world changes, it will always continue. KBs are therefore always in need of updates and corrections. Their structure is roughly equivalent to a graph in which entities are nodes and edges are relations between entities. Each node is also associated with one or more semantic classes, called types. Most prior work tries to complete the edges between entities, but here in this work, the focus is on completion of entity types in KBs.

The approach we adopt in this work to address incompleteness of KBs is extraction of information from large text corpora. Text can be argued to be the main source of the knowledge represented in KBs. Thus, it is reasonable to attempt completing them based on text. There is in fact a significant body of work on corpus-based methods for extracting knowledge from text; however, most of it has addressed relation extraction. Our focus instead is acquisition of types.

More specifically, we address the problem of *fine-grained entity typing*, i.e., inferring from a large corpus that an entity is a member of a fine-grained class such as FOOD or ARTIST. We propose two approaches (i) a global model that predicts types based on the aggregated representation of entities and (ii) a context model that first scores the individual contexts of an entity and then aggregates the

scores to make the type predictions.

Neural networks (NNs) are very successful to model machine learning problems in NLP. These models mostly work on distributed representations of language units and high dimensional real-valued vectors. We use distributed representations and NNs to learn functions that model entities, their contexts and names.

In Section 1.1, we give background about distributed representation, followed by Section 1.2 which focuses on NLP and how distributed representations and NNs are used there. In Section 1.3, the task of fine-grained entity typing is defined and motivated. Finally, in Section 1.4, we introduce our models for the task of fine-grained entity typing.

1.1 Distributed Representations

In the domain of machine learning, data representation is one of the essential elements to get good performance (Bengio et al., 2013). Feature engineering has been widely the standard way of designing data representations based on human prior knowledge of the task. In this way, a human designs some features for each task, applies them on the data to build the data representation, and then trains a function from input to output representations.

Feature engineering has some drawbacks, e.g., it is time consuming, suboptimal and domain specific. Another alternative is to learn representations automatically. In this way, the learner should identify and disentangle the important factors of the data to get good performance on the objective function. Representation learning is mostly addressed using neural networks (NNs) and it is the focus of this work.

Generally, there are two ways of representation in NNs. (i) *Local representation*: each concept is represented with one computing unit (i.e., neuron), and vice versa, each unit is representing one concept. (ii) *Distributed representation*: each concept is represented by a pattern of activity over many computing units, and vice versa, each computing unit is involved in representing many concepts (Hinton, 1984).

Local representations are easy to understand and implement, but they do not support generalization; the model has to remember all the units and their concepts. For an unseen concept, local representations are unable to do any inference. Distributed representations are not easy to understand and implement, but they are generalizable by nature. Activation patterns, i.e., the representations, have a notion of similarity, meaning that similar concepts have similar representations. This will enable distributed representations to be generalizable and for an unseen concept, we can still infer something based on its similar activation patterns. In this

work, we use distributed representations.

1.2 Natural Language Processing and Distributed Representations

Natural language processing (NLP) is an important area in artificial intelligence, dealing with understanding and generating human language. Machine learning (ML) methods are dominant to solve different NLP tasks such as part of speech tagging, named entity recognition, sentiment analysis, machine translation. Therefore, representation learning for NLP is important. Distributed representations have been widely used in NLP, especially after the recent rise of neural networks (NNs). One good example is the applications of word embeddings, i.e., distributed representations of words. Word embeddings are discussed more in Section 1.2.1.

Input units in NLP applications can be defined on different levels: characters, character n-grams, words, phrases, sentences, paragraphs, documents, books, etc. Distributed representations of all these language units can be achieved using NNs, usually by representing higher level units as a function of lower level ones. In this section, we describe how NNs and distributed representations are used in NLP with more focus on the classification problems.

1.2.1 Neural Networks

Neural networks (NNs) are powerful learning models that are inspired by biology of brains. NNs consist of a large number of neurons, i.e., computational nodes. They usually consist of several layers including input and output layers and one or several hidden layers. Each layer consists of multiple units (neurons). The input layer is responsible to receive the input signals. Hidden layers are transforming data through non-linear functions to compute more abstract representations of the input. Output layer is transforming hidden layers to the desired output format, which in classification is usually the set of label scores.

Recent advances in training NNs made them very popular, and they are now state-of-the-art in many NLP problems (Goldberg, 2016). In the following, we discuss how NN layers are usually defined for NLP tasks. Specifically, we cover input layer, hidden layer (feed-forward, convolutional-maxpooling and recurrent) and output layer.

Input Layer

In NNs, the input layer represents the data in a format suitable for learning. In NLP, words are usually considered as basic units of the data (i.e., language). Other

unit types are also widely used, e.g., characters (dos Santos and Zadrozny, 2014; Zhang et al., 2015; Kim et al., 2016), morphemes (Botha and Blunsom, 2014), character n-grams (Bojanowski et al., 2016) and even sentences (Kiros et al., 2015). The first step is thus choosing the right unit type for the application, and how to segment data into a sequence of those units. Next, we need to represent the units properly.

To get better generalization, we feed vector representations of the units to the NNs. To do so, we first map each unit to a d dimensional vector – the distributed representations. These mappings are done for each unit in the vocabulary and are stored in a matrix called *lookup table*. This matrix is initialized randomly or by pre-trained embeddings. The word embeddings, discussed in Section 1.2.1, trained for language modeling objectives, are very common types of pre-trained embeddings.

Then, onsidering an input sequence of $x_1 x_2 \dots x_l$ of language elements of x_i, we represent each with a d dimensional vector $\mathbf{x}_i \in \mathbb{R}^d$. Then the representation of the input sequence will be a matrix $\mathbf{X} \in \mathbb{R}^{d \times l}$ where l is the length of the input sequence:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \dots & \mathbf{x}_{1,d} \\ \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \dots & \mathbf{x}_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{l,1} & \mathbf{x}_{l,2} & \dots & \mathbf{x}_{l,d} \end{bmatrix}^T \tag{1.1}$$

Column i of $\mathbf{X}$ represents the vector representation of ith input unit x_i.

Hidden Layer

Here, we briefly introduce the three most typical architectures for the hidden layers: fully connected feed-forward, convolutional-maxpooling, and recurrent. Each one has some properties and for a particular task might be a better fit. It is also common to use a combination of different architectures, e.g., Kim et al. (2016) and Xiao and Cho (2016) use combinations of convolutional-maxpooling and recurrent architectures.

Fully connected feed-forward architecture. The fully connected feed-forward architecture is the most simple architecture to transform input to hidden layers or to transform a hidden layer to an upper level one. In this architecture, each neuron is connected to all the neurons in the upper layer. An example architecture is shown in Figure 1.1. Each layer in this architecture does this transformation:

$$\mathbf{h}_i = f(\mathbf{W}_i \mathbf{h}_{i-1}) \tag{1.2}$$

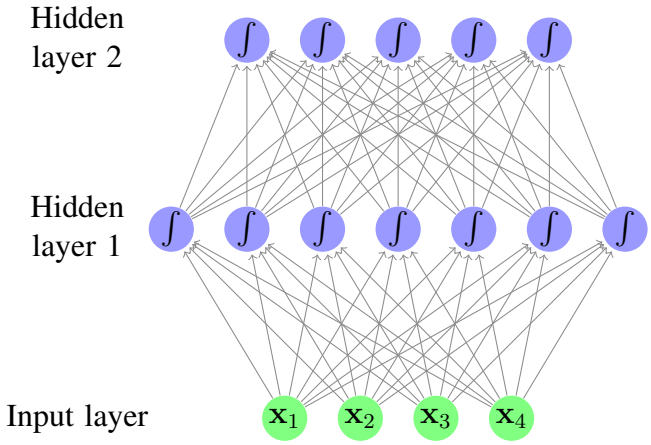

Figure 1.1 – *Fully connected feed-forward architecture. In this example, two fully connected hidden layers are applied on the input layer. The input units are represented by vectors* $\mathbf{x}_i$.

which basically is a linear transformation of previous layer $\mathbf{h}_{i-1}$ using matrix $\mathbf{W}_i$ and then applying the non-linearity f. $\tanh$ and ReLu (rectified linear unit) are typical choices for f. $\mathbf{h}_0 \in \mathbb{R}^{d \cdot l}$ is the vector representation of the input matrix, i.e, concatenation of all l vectors $x_i \in \mathbb{R}^d$.

Convolutional-maxpooling architecture. The convolutional-maxpooling architecture (Lecun and Bengio, 1995) is useful for classification tasks in which strong local clues exist about a class membership. These clues can appear in different places in the input. In other words, convolution-maxpooling can find some certain n-grams of units useful for a particular task, independent of their position in the input. They show promising results on many tasks, including document classification (Johnson and Zhang, 2015), short-text categorization (Wang et al., 2015), sentiment classification (Kalchbrenner et al., 2014; Kim, 2014), relation type classification between entities (Zeng et al., 2014; dos Santos et al., 2015), paraphrase identification (Yin and Schütze, 2015).

Convolutional architecture uses k filters of different window widths w (typically $w \in [1, .., 8]$) to narrowly convolve $\mathbf{X}$. For each filter $\mathbf{M} \in \mathbb{R}^{d \times w}$, the result of the convolution of $\mathbf{M}$ over matrix $\mathbf{X}$ is feature map $\mathbf{m} \in \mathbb{R}^{l-w+1}$:

$$\mathbf{m}[i] = f(\mathbf{X}_{[:,i:i+w-1]} \odot \mathbf{M} + b)$$

where f is the activation (e.g., $\tanh$ or ReLu) function, b is the bias, $\mathbf{X}_{[:,i:i+w-1]}$ are the columns i to $i+w-1$ of $\mathbf{X}$ and $\odot$ is the sum of element-wise multiplication

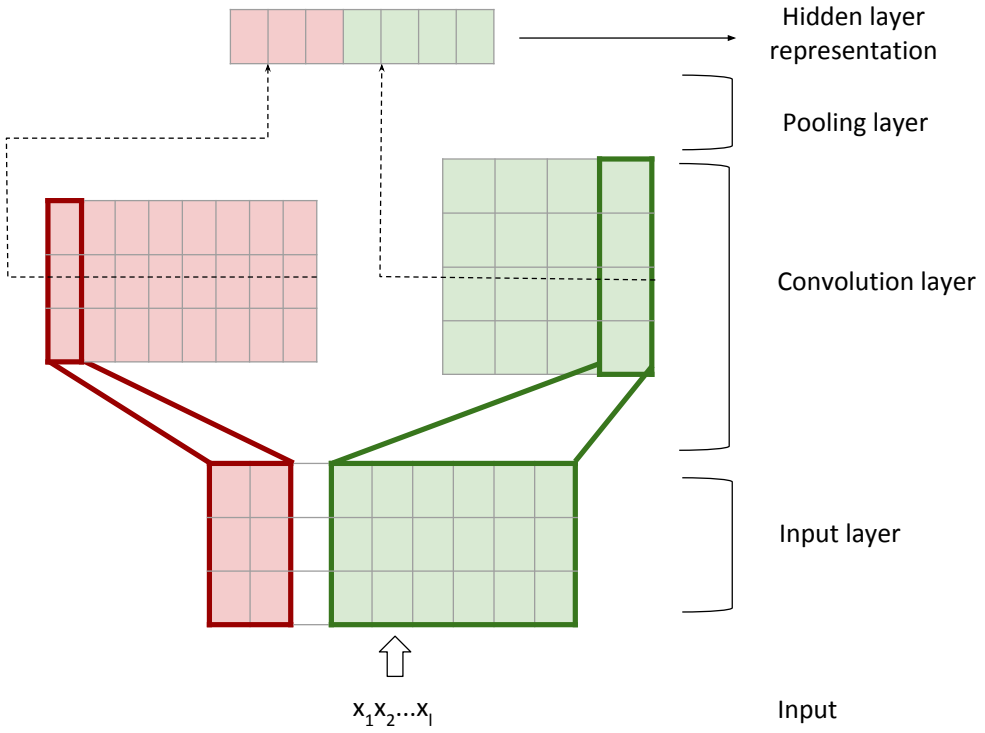

Figure 1.2 – *Convolutional-maxpooling architecture. In this example, there are three convolution filters of widths two and four filters of width six. Max-pooling outputs are considered as the hidden layer representation.*

(Frobenius inner product). Finally, we take maxpooling (maximum over time) to get the feature corresponding to the filter $\mathbf{M}$:

$$u = \max_i \mathbf{m}[i] \tag{1.3}$$

The concatenation of all k of these features is our representation: $\mathbf{h} \in \mathbb{R}^k$. An example convolutional-maxpooling architecture is show in Figure 1.2.

Recurrent architecture. In natural language we often work with sequences of arbitrary size. Fully connected architecture is sequence aware, but it is hard to train because it is too sensitive to the order and an insertion/deletion of a unit will cause a big change in the input space. Also, it does not support arbitrary size of input. Convolutional-maxpooling architectures could handle the arbitrary size of input, but it is not suitable for modeling the whole sequence; each filter can model just a local sequence of input units. Recurrent architecture (Elman, 1990) is designed to model NLP problems with arbitrary size sequences of input units. To do

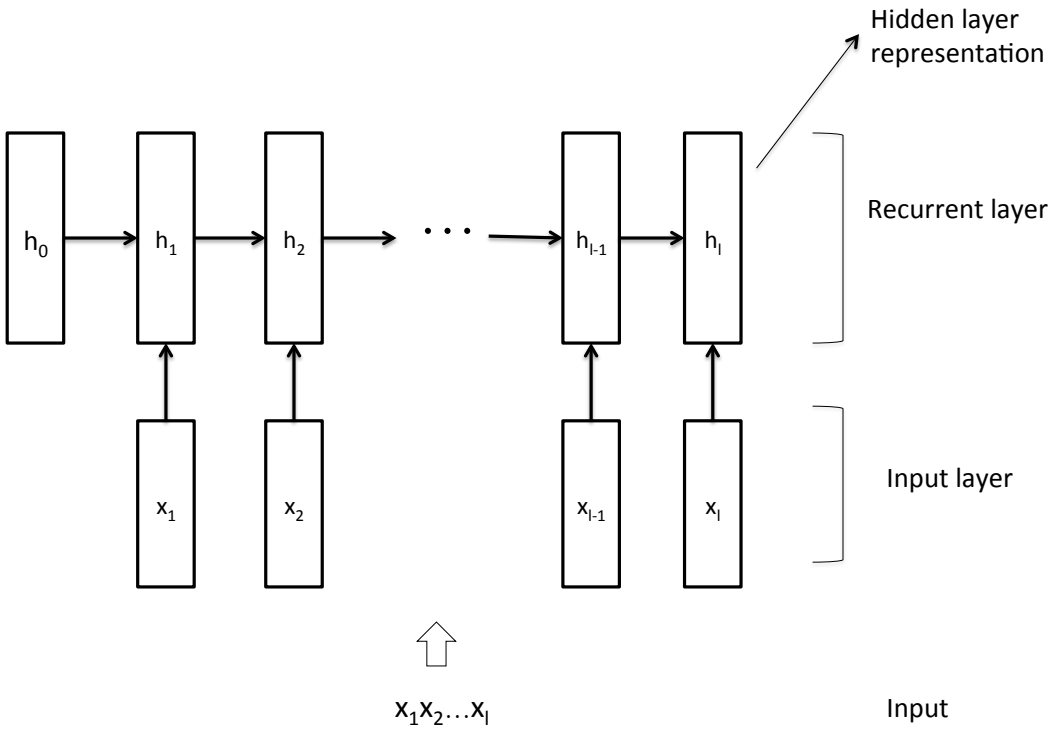

Figure 1.3 – *Recurrent architecture. h_t is the hidden state at time t and is updated by the current input x_t and previous state h_{t-1}.*

so, we can learn a fixed length vector by processing the input sequentially, i.e., applying a composition function and updating a memory (or general representation) at each time step in the input sequence. The current input x_t at time t together with the previous hidden state h_{t-1} generate a new hidden state at time t as:

$$\mathbf{h}_t = f(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1}) \tag{1.4}$$

where $\mathbf{W}_x \in \mathbb{R}^{|h| \times d}$ and $\mathbf{W}_h \in \mathbb{R}^{|h| \times |h|}$ are the transformation matrices. At $t = 1$, $\mathbf{h}_0$ is initialized to zero.

Vanishing gradient problem prevents standard recurrent model to work for long sequences. Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is designed to tackle this problem. It does so by introducing different

gates that control updating the memories in the recurrent states. It models the unit sequence x as follows (**?**):

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_x^i \mathbf{x}_t + \mathbf{W}_h^i \mathbf{h}_{t-1} + \mathbf{W}_c^i \mathbf{c}_{t-1} + \mathbf{b}^i) && (1.5)\\
\mathbf{f}_t &= \sigma(\mathbf{W}_x^f \mathbf{x}_t + \mathbf{W}_h^f \mathbf{h}_{t-1} + \mathbf{W}_c^f \mathbf{c}_{t-1} + \mathbf{b}^f) && (1.6)\\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{W}_x^c \mathbf{x}_t + \mathbf{W}_h^c \mathbf{h}_{t-1} + \mathbf{b}^c) && (1.7)\\
\mathbf{o}_t &= \sigma(\mathbf{W}_x^o \mathbf{x}_t + \mathbf{W}_h^o \mathbf{h}_{t-1} + \mathbf{W}_c^o \mathbf{c}_t + \mathbf{b}^o) && (1.8)\\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t) && (1.9)
\end{aligned}$$

where $\circ$ is Hadamard product and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input, forget and output gates. Parameters of the LSTM are $\mathbf{W}_x^j$, $\mathbf{W}_h^j$, $\mathbf{b}^j$ for $j \in \{i, f, c, o\}$ and $\mathbf{W}_c^j$ for $j \in \{i, f, o\}$. At $t = 1$, $\mathbf{h}_0$ and $\mathbf{c}_0$ are initialized to zero. The last hidden LSTM state $\mathbf{h}_l$ is usually considered as the whole input $\mathbf{X}$ representation.

The bi-directional LSTM consists of two separate LSTMs that are applied on the input sequence, one going forward and one going backward. The bi-directional LSTM representation is usually the concatenation of last states of the forward and backward LSTMs.

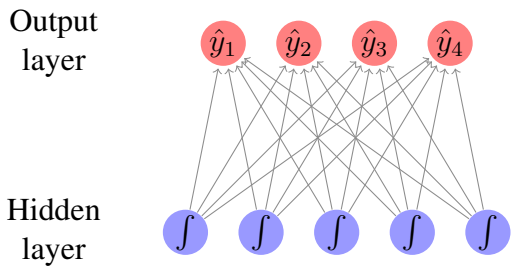

Figure 1.4 – *Example output layer with four units.*

Output Layer

In neural networks, output layer is responsible to generate output variables for specific inputs based on the states in the hidden layer. For a classification task, usually each unit in the output layer represents the score or probability of a class. Therefore, there is usually a fully connected layer with the size equal to the number of classes. Example output layer is shown in Figure 1.4. This layer is usually connected to the last hidden layer (in some cases, e.g., in bidirectional recurrent architecture, some hidden layers are concatenated before feeding to the output layer). We define the output layer $\hat{\mathbf{y}} \in \mathbb{R}^{|y|}$ as:

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_{\text{out}} \mathbf{h}) \qquad (1.10)$$

where $\mathbf{h} \in \mathbb{R}^{|h|}$ is the hidden layer, $\mathbf{W}_{\text{out}} \in \mathbb{R}^{|y| \times |h|}$ is the weight matrix from hidden layer to the output layer of size $|y|$. σ is the sigmoid function: $\sigma(x) = 1/(1+e^{-x})$ that converts the value x to a value in $[0, 1]$.[1]

Training

To train neural networks, we need to compare the predictions with the gold outputs. It is common to convert the gold outputs to a binary vector $\mathbf{y} \in \{0, 1\}^{|y|}$, in which each index corresponds to a specific label and it has the value of 1 if the example has that label.

Cross entropy is a common loss function for classification problems and for each training example is defined as:

$$\sum_{j}^{|y|} -\Big(\mathbf{y}_j \log \hat{\mathbf{y}}_j + (1 - \mathbf{y}_j) \log (1 - \hat{\mathbf{y}}_j)\Big) \tag{1.11}$$

where $\mathbf{y}_j$ and $\hat{\mathbf{y}}_j$ are truth and prediction for jth output, respectively.

Stochastic gradient descent (SGD) (Bottou, 2012; LeCun et al., 1998) is usually used for updating the parameters to minimize the loss function. Some more advanced variants of standard SGD, such as AdaGrad (Duchi et al., 2011), Adam (Kingma and Ba, 2014), and Momentum (Polyak, 1964), are also introduced to find more optimal parameters or to converge faster.

Distributed Representation of Words

Distributed word representations or word embeddings are currently an active area of research in NLP. The motivation for embeddings is that knowledge about words is helpful in NLP. Representing words as vocabulary indexes, i.e, local representation, may be a good approach if large training sets allow us to learn everything we need to know about a word to solve a particular task; but in most cases it helps to have a representation that contains some information about the word and allows inferences like: "above" and "below" have similar syntactic behavior or "engine" and "motor" have similar meaning.

Language model based objectives are considered generic and since they do not need labeled data, they have been widely used to compute word embeddings. Several neural network architectures with different properties have been proposed to learn word embeddings. In these settings, a network is trained to predict a word in a context and the weights in the first layer of the network are considered as word embeddings.

[1] The jth unit in the output, $\hat{y}_j$, is the probability of the jth label to be one. Multiple labels can have the value of one.

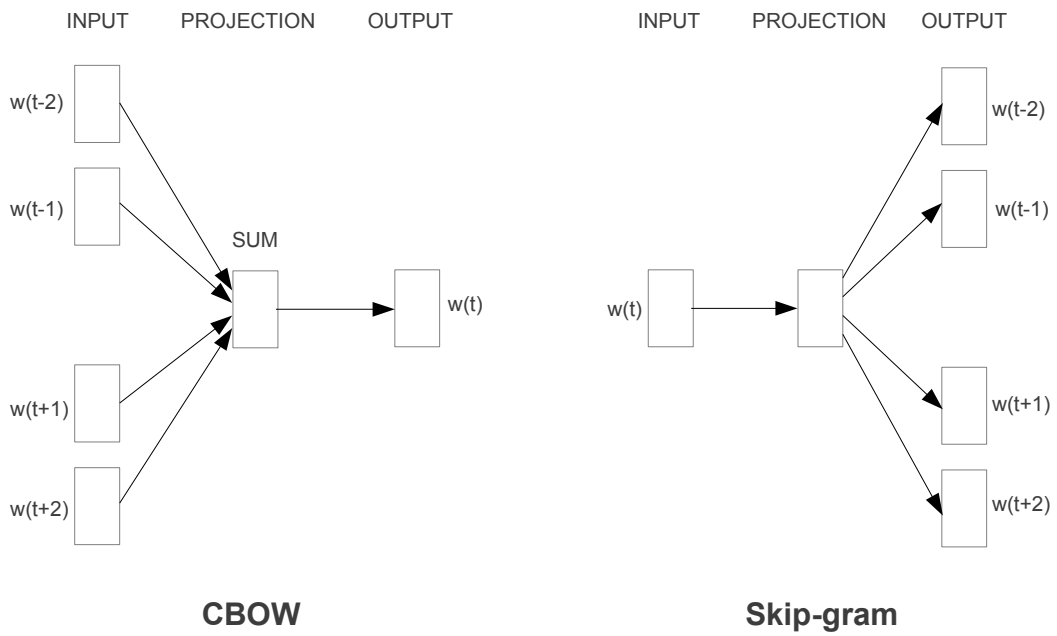

Figure 1.5 – *Two architectures for learning word embedding proposed by Mikolov et al. (2013). CBOW predicts the current word based on the context, and the Skip-gram predicts context words given the current word.*

Mikolov et al. (2013) introduce Skip-gram (skipgram bag-of-word model), (iii) CBOW (continuous bag-of-word model) models. For a given context, represented by the input space representations of the left and right neighbors w_{t-2}, w_{t-1} and w_{t+1}, w_{t+2}, CBOW predicts w_t by adding the context vectors. Skip-gram predicts the context words w_{t-2}, w_{t-1}, w_{t+1} and w_{t+2} given the input word w_t. The architectures are shown in Figure 1.5. Both CBOW and Skip-gram are learning embeddings using bag-of-word (BoW) models. There are other architectures where order of words in the sentence is also implemented, cf., (Mnih and Kavukcuoglu, 2013; Ling et al., 2015a).

Evaluation. Two types of evaluation, intrinsic and extrinsic, have been applied to assess which models are more suitable for learning word embeddings.

Intrinsic evaluations assess the quality of embeddings independent of an NLP task. Currently, this evaluation mostly is done by testing overall distance/similarity of words in the embedding space, i.e., it is based on viewing word representations as points and then computing *full-space similarity*. Similarity and analogy datasets are widely used as intrinsic evaluation. In similarity, datasets compare

the human judgments of word similarities with the embedding similarities. Similarity of two embeddings is computed mostly using cosine function defined as: $\text{cosine}(\mathbf{v}, \mathbf{w}) = (\mathbf{v}^T\mathbf{w})/(||\mathbf{v}||_2 \times ||\mathbf{w}||_2)$. In analogy, the similarity of words along a property is evaluated using analogical questions like "Berlin" to "Germany" is "Paris" to X.

Extrinsic evaluations test embeddings for a specific NLP task (cf. (Li and Jurafsky, 2015; Köhn, 2015; Lai et al., 2015)). Extrinsic evaluation is a valid methodology, but it does not allow us to understand the properties of representations without further analysis; e.g., if an evaluation shows that embedding A works better than embedding B on a task, then that is not an analysis of the causes of the improvement.

Each single word is a combination of a large number of morphological, lexical, syntactic, semantic, discourse and other features. Its embedding should accurately and consistently represent these features, and ideally a good evaluation method must clarify this and give a way to analyze the results. The goal of Chapter 3 is to build such an evaluation. We introduce some criteria for word embeddings and build grammars to generate artificial text based on them and evaluate embeddings on those criteria. We also show some issues with the intrinsic evaluations based on similarity and analogy.

1.3 Fine-grained Entity Typing

Large scale knowledge bases (KBs) like Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and Google knowledge graph are designed to store world knowledge. Their structure is usually graph-based and with different schemas. In Figure 1.6, we show some parts of an example KB (Buitelaar, 2007) that contains some geographical entities and their properties and relations. In this graph, the upper parts are the schema of the KB including the entity types (e.g., RIVER and GEOGRAPHICAL ENTITY) and the relations between them (e.g., the relation "flow_through" from type RIVER to type GEOGRAPHICAL ENTITY). The lower parts are the instances of types ("Neckar" is an instance of type RIVER and GERMANY is a COUNTRY), relations between them (relation "flow_through" from "Neckar" to "Germany"), and their property-values (e.g., "length(m)" of 367 for "Neckar").

Here in this work, we use Freebase. Freebase is a labeled graph, with nodes and directed edges. Topics (or entities) are the essential part of Freebase, which are represented as graph nodes. These topics can be named entities (like "Germany") or abstract concepts (like "love"). In this work, we refer to Freebase topics as entities. Apart from entities, Freebase uses types like CITY, COUNTRY, BOOK_SUBJECT, PERSON, etc. Each entity can have one or many types, e.g.,

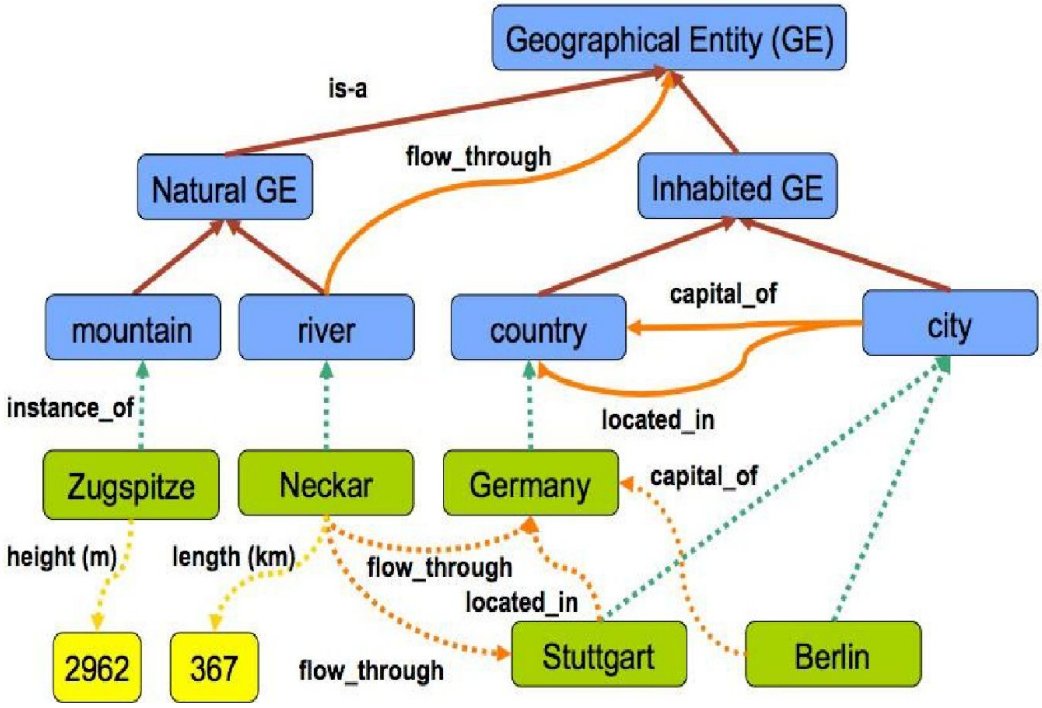

Figure 1.6 – *Part of a knowledge base with types, entities, relations and properties.*

"Arnold Schwarzenegger" is a PERSON, ACTOR, POLITICIAN, SPORT_FIGURE, etc. There are about 1,500 types in Freebase, organized by domains; e.g., the domain FOOD has types like FOOD, INGREDIENT and RESTAURANT. Each type contains some specific properties about entities, e.g., ACTOR type contains a property that lists all films that "Arnold Schwarzenegger" has acted in. In other words, entities are connected to each other by properties because they are in certain types. For example, "Arnold Schwarzenegger" is connected to "California" with property "Governor_of" which is defined in the type POLITICIAN.

Even though Freebase is one of the largest publicly available KB of its kind, it still has significant coverage problems; e.g., 78.5% of persons in Freebase do not have *nationality* (Min et al., 2013). In our Freebase dump, 22% of entities have only one type. This is unavoidable, partly because Freebase is user-generated, partly because the world changes. All existing KBs that attempt to model a large part of the world suffer from this incompleteness problem. Incompleteness is likely to become an even bigger problem in the future as the number of types covered by KBs like Freebase increases. As more and more fine-grained types are added, achieving good coverage for these new types using only human editors will become impossible.

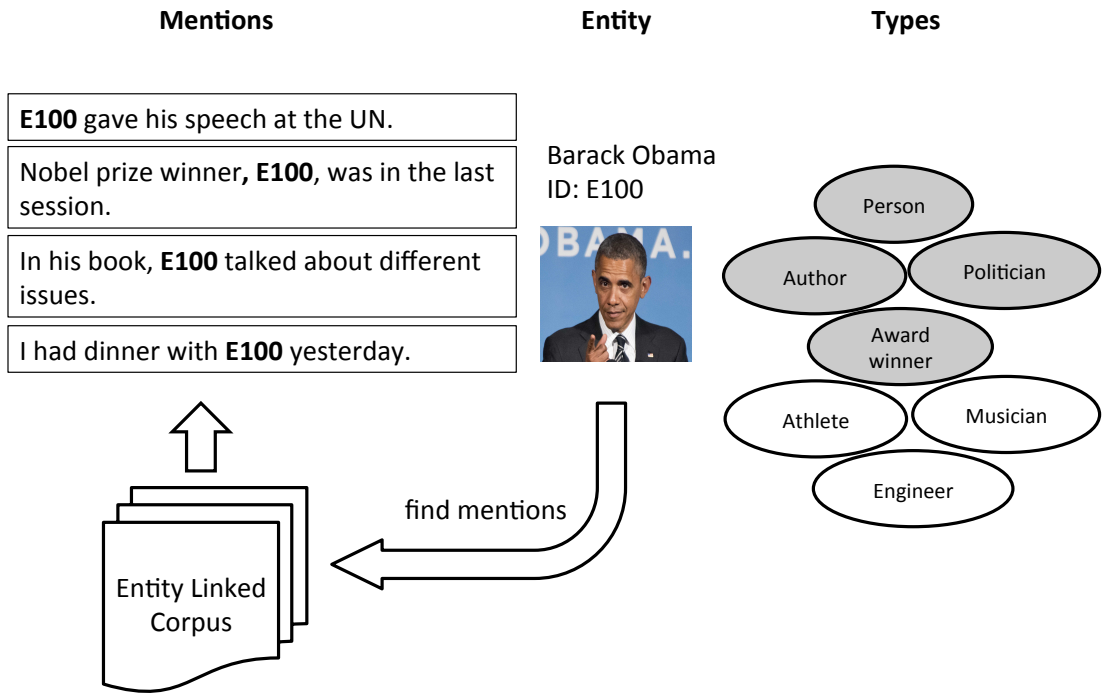

Figure 1.7 – *Fine-grained entity typing: task definition.*

The approach we adopt in this paper is to address incompleteness of Freebase as an example KB. We aim to do that by extracting information from large text corpora. We focus on completing types of entities. More specifically, in our problem setting we assume that the following is given: a KB with a set of entities E with their names, a set of types T and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t; and a large linked corpus C in which mentions of E are annotated to the KB. In this problem setting, we address the task of *fine-grained entity typing*: we want to infer from the corpus for each pair of entity e and type t whether $m(e, t) = 1$ holds, i.e., whether entity e is a member of type t.

An example is shown in Figure 1.7. "Barack Obama" is taken from a KB with ID "E100". We then look for the mentions of "E100" in the entity linked corpus. In the figure, there are four examples of the mentions. Then, the task is to separate the types that E100 belongs to from other types. In this case, we have to find PERSON, POLITICIAN, AUTHOR and AWARD_WINNER.

1.3.1 Related Work

Our task is to infer fine-grained types of KB entities. Neelakantan and Chang (2015) and Xie et al. (2016) also address a similar task, but they rely on entity descriptions in KBs. Thus, in contrast to our approach, their system is not able to type entities that are not covered by existing KBs. We infer classes for entities from a large corpus and do not assume that these entities occur in the KB. The problem of Fine-grained mention typing (FGMT) (Yosef et al., 2012; Ling and Weld, 2012; Yogatama et al., 2015; Del Corro et al., 2015; Shimaoka et al., 2016; Ren et al., 2016b) is related to our task. FGMT classifies single *mentions* of named entities to their context dependent types whereas we attempt to identify all types of a KB *entity* from the aggregation of all its mentions. FGMT can still be evaluated in our task by aggregating the mention level decisions.

Entity set expansion (ESE) is the problem of finding entities in a class (e.g., medications) given a seed set (e.g., {"Ibuprofen", "Maalox", "Prozac"}). The standard solution is pattern-based bootstrapping (Thelen and Riloff, 2002; Gupta and Manning, 2014). ESE is different from the problem we address because ESE starts with a small seed set whereas we assume that a large number of examples from a knowledge base (KB) is available. Initial experiments with the system of Gupta and Manning (2014) showed that it was not performing well for our task – this is not surprising given that it is designed for a task with properties quite different from entity typing.

Fine-grained entity typing can be used for *knowledge base completion (KBC)*. Most KBC systems focus on *relations* between entities, not on *types* as we do. Some generalize the patterns of relationships within the KB (Nickel et al., 2012; Bordes et al., 2013) while others use a combination of within-KB generalization and information extraction from text (Weston et al., 2013; Socher et al., 2013; Jiang et al., 2012; Riedel et al., 2013; Wang et al., 2014).

The first step in extracting information about entities from text is to reliably identify mentions of these entities. This problem of *entity linking* has some mutual dependencies with entity typing. Indeed, some recent work shows large improvements when entity typing and linking are jointly modeled (Ling et al., 2015c; Durrett and Klein, 2014). However, there are constraints that are important for high-performance entity linking, but that are of little relevance to entity typing. For example, there is a large literature on entity linking that deals with coreference resolution and inter-entity constraints – e.g., "Naples" is more likely to refer to a US (resp. an Italian) city in a context mentioning "Fort Myers" (resp. "Sicily"). Therefore, we will only address entity typing in this work and consider entity linking as an independent module that provides contexts of entities for the system. A similar process is used in relation extraction (cf. (Zeng et al., 2015; Lin et al., 2016)).

1.3.2 Freebase and FIGER types

In about 1500 types of Freebase, some are very general like LOCATION, some are very fine-grained, e.g., VIETNAMESE_URBAN_DISTRICT. There are types that have a large number of instances like CITYTOWN and types that have very few like CAMERA_SENSOR. The types are not organized in a strict taxonomy even though there exists an *included type* relationship between types in Freebase. The reason is that for a user-generated KB it is difficult to maintain taxonomic consistency. For example, almost all instances of AUTHOR are also instances of PERSON, but sometimes organizations author and publish documents.

Our goal is fine-grained typing of entities, but types like VIETNAMESE_URBAN_DISTRICT are too fine-grained. To create a reliable setup for evaluation and to make sure that all types have a reasonable number of instances, we adopt the FIGER type set (Ling and Weld, 2012) that was created with the same goals in mind. FIGER consists of 113 tags and was created in an attempt to preserve the diversity of Freebase types while consolidating infrequent and unusual types through filtering and merging. For example, the Freebase types DISH, INGREDIENT, FOOD and CHEESE are mapped to one type FOOD. See (Ling and Weld, 2012) for a complete list of FIGER types.

1.4 Models for Fine-grained Entity Typing

In Section 1.3, we introduced the task of fine-grained entity typing. In summary, the setting is that the following is given: a KB with a set of entities E, a set of fine-grained types T and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t; and a large annotated corpus C in which mentions of E are linked.

Our general approach is that we use a set of training examples to learn $P(t|e)$: the probability that entity e has type t. These probabilities can be used to assign *new types* to entities covered in the KB as well as typing *unknown entities* – i.e., entities not covered by the KB. To work for new or unknown entities, we would need an entity linking system such as the ones participating in TAC KBP (McNamee and Dang, 2009) that identifies and clusters mentions of them.

We use two general types of modeling for this problem: global model, context model and a joint model of the two. In the following, we introduce each model and the distributed representations we used to implement them.

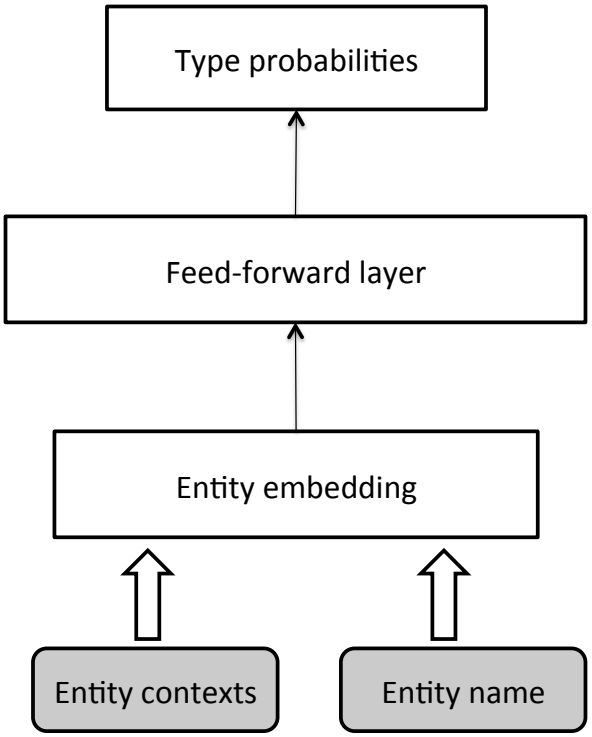

Figure 1.8 – *Global model. Entity embedding is learned from contexts and name of the entity.*

1.4.1 Global Model

The global model (GM) scores possible types of entity e based on a *distributed representation* or *embedding* $\mathbf{e} \in \mathbb{R}^d$ of e. $\mathbf{e}$ can be learned from corpus or entity name. Accordingly, we define representations of entities on three levels: (i) entity (ii) word (iii) character.

After learning this vector representation, we learn $P(t|e)$ by using a fully connected feed-forward hidden layer and an output layer of size $|T|$, i.e., number of types. We model $P(t|e)$ as a multi-label classification. In the following, we present our entity, word and character level models to learn entity embeddings.

Entity-level representation of entities. We learn distributed representations for entities so that entities with similar meanings will have similar representations. Thus, we can learn a d dimensional embedding $\mathbf{e}$ of entity e, in the same space as word embeddings, by running an embedding learner like `word2vec` on a corpus in which all mentions of the entity have been replaced by a special identifier. Similar method is used by (Wang et al., 2014; Wang and Li, 2016; Yamada et al.,

2016; Fang et al., 2016) to learn entity embeddings by replacing Wikipedia anchors with their referred article ID. We refer to these entity vectors as the *entity level representation* (ELR). (More details in Chapter 2 and Chapter 4)

Word-level representation of entities. Words inside entity names are important sources of information for typing entities. We define the word-level representation (WLR) as the *average of the embeddings of the words* that the entity name contains $\mathbf{e} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{w}_i$ where $\mathbf{w}_i$ is the embedding of the ith word of an entity name of length n. We opt for simple averaging since entity names often consist of a small number of words with clear semantics.

Character-level representation of entities. For computing the *character level representation* (CLR), we design models that try to type an entity based on the sequence of characters of its name. Our hypothesis is that names of entities of a specific type often have similar character patterns. Entities of type ETHNICITY often end in "ish" and "ian", e.g., "Spanish" and "Russian". Entities of type MEDICINE often end in "en": "Lipofen", "Acetaminophen". Also, some types tend to have specific cross-word shapes in their entities, e.g., PERSON names usually consist of two words, or MUSIC names are usually long, containing several words.

We compute character-level representations of entities by segmenting the entity names into their sequence of characters, and then represent them using distributed representations and applying either fully connected feed-forward, convolutional-maxpooling or recurrent hidden layers. The parameters of these models are trained to predict the types. See Chapter 4 for more details.

1.4.2 Context Model

For the context model (CM), we first learn a probability function $P(t|c)$ for individual contexts c in the corpus. $P(t|c)$ is the probability that an entity occurring in context c has type t. For example, consider the contexts c_1 = "he served SLOT cooked in wine" and c_2 = "she loves SLOT more than anything". SLOT marks the occurrence of an entity and it also shows that we do not care about the entity mention itself but only its context. For the type t = "food", $P(t|c_1)$ is high whereas $P(t|c_2)$ is low. This example demonstrates that some contexts of an entity like "beef" allow specific inferences about its type whereas others do not. Based on the context probability function $P(t|c)$, we then compute the entity-level CM probability function $P(t|e)$.

More specifically, consider $B = \{c_1, c_2, \ldots, c_q\}$ as the set of q contexts of e in the corpus. Each c_i is an instance of e and since e can have several labels, it is a multi-instance multi-label (MIML) learning problem. We address MIML using neural networks by representing each context as a vector $\mathbf{c}_i \in \mathbb{R}^h$, and learn $P(t|e)$ from the set of contexts of entity e. We represent the contexts by two types

of hidden layers: fully connected feed-forward and convolutional-maxpooling. In both cases, the input layer is the matrix of context word embeddings. The architecture also includes a hidden layer of either fully connected feed-forward or convolutional-maxpooling architecture (details are in Chapter 2 and Chapter 5). In the following, we describe our MIML algorithms that work on the contexts representations to compute $P(t|e)$.

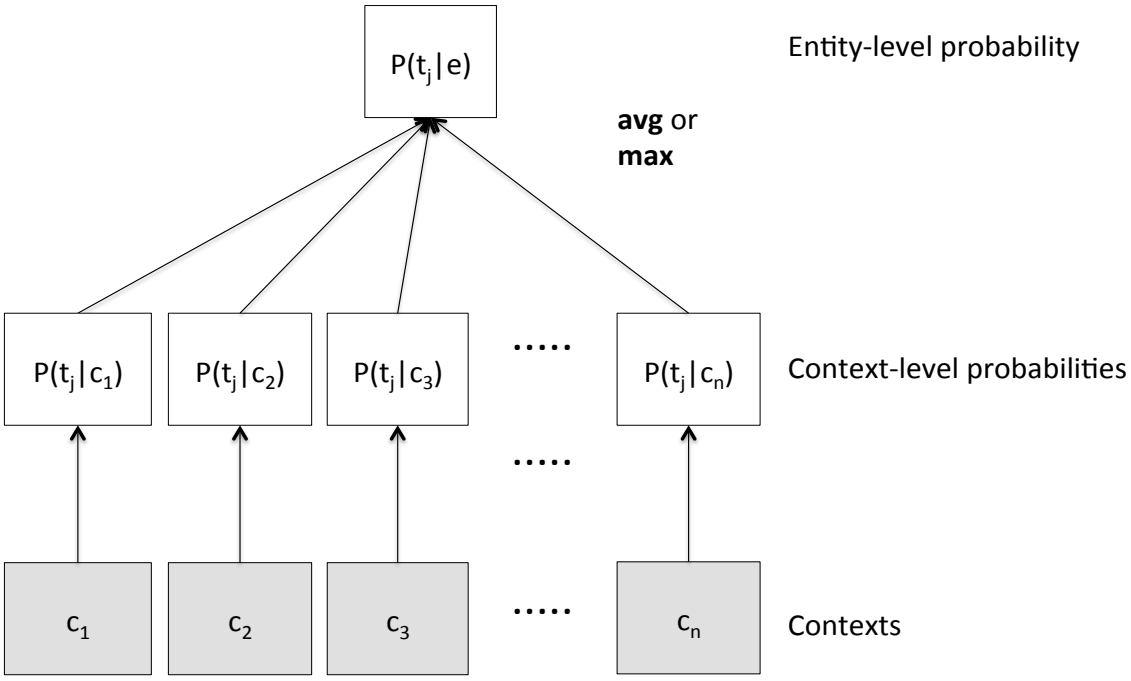

Figure 1.9 – *Learning the entity-level (bag-level) probability for type t_j from the context-level (instance-level) probabilities using average or max as aggregation functions. In distant supervision, we apply the aggregation function (AVG or MAX) only during test time. In MIML-AVG, MIML-MAX, we apply the AVG and MAX during train and test time.*

The **distant supervision** assumption is that *all* contexts of an entity with type t are contexts of t; e.g., we label all contexts mentioning "Barack Obama" with all of his types, including PERSON, POLITICIAN, AUTHOR and AWARD_WINNER. Therefore, we can learn $P(t|c_i)$ for each context c_i of e, and aggregate them using a function like "average" or "maximum" to get $P(t|e)$. See Figure 1.9.

Obviously when distant supervision is used, the labels are incorrect or *noisy* for some contexts. Multi-instance multi-label (MIML) learning addresses this problem and has been applied before in similar task of relation extraction (Surdeanu et al., 2012). We apply MIML to fine-grained Entity Typing. Our assump-

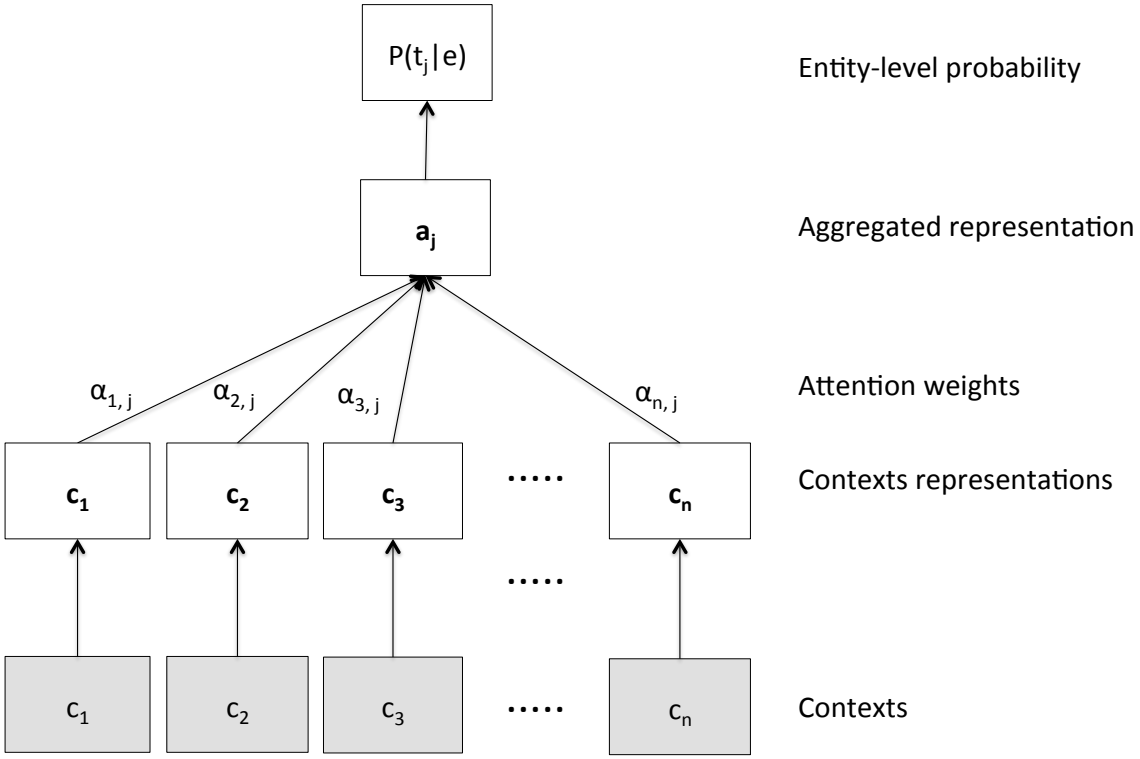

Figure 1.10 – *Multi-instance multi-label learning using attention (MIML-ATT). The entity-level (bag-level) probability of type t_j is computed based on an aggregated representation $\mathbf{a}_j$ of all the contexts.*

tion is: if entity e has type t, then there is at least one context of e in the corpus in which e occurs as type t. So, we apply this assumption during training with the following estimation of the type probability of an entity. which means we take the **maximum** probability of type t over all the contexts of entity e as $P(t|e)$. We call this approach **MIML-MAX**. See Figure 1.9.

MIML-MAX picks the most confident context for t, ignoring the probabilities of all the other contexts. Apart from missing information, this can be especially harmful if the entity annotations in the corpus are the result of an entity linking system. In that case, the most confident context might be wrongly linked to the entity. So, it can be beneficial to leverage all the contexts into the final prediction, e.g., by **averaging** the type probabilities of all the contexts of entity e. We call this approach **MIML-AVG**.

MIML-AVG treats every context equally which might be problematic since many contexts are irrelevant for a particular type. A better way is to weight the

contexts according to their similarity to the types. Therefore, we propose using selective **attention** over contexts as follows and call this approach **MIML-ATT**. MIML-ATT is the multi-label version of the selective attention method proposed in Lin et al. (2016). $\mathbf{a}_t$ is the type t specific aggregated representation of all the contexts c_i of the entity e and $\alpha_{i,t}$ is the attention score (i.e., weight) of context c_i for type t and $\mathbf{a}_t \in \mathbb{R}^h$ can be interpreted as the representation of entity e for type t. See Chapter 5 for more details on MIML methods.

1.5 Summary and Overview

In this chapter, we gave short introductions to several concepts that we used in this work. We also defined our task and models. The next four chapters are our published papers on this topic. In Chapter 2, we introduce the task of fine-grained entity typing and introduce the first version of the two mentioned models for solving the task. In Chapter 3, we investigate a new intrinsic evaluation method for distributed representation of words, and also show that this task is a good fit for extrinsic evaluation. This evaluation highlights some important performance differences between different learning models and gives us hints for further improvements of the entity typing models. In Chapter 4, we improve the global model by introducing multi-level representations of entities in three levels of characters, words and entities. In Chapter 5, our aim is to improve the context model by tackling its key problem, i.e, noisy labels because of distant supervision. We introduce and apply new algorithms that effectively mitigate the noise and increase the performance.

Chapter 2

Corpus-level Fine-grained Entity Typing Using Contextual Information

Corpus-level Fine-grained Entity Typing Using Contextual Information

Yadollah Yaghoobzadeh and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
yadollah@cis.lmu.de

Abstract

This paper addresses the problem of corpus-level entity typing, i.e., inferring from a large corpus that an entity is a member of a class such as "food" or "artist". The application of entity typing we are interested in is knowledge base completion, specifically, to learn which classes an entity is a member of. We propose FIGMENT to tackle this problem. FIGMENT is embedding-based and combines (i) a global model that scores based on aggregated contextual information of an entity and (ii) a context model that first scores the individual occurrences of an entity and then aggregates the scores. In our evaluation, FIGMENT strongly outperforms an approach to entity typing that relies on relations obtained by an open information extraction system.

1 Introduction

Natural language understanding (NLU) is not possible without knowledge about the world – partly so because world knowledge is needed for many NLP tasks that must be addressed as part of NLU; e.g., many coreference ambiguities can only be resolved based on world knowledge. It is also true because most NLU applications combine a variety of information sources that include both text sources and knowledge bases; e.g., question answering systems need access to knowledge bases like gazetteers. Thus, high-quality knowledge bases are critical for successful NLU.

Unfortunately, most knowledge bases are incomplete. The effort required to create knowledge bases is considerable and since the world changes, it will always continue. Knowledge bases are therefore always in need of updates and corrections. To address this problem, we present an information extraction method that can be used for knowledge base completion. In contrast to most other work on knowledge base completion, we focus on fine-grained *classification of entities* as opposed to *relations between entities*.

The goal of knowledge base completion is to acquire knowledge in general as opposed to detailed analysis of an individual context or sentence. Therefore, our approach is *corpus-level*: We infer the types of an entity by considering the set of all of its mentions in the corpus. In contrast, named entity recognition (NER) is *context-level* or *sentence-level*: NER infers the type of an entity in a particular context. As will be discussed in more detail in the following sections, the problems of corpus-level entity typing vs. context/sentence-level entity typing are quite different. This is partly because the objectives of optimizing accuracy on the context-level vs. optimizing accuracy on the corpus-level are different and partly because evaluation measures for corpus-level and context-level entity typing are different.

We define our problem as follows. Let K be a knowledge base that models a set E of entities, a set T of fine-grained classes or *types* and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t. Let C be a large corpus of text. Then, the problem we address in this paper is *corpus-level entity typing*: For a given pair of entity e and type t determine – based on the evidence available in C – whether e is a member of type t (i.e., $m(e, t) = 1$) or not (i.e., $m(e, t) = 0$) and update the membership relation m of K with this information.

We investigate two approaches to entity typing: a global model and a context model.

The *global model* aggregates all contextual information about an entity e from the corpus and then based on that, makes a classification decision on a particular type t – i.e., $m(e, t) = 0$ vs. $m(e, t) = 1$.

The *context model* first scores each individual

context of e as expressing type t or not. A final decision on the value of $m(e, t)$ is then made based on the distribution of context scores. One difficulty in knowledge base completion based on text corpora is that it is too expensive to label large amounts of text for supervised approaches. For our context model, we address this problem using *distant supervision*: we treat *all* contexts of an entity that can have type t as contexts of type t even though this assumption will in general be only true for a *subset* of these contexts. Thus, as is typical for distant supervision, the labels are incorrect in some contexts, but we will show that the labeling is good enough to learn a high-quality context model.

The global model is potentially more robust since it looks at all the available information at once. In contrast, the context model has the advantage that it can correctly predict types for which there are only a small number of reliable contexts. For example, in a large corpus we are likely to find a few reliable contexts indicating that "Barack Obama" is a bestselling author even though this evidence may be obscured in the global distribution because the vast majority of mentions of "Obama" do not occur in author contexts.

We implement the global model and the context model as well as a simple combination of the two and call the resulting system FIGMENT: FIne-Grained eMbedding-based Entity Typing. A key feature of FIGMENT is that it makes extensive use of distributed vector representations or *embeddings*. We compute embeddings for words as is standard in a large body of NLP literature, but we also compute embeddings for *entities* and for *types*. The motivation for using embeddings in these cases is (i) better generalization and (ii) more robustness against noise for text types like web pages. We compare the performance of FIGMENT with an approach based on Open Information Extraction (OpenIE).

The main contributions of this paper can be summarized as follows.

- We address the problem of corpus-level entity typing in a knowledge base completion setting. In contrast to other work that has focused on learning *relations* between entities, we learn *types* of entities.
- We show that context and global models for entity typing provide complementary information and combining them gives the best results.
- We use embeddings for words, entities and types to improve generalization and deal with noisy input.
- We show that our approach outperforms a system based on OpenIE relations when the input corpus consists of noisy web pages.

2 Related work

Named entity recognition (NER) is the task of detecting and classifying named entities in text. While most NER systems (e.g., Finkel et al. (2005)) only consider a small number of entity classes, recent work has addressed fine-grained NER (Yosef et al., 2012; Ling and Weld, 2012; Yogatama et al., 2015; Dong et al., 2015; Del Corro et al., 2015). These methods use a variety of lexical and syntactic features to segment and classify entity mentions. Some more recent work assumes the segmentation is known and only classifies entity mentions. Dong et al. (2015) use distributed representations of words in a hybrid classifier to classify mentions to 20 types. Yogatama et al. (2015) classify mentions to more fine-grained types by using different features for mentions and embedding labels in the same space. These methods as well as standard NER systems try to maximize correct classification of mentions in individual contexts whereas we aggregate individual contexts and evaluate on accuracy of entity-type assignments inferred from the entire corpus. In other words, their evaluation is *sentence-level* whereas ours is *corpus-level*.

Entity set expansion (ESE) is the problem of finding entities in a class (e.g., medications) given a seed set (e.g., {"Ibuprofen", "Maalox", "Prozac"}). The standard solution is pattern-based bootstrapping (Thelen and Riloff, 2002; Gupta and Manning, 2014). ESE is different from the problem we address because ESE starts with a small seed set whereas we assume that a large number of examples from a knowledge base (KB) is available. Initial experiments with the system of Gupta and Manning (2014) showed that it was not performing well for our task – this is not surprising given that it is designed for a task with properties quite different from entity typing.

More closely related to our work are the OpenIE systems NNPLB (Lin et al., 2012) and PEARL

(Nakashole et al., 2013) for *fine-grained typing of unlinkable and emerging entities*. Both systems first extract relation tuples from a corpus and then type entities based on the tuples they occur in (where NNPLB only uses the subject position for typing). To perform typing, NNPLB propagates activation from known members of a class to other entities whereas PEARL assigns types to the argument slots of relations. The main difference to FIGMENT is that we do not rely on relation extraction. In principle, we can make use of any context, not just subject and object positions. FIGMENT also has advantages for noisy text for which relation extraction can be challenging. This will be demonstrated in our evaluation on web text. Finally, our emphasis is on making yes-no decisions about possible types (as opposed to just ranking possibilities) for all entities (as opposed to just emerging or unlinkable entities). Our premise is that even existing entities in KBs are often not completely modeled and have entries that require enhancement. We choose NNPLB as our baseline.

The fine-grained typing of entities performed by FIGMENT can be used for *knowledge base completion (KBC)*. Most KBC systems focus on *relations* between entities, not on *types* as we do. Some generalize the patterns of relationships within the KB (Nickel et al., 2012; Bordes et al., 2013) while others use a combination of within-KB generalization and information extraction from text (Weston et al., 2013; Socher et al., 2013; Jiang et al., 2012; Riedel et al., 2013; Wang et al., 2014). Neelakantan and Chang (2015) address entity typing in a way that is similar to FIGMENT. Their method is based on KB information, more specifically entity descriptions in Wikipedia and Freebase. Thus, in contrast to our approach, their system is not able to type entities that are not covered by existing KBs. We infer classes for entities from a large corpus and do not assume that these entities occur in the KB.

Learning embeddings for words is standard in a large body of NLP literature (see Baroni et al. (2014) for an overview). In addition to words, we also learn *embeddings for entities and types*. Most prior work on entity embeddings (e.g., Weston et al. (2013), Bordes et al. (2013)) and entity and type embeddings (Zhao et al., 2015) has mainly used KB information as opposed to text corpora. Wang et al. (2014) learn embeddings of words and entities in the same space by replacing Wikipedia anchors with their corresponding entities. For our global model, we learn entity embedding in a similar way, but on a corpus with automatically annotated entities. For our context model, we learn and use type embeddings jointly with corpus words to improve generalization, a novel contribution of this paper to the best of our knowledge. We learn all our embeddings using `word2vec` (Mikolov et al., 2013).

Our problem can be formulated as *multi-instance multi-label* (MIML) learning (Zhou and Zhang, 2006), similar to the formulation for relation extraction by Surdeanu et al. (2012). In our problem, each example (entity) can have several instances (contexts) and each instance can have several labels (types). Similar to Zhou and Zhang (2006)'s work on scene classification, we also transform MIML into easier tasks. The global model transforms MIML into a multi-label problem by merging all instances of an example. The context model solves the problem by combining the instance-label scores to example-label scores.

3 Motivation and problem definition

3.1 Freebase

Large scale KBs like Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and Google knowledge graph are important NLP resources. Their structure is roughly equivalent to a graph in which entities are nodes and edges are relations between entities. Each node is also associated with one or more semantic classes, called types. These types are the focus of this paper.

We use Freebase, the largest available KB, in this paper. In Freebase, an entity can belong to several classes, e.g., "Barack Obama" is a member of 37 types including "US president" and "author". One *notable type* is also defined for each entity, e.g., "US-president" for "Obama" since it is regarded as his most prominent characteristic and the one that would be used to disambiguate references to him, e.g., to distinguish him from somebody else with the same name.

There are about 1500 types in Freebase, organized by domain; e.g., the domain "food" has types like "food", "ingredient" and "restaurant". Some types like "location" are very general, some are very fine-grained, e.g., "Vietnamese urban district". There are types that have a large number of instances like "citytown" and types that have very

few like "camera_sensor". Entities are defined as instances of types. They can have several types based on the semantic classes that the entity they are referring to is a member of – as in the above example of Barack Obama.

The types are not organized in a strict taxonomy even though there exists an *included type* relationship between types in Freebase. The reason is that for a user-generated KB it is difficult to maintain taxonomic consistency. For example, almost all instances of "author" are also instances of "person", but sometimes organizations author and publish documents. We follow the philosophy of Freebase and assume that the types do not have a hierarchical organization.

3.2 Incompleteness of knowledge base

Even though Freebase is the largest publicly available KB of its kind, it still has significant coverage problems; e.g., 78.5% of persons in Freebase do not have *nationality* (Min et al., 2013).

This is unavoidable, partly because Freebase is user-generated, partly because the world changes and Freebase has to be updated to reflect those changes. All existing KBs that attempt to model a large part of the world suffer from this incompleteness problem. Incompleteness is likely to become an even bigger problem in the future as the number of types covered by KBs like Freebase increases. As more and more fine-grained types are added, achieving good coverage for these new types using only human editors will become impossible.

The approach we adopt in this paper to address incompleteness of KBs is extraction of information from large text corpora. Text can be argued to be the main repository of the type of knowledge represented in KBs, so it is reasonable to attempt completing them based on text. There is in fact a significant body of work on corpus-based methods for extracting knowledge from text; however, most of it has addressed relation extraction, not the acquisition of type information – roughly corresponding to unary relations (see Section 2). In this paper, we focus on typing entities.

3.3 Entity linking

The first step in extracting information about entities from text is to reliably identify mentions of these entities. This problem of *entity linking* has some mutual dependencies with entity typing. Indeed, some recent work shows large improvements when entity typing and linking are jointly modeled (Ling et al., 2015; Durrett and Klein, 2014). However, there are constraints that are important for high-performance entity linking, but that are of little relevance to entity typing. For example, there is a large literature on entity linking that deals with coreference resolution and inter-entity constraints – e.g., "Naples" is more likely to refer to a US (resp. an Italian) city in a context mentioning "Fort Myers" (resp. "Sicily").

Therefore, we will only address entity typing in this paper and consider entity linking as an independent module that provides contexts of entities for FIGMENT. More specifically, we build FIGMENT on top of the output of an existing entity linking system and use FACC1,[1] an automatic Freebase annotation of ClueWeb (Gabrilovich et al., 2013). According to the FACC1 distributors, precision of annotated entities is around 80-85% and recall is around 70-85%.

3.4 FIGER types

Our goal is fine-grained typing of entities, but types like "Vietnamese urban district" are too fine-grained. To create a reliable setup for evaluation and to make sure that all types have a reasonable number of instances, we adopt the FIGER type set (Ling and Weld, 2012) that was created with the same goals in mind. FIGER consists of 112 tags and was created in an attempt to preserve the diversity of Freebase types while consolidating infrequent and unusual types through filtering and merging. For example, the Freebase types "dish", "ingredient", "food" and "cheese" are mapped to one type "food". See (Ling and Weld, 2012) for a complete list of FIGER types. We use "type" to refer to FIGER types in the rest of the paper.

4 Global, context and joint models

We address a problem setting in which the followings are given: a KB with a set of entities E, a set of types T and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t; and a large annotated corpus C in which mentions of E are linked. As mentioned before, we use FACC1 as our corpus.

In this problem setting, we address the task of *corpus-level fine-grained entity typing*: we want to infer from the corpus for each pair of entity e and type t whether $m(e, t) = 1$ holds, i.e., whether entity e is a member of type t.

[1] lemurproject.org/clueweb12/FACC1

We use three scoring models in FIGMENT: global model, context model and joint model. The models return a score $S(e,t)$ for an entity-type pair (e,t). $S(e,t)$ is an assessment of the extent to which it is true that the semantic class t contains e and we learn it by training on a subset of E. The trained models can be applied to large corpora and the resulting scores can be used for learning new types of entities covered in the KB as well as for typing new or unknown entities – i.e., entities not covered by the KB. To work for new or unknown entities, we would need an entity linking system such as the ones participating in TAC KBP (McNamee and Dang, 2009) that identifies and clusters mentions of them.

4.1 Global model

The global model (GM) scores possible types of entity e based on a *distributed vector representation* or *embedding* $\vec{v}(e) \in \mathbb{R}^d$ of e. $\vec{v}(e)$ can be learned from the entity-annotated corpus C.

Embeddings of words have been widely used in different NLP applications. The embedding of a word is usually derived from the distribution of its context words. The hypothesis is that words with similar meanings tend to occur in similar contexts (Harris, 1954) and therefore cooccur with similar context words. By extension, the assumption of our model is that entities with similar types tend to cooccur with similar context words.

To learn a score function $S_{\text{GM}}(e,t)$, we use a multilayer perceptron (MLP) with one shared hidden layer and an output layer that contains, for each type t in T, a logistic regression classifier that predicts the probability of t:

$$S_{\text{GM}}(e,t) = G_t\Big(\tanh\left(\mathbf{W}_{\text{input}}\vec{v}(e)\right)\Big)$$

where $\mathbf{W}_{\text{input}} \in \mathbb{R}^{h\times d}$ is the weight matrix from $\vec{v}(e) \in \mathbb{R}^d$ to the hidden layer with size h. G_t is the logistic regression classifier for type t that is applied on the hidden layer. The shared hidden layer is designed to exploit the dependencies among labels. Stochastic gradient descent (SGD) with AdaGrad (Duchi et al., 2011) and minibatches are used to learn the parameters.

4.2 Context model

For the context model (CM), we first learn a scoring function $S_{\text{c2t}}(c,t)$ for individual contexts c in the corpus. $S_{\text{c2t}}(c,t)$ is an assessment of how likely it is that an entity occurring in context c has type t. For example, consider the contexts c_1 = "he served SLOT cooked in wine" and c_2 = "she loves SLOT more than anything". SLOT marks the occurrence of an entity and it also shows that we do not care about the entity mention itself but only its context. For the type t = "food", $S_{\text{c2t}}(c_1,t)$ is high whereas $S_{\text{c2t}}(c_2,t)$ is low. This example demonstrates that some contexts of an entity like "beef" allow specific inferences about its type whereas others do not. We aim to learn a scoring function S_{c2t} that can distinguish these cases.

Based on the context scoring function S_{c2t}, we then compute the corpus-level CM scoring function S_{CM} that takes the scores $S_{\text{c2t}}(c_i,t)$ for all contexts of entity e in the corpus as input and returns a score $S_{\text{CM}}(e,t)$ that assesses the appropriateness of t for e. In other words, S_{CM} is:

$$S_{\text{CM}}(e,t) = g(U_{e,t}) \quad (1)$$

where $U_{e,t} = \{S_{\text{c2t}}(c_1,t), \ldots, S_{\text{c2t}}(c_n,t)\}$ is the set of scores for t based on the n contexts $c_1 \ldots c_n$ of e in the corpus. The function g is a summary function of the distribution of scores, e.g., the mean, median or maximum. We use the mean in this paper.

We now describe how we learn S_{c2t}. For training, we need contexts that are labeled with types. We do not have such a dataset in our problem setting, but we can use the contexts of linked entities as distantly supervised data. Specifically, assume that entity e has n types. For each mention of e in the corpus, we generate a training example with n labels, one for each of the n types of e.

For training S_{c2t}, a context c of a mention is represented as the concatenation of two vectors. One vector is the *average* of the embeddings of the $2l$ words to the left and right of the mention. The other vector is the *concatenation* of the embeddings of the $2k$ words to the left and right of the mention. E.g., for $k = 2$ and $l = 1$ the context c is represented as the vector: $\Phi(c) = \left[x_{-2}, x_{-1}, x_{+1}, x_{+2}, \text{avg}(x_{-1}, x_{+1})\right]$ where $x_i \in \mathbb{R}^d$ is the embedding of the context word at position i relative to the entity in position 0.

We train S_{c2t} on context representations that consist of embeddings because our goal is a robust model that works well on a wide variety of genres, including noisy web pages. If there are other entities in the contexts, we first replace them with their notable type to improve generalization. We learn word and type embeddings from the corpus C by replacing train entities with their notable type.

The next step is to score these examples. We use an MLP similar to the global model to learn S_{c2t}, which predicts the probability of type t occurring in context c:

$$S_{\text{c2t}}(c,t) = G_t\Big(\tanh\left(\mathbf{W}_{\text{input}}\Phi(c)\right)\Big)$$

where $\Phi(c) \in \mathbb{R}^n$ is the feature vector of the context c as described above, $n = (2k+1) * d$ and $\mathbf{W}_{\text{input}} \in \mathbb{R}^{h \times n}$ is the weight matrix from input to hidden layer with h units. Again, we use SGD with AdaGrad and minibatch training.

4.3 Joint model

Global model and context model have complementary strengths and weaknesses.

The strength of CM is that it is a direct model of the only source of reliable evidence we have: the context in which the entity occurs. This is also the way a human would ordinarily do entity typing: she would determine if a specific context in which the entity occurs implies that the entity is, say, an author or a musician and type it accordingly. The order of words is of critical importance for the accurate assessment of a context and CM takes it into account. A well-trained CM will also work for cases for which GM is not applicable. In particular, if the KB contains only a small number of entities of a particular type, but the corpus contains a large number of contexts of these entities, then CM is more likely to generalize well.

The main weakness of CM is that a large proportion of contexts does not contain sufficient information to infer all types of the entity; e.g., based on our distant supervised training data, we label every context of "Obama" with "author", "politician" and Obama's other types in the KB. Thus, CM is trained on a noisy training set that contains only a relatively small number of informative contexts.

The main strength of GM is that it bases its decisions on the entire evidence available in the corpus. This makes it more robust. It is also more efficient to train since its training set is by a factor of $|M|$ smaller than the training set of CM where $|M|$ is the average number of contexts per entity.

The disadvantage of GM is that it does not work well for rare entities since the aggregated representation of an entity may not be reliable if it is based on few contexts. It is also less likely to work well for non-dominant types of an entity which might be swamped by dominant types; e.g., the author contexts of "Obama" may be swamped by the politician contexts and the overall context signature of the entity "Obama" may not contain enough signal to infer that he is an author. Finally, methods for learning embeddings like `word2vec` are bag-of-word approaches. Therefore, word order information – critical for many typing decisions – is lost.

Since GM and CM models are complementary, a combination model should work better. We test this hypothesis for the simplest possible joint model (JM), which adds the scores of the two individual models:

$$S_{\text{JM}}(e,t) = S_{\text{GM}}(e,t) + S_{\text{CM}}(e,t)$$

5 Experimental setup and results

5.1 Setup

Baseline: Our baseline system is the OpenIE system no-noun-phrase-left-behind (NNPLB) by Lin et al. (2012) (see Section 2). Our reimplementation performs on a par with published results.[2] We use NNPBL as an alternative way of computing scores $S(e,t)$. Scores of the four systems we compare – NNPBL, GM, CM, JM – are processed the same way to perform entity typing (see below).

Corpus: We select a subset of about 7.5 million web pages, taken from the first segment of ClueWeb12,[3] from different crawl types: 1 million Twitter links, 120,000 WikiTravel pages and 6.5 million web pages. This corpus is preprocessed by eliminating HTML tags, replacing all numbers with "7" and all web links and email addresses with "HTTP", filtering out sentences with length less than 40 characters, and finally doing a simple tokenization. We merge the text with the FACC1 annotations. The resulting corpus has 4 billion tokens and 950,000 distinct entities. We use the 2014-03-09 Freebase data dump as our KB.

Entity datasets: We consider all entities in the corpus whose notable types can be mapped to one of the 112 FIGER types, based on the mapping provided by FIGER. 750,000 such entities form our set of entities. 10 out of 112 FIGER types have no entities in this set.[4]

[2] The precision of our implementation on the dataset of three million relation triples distributed by (Lin et al., 2012) is 60.7% compared to 59.8% and 61% for tail and head entities reported by Lin et al. (2012).

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

[4] The reason is that the FIGER mapping uses Freebase user-created classes. The 10 missing types are not the notable type of any entity in Freebase.

We run the OpenIE system Reverb (Fader et al., 2011) to extract relation triples of the form <subject, relation, object>. Since NNPLB only considers entities in the subject position, we filter out triples whose subject is not an entity. The size of the remaining set of triples is 4,000,000. For a direct comparison with NNPLB, we divide the 750,000 entities into those that occur in subject position in one of the extracted triples (about 250,000 *subject entities* or SE) and those that do not (about 500,000 *non-subject entities* or NSE). We split SE 50:20:30 into train, dev and test sets. The average and median number of FIGER types of the training entities are 1.8 and 2, respectively. We use NSE to evaluate performance of FIGMENT on entities that do not occur in subject position.[5]

Context sampling: For S_{c2t}, we create train', dev' and test' sets of *contexts* that correspond to train, dev and test sets of *entities*. Because the number of contexts is unbalanced for both entities and types and because we want to accelerate training and testing, we downsample contexts. For the set train', we use the notable type feature of Freebase: For each type t, we take contexts from the mentions of those entities whose notable type is t. Recall, however, that each context is labeled with all types of its entity – see Section 4.2.

Then if the number of contexts for t is larger than a minimum, we sample the contexts based on the number of training entities of t. We set the minimum to 10,000 and constrain the number of samples for each t to 20,000. Also, to reduce the effect of distant supervision, entities with fewer distinct types are preferred in sampling to provide discriminative contexts for their notable types. For test' and dev' sets, we sample 300 and 200 random contexts, respectively, for each entity.

System setup: As the baseline, we apply NNPLB to the 4 million extracted triples. To learn entity embeddings for GM, we run `word2vec` (skipgram, 200 dimensions, window size 5) on a version of the corpus in which entities have been replaced by their Freebase IDs, based on the FACC1 annotation. We then train MLP with number of hidden units $h = 200$ on the embeddings of training entities until the error on dev entities stops decreasing.

Our reasoning for the unsupervised training setup is that we do not use any information about the types of entities (e.g., no entities annotated by humans with types) when we run an unsupervised algorithm like `word2vec`. In a real-world application of FIGMENT to a new corpus, we would first run `word2vec` on the merger of our corpus and the new corpus, retrain GM on training entities and finally apply it to entities in the new corpus. This scenario is simulated by our setup.

Recall that the input to CM consists of $2k$ unit embeddings and the average of $2l$ unit embeddings where we use the term *unit* to refer to both words and types. We set k to 4 and l to 5. To learn embeddings for units, we first exclude lines containing test entities, and then replace each entity with its notable type. Then, we run `word2vec` (skipgram, 100 dimensions, window size 5) on this new corpus and learn embeddings for words and types.

Using the embeddings as input representations, we train S_{c2t} on train' until error on dev' stops decreasing. We set the number of hidden units to 300. We then apply the trained scoring function S_{c2t} to test' and get the scores $S_{\text{c2t}}(c, t)$ for test' contexts. As explained in Section 4.2, we compute the corpus-level scores $S_{\text{CM}}(e, t)$ for each entity by averaging its context-level scores (see Equation 1).

Ranking evaluation: This evaluation shows how well the models rank types for entities. The ranking is based on the scores $S(e, t)$ produced by the different models and baselines. Similar to the evaluation performed by Lin et al. (2012), we use precision at 1 (P@1) and breakeven point (BEP, Boldrin and Levine (2008)). BEP is F_1 at the point in the ranked list at which precision and recall have the same value.

Classification evaluation: This evaluation demonstrates the quality of the thresholded assignment decisions produced by the models. These measures more directly express how well FIGMENT would succeed in enhancing the KB with new information since for each pair (e, t), we have to make a binary decision about whether to put it in the KB or not. We compare our decisions with the gold KB information.

Our evaluation measures are (i) accuracy: an entity is correct if all its types and no incorrect types are assigned to it; (ii) micro average: F_1 of all type-entity assignment decisions; (iii) entity macro average F_1: F_1 of types assigned to an entity, averaged over entities; (iv) type macro average F_1: F_1 of entities assigned to a type, averaged over types.

[5]The entity datasets are available at `http://cistern.cis.lmu.de/figment`

The assignment decision is made based on thresholds, one per type, for each $S(e,t)$. We select the threshold that maximizes F_1 of entities assigned to the type on dev.

5.2 Results

Table 1 presents results for the ranking evaluation as well as for the first three measures of the classification evaluation. MFT is the most frequent type baseline that ranks types according to their frequency in train. We also show the results for head entities (frequency higher than 100) and tail entities (frequency less than 5). The performance of the systems is in this order: JM $>$ GM $>$ CM $>$ NNPLB $>$ MFT.

Table 2 shows the results of the fourth classification measure, type macro average F_1, for all, head (more than 3000 train entities, 11 types), and tail (less than 200 train entities, 36 types) types. The ordering of models for Table 2 is in line with Table 1: JM $>$ GM $>$ CM $>$ NNPLB $>$ MFT.

We can easily run FIGMENT for **non-subject entities** (NSE) exactly the same way we have run it for subject entities. We test our JM on the 67,000 NSE entities with a frequency of more than 10. The top ranked type returned for 73.5% of entities was correct. Thus, due to our ability to deal with NSE, we can type an additional 50,000 entities correctly.

6 Analysis

Effect of window size in CM: Table 3 explores the effect of using different context sizes. Recall that CM was trained with $2k = 8$ for the concatenation and $2l = 10$ for the average window size. We change $2k$ from 0 to 14 while keeping $2l = 10$. The number of hidden units used in each model is also reported. The table shows that CM can leverage larger context sizes well.

Poor results of NNPLB: NNPLB is mostly hampered by Reverb, which did not work well on the noisy web corpus. As a result, the quality of the extracted relations – which NNPLB entity typing is based on – is too low for reliable typing decisions. The good results of NNPLB on their non-noisy published relation triples confirm that. On the three million relation triples, when mapping Freebase types to FIGER, P@1 of NNPLB is .684; when limiting entities to those with more than 10 relations, the results improve to .776.

GM performs better than CM and JM performs best: The fact that GM outperforms CM shows that decisions based on one global vector of an entity work better than aggregating multiple weak decisions on their contexts. That is clearest for tail entities – where one bad context can highly influence the final decision – and for tail types, which CM was not able to distinguish from other similar types. However, the good results of the simple JM confirm that the score distributions in CM do help. As an example, consider one of the test entities that is an "author". GM and CM wrongly predict "written_work" and "artist", respectively, but JM correctly outputs "author".

Errors of CM: Many CM errors are caused by its simple input representation: it has to learn all linguistic abstractions that it wants to rely on from the training set. One manifestation of this problem is that CM confuses the types "food" and "restaurant". There are only few linguistic contexts in which entities of these types can be exchanged for each other. On the other hand, the context words they cooccur with in a bag-of-words (BOW) sense are very similar. Thus, this indicates that CM pays too much attention to BOW information and that its representation of contexts is limited in terms of generalization.

Assumptions that result in errors: The performance of all models suffers from a number of assumptions we made in our training / evaluation setup that are only approximately true.

The first assumption is that FACC1 is correct. But it has a precision of only 80-85% and this caused many errors. An example is the lunar crater "Buffon" in Freebase, a "location". Its predicted type is "athlete" because some FACC1 annotations of the crater link it to the Italian goalkeeper.

The second assumption of our evaluation setup is the completeness of Freebase. There are about 2,600 entities with the single type "person" in SE test. For 62% of the errors on this subset, the top predicted type is a subtype of person: "author", "artist" etc. We manually typed a random subset of 50 and found that the predicted type is actually correct for 44 of these entities.

The last assumption is the mapping from Freebase to FIGER. Some common Freebase types like "award-winner" are not mapped. This negatively affects evaluation measures for many entities. On the other hand, the resulting types do not have a balanced number of instances. Based on our training entities, 11 types (e.g., "law") have less than

	all entities					head entities					tail entities				
	P@1	BEP	acc	mic	mac	P@1	BEP	acc	mic	mac	P@1	BEP	acc	mic	mac
MFT	.101	.406	-	-	-	.111	.410	-	-	-	.097	.394	-	-	-
NNPLB	.365	.480	.000	.099	.096	.378	.503	.000	.114	.109	.368	.474	.000	.086	.084
CM	.694	.734	.299	.668	.635	.713	.751	.385	.738	.702	.608	.661	.118	.487	.452
GM	.805	.856	.426	.733	.688	.869	.899	.489	.796	.769	.665	.757	.299	.578	.510
JM	**.816**	**.860**	**.435**	**.743**	**.699**	**.874**	**.900**	**.500**	**.803**	**.776**	**.688**	**.764**	**.306**	**.601**	**.532**

Table 1: Ranking and classification results for SE entities. P@1 and BEP are ranking measures. Accuracy (acc), micro (mic) and macro (mac) are classification measures.

	all types	head types	tail types
NNPLB	.092	.246	.066
CM	.406	.662	.268
GM	.533	.725	.387
JM	**.545**	**.734**	**.407**

Table 2: Type macro average F_1 for all, head and tail types

2k	0	2	4	6	8	10	12	14
h	50	100	200	250	300	400	450	450
micro	.576	.613	.672	.673	.668	.674	**.680**	.674
P@1	.663	.685	.687	.718	.694	**.744**	.722	.742

Table 3: Effect of the context size $2k$ in CM (2k: context size, h: number of hidden units in MLP)

50 instances while 26 types (e.g., "software") have more than 1000 instances. Even sampling the contexts could not resolve this problem and this led to low performance on tail types.

7 Future work

The performance of FIGMENT is poor for tail types and entities. We plan to address this in the future (i) by running FIGMENT on larger corpora, (ii) by refining the FIGER type set to cover more Freebase entities, (iii) by exploiting a hierarchy over types and (iv) by exploring more complex input representations of the context for the CM.

FIGMENT's context model can in principle be based on any system that provides entity-type assessment scores for individual contexts. Thus, as an alternative to our scoring model $S_{\text{c2t}}(c, t)$, we could use sentence-level entity classification systems such as FIGER (Ling and Weld, 2012) and (Yogatama et al., 2015)'s system. These systems are based on linguistic features different from the input representation we use, so a comparison with our embedding-based approach is interesting. Our assumption is that FIGMENT is more robust against noise, but investigation is needed.

The components of the version of FIGMENT we presented, in particular, context model and global model, do not use features derived from the mention of an entity. Our assumption was that such features are less useful for fine-grained entity typing. However, there are clearly some types for which mention-based features are useful (e.g., medications or organizations referred to by abbreviations), so we will investigate the usefulness of such features in the future.

8 Conclusion

We presented FIGMENT, a corpus-level system that uses contextual information for entity typing. We designed two scoring models for pairs of entities and types: a global model that scores based on aggregated context information and a context model that aggregates the scores of individual contexts. We used embeddings of words, entities and types to represent contextual information. Our experimental results show that global model and context model provide complementary information for entity typing. We demonstrated that, comparing with an OpenIE-based system, FIGMENT performs well on noisy web pages.

Acknowledgements. Thanks to the anonymous reviewers for their valuable comments. This work was supported by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/8-2, SPP 1335).

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 238–247.

Michele Boldrin and David K. Levine. 2008. *Against intellectual monopoly*. Cambridge University Press Cambridge.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Irreflexive and hierarchical relations as translations. *CoRR*, abs/1304.7158.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878, Lisbon, Portugal, September. Association for Computational Linguistics.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1243–1249.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *TACL*, 2:477–490.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1535–1545.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Sonal Gupta and Christopher D. Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 98–108.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10:146–162.

Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. 2012. Link prediction in multi-relational graphs using additive models. In *Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data, Boston, USA, November 11, 2012*, pages 1–12.

Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 893–903.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *TACL*, 3:315–328.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

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

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 777–782.

Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained semantic typing of emerging entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1488–1497.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 515–525.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *World Wide Web Conference*, pages 271–280.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 74–84.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 455–465.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Stroudsburg, PA, USA*, pages 214–221.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1591–1601.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1366–1371.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Representation learning for measuring entity relatedness with rich information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1412–1418.

Zhi-Hua Zhou and Min-Ling Zhang. 2006. Multi-instance multi-label learning with application to scene classification. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 1609–1616.

Chapter 3

Intrinsic Subspace Evaluation of Word Embedding Representations

Intrinsic Subspace Evaluation of Word Embedding Representations

Yadollah Yaghoobzadeh and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
yadollah@cis.lmu.de

Abstract

We introduce a new methodology for intrinsic evaluation of word representations. Specifically, we identify four fundamental criteria based on the characteristics of natural language that pose difficulties to NLP systems; and develop tests that directly show whether or not representations contain *the subspaces necessary to satisfy these criteria*. Current intrinsic evaluations are mostly based on the overall similarity or *full-space similarity* of words and thus view vector representations as *points*. We show the limits of these point-based intrinsic evaluations. We apply our evaluation methodology to the comparison of a count vector model and several neural network models and demonstrate important properties of these models.

1 Introduction

Distributional word representations or *embeddings* are currently an active area of research in natural language processing (NLP). The motivation for embeddings is that knowledge about words is helpful in NLP. Representing words as vocabulary indexes may be a good approach if large training sets allow us to learn everything we need to know about a word to solve a particular task; but in most cases it helps to have a representation that contains distributional information and allows inferences like: "above" and "below" have similar syntactic behavior or "engine" and "motor" have similar meaning.

Several methods have been introduced to assess the quality of word embeddings. We distinguish two different types of evaluation in this paper: (i) *extrinsic evaluation* evaluates embeddings in an NLP application or task and (ii) *intrinsic evaluation* tests the quality of representations independent of a specific NLP task.

Each single word is a combination of a large number of morphological, lexical, syntactic, semantic, discourse and other features. Its embedding should accurately and consistently represent these features, and ideally a good evaluation method must clarify this and give a way to analyze the results. The goal of this paper is to build such an evaluation.

Extrinsic evaluation is a valid methodology, but it does not allow us to understand the properties of representations without further analysis; e.g., if an evaluation shows that embedding A works better than embedding B on a task, then that is not an analysis of the causes of the improvement. Therefore, extrinsic evaluations do not satisfy our goals.

Intrinsic evaluation analyzes the generic quality of embeddings. Currently, this evaluation mostly is done by testing overall distance/similarity of words in the embedding space, i.e., it is based on viewing word representations as points and then computing *full-space similarity*. The assumption is that the high dimensional space is smooth and similar words are close to each other. Several datasets have been developed for this purpose, mostly the result of human judgement; see (Baroni et al., 2014) for an overview. We refer to these evaluations as *point-based* and as *full-space* because they consider embeddings as points in the space – sub-similarities in subspaces are generally ignored.

Point-based intrinsic evaluation computes a score based on the full-space similarity of two words: a single number that generally does not say anything about the underlying reasons for a lower or higher value of full-space similarity. This makes it hard to interpret the results of point-based evaluation and may be the reason that contradictory results have been published; e.g., based on

point-based evaluation, some papers have claimed that count-based representations perform as well as learning-based representations (Levy and Goldberg, 2014a). Others have claimed the opposite (e.g., Mikolov et al. (2013), Pennington et al. (2014), Baroni et al. (2014)).

Given the limits of current evaluations, we propose a new methodology for intrinsic evaluation of embeddings by identifying generic fundamental criteria for embedding models that are important for representing features of words accurately and consistently. We develop corpus-based tests using supervised classification that directly show whether the representations contain the information necessary to meet the criteria or not. The fine-grained corpus-based supervision makes the sub-similarities of words important by looking at the subspaces of word embeddings relevant to the criteria, and this enables us to give direct insights into properties of representation models.

2 Related Work

Baroni et al. (2014) evaluate embeddings on different intrinsic tests: similarity, analogy, synonym detection, categorization and selectional preference. Schnabel et al. (2015) introduce tasks with more fine-grained datasets. These tasks are unsupervised and generally based on cosine similarity; this means that only the overall direction of vectors is considered or, equivalently, that *words are modeled as points* in a space and only their full-space distance/closeness is considered. In contrast, we test embeddings in a classification setting and *different subspaces of embeddings are analyzed*. Tsvetkov et al. (2015) evaluate embeddings based on their correlations with WordNet-based linguistic embeddings. However, correlation does not directly evaluate how accurately and completely an application can extract a particular piece of information from an embedding.

Extrinsic evaluations are also common (cf. (Li and Jurafsky, 2015; Köhn, 2015; Lai et al., 2015)). Li and Jurafsky (2015) conclude that embedding evaluation must go beyond human-judgement tasks like similarity and analogy. They suggest to evaluate on NLP tasks. Köhn (2015) gives similar suggestions and also recommends the use of supervised methods for evaluation. Lai et al. (2015) evaluate embeddings in different tasks with different setups and show the contradictory results of embedding models on different tasks. Idiosyncrasies of different downstream tasks can affect extrinsic evaluations and result in contradictions.

3 Criteria for word representations

Each word is a combination of different properties. Depending on the language, these properties include lexical, syntactic, semantic, world knowledge and other features. We call these properties *facets*. The ultimate goal is to learn representations for words that accurately and consistently contain these facets. Take the facet gender (GEN) as an example. We call a representation 100% *accurate* for GEN if information it contains about GEN is always accurate; we call the representation 100% *consistent* for GEN if the representation of every word that has a GEN facet contains this information.

We now introduce four important criteria that a representation must satisfy to represent facets accurately and consistently. These criteria are applied across different problems that NLP applications face in the effective use of embeddings.

Nonconflation. A word embedding must keep the evidence from different local contexts separate – "do not conflate" – because each context can infer specific facets of the word. Embeddings for different word forms with the same stem, like plural and singular forms or different verb tenses, are examples vulnerable to conflation because they occur in similar contexts.

Robustness against sparseness. One aspect of natural language that poses great difficulty for statistical modeling is sparseness. Rare words are common in natural language and embedding models must learn useful representations based on a small number of contexts.

Robustness against ambiguity. Another central problem when processing words in NLP is lexical ambiguity (Cruse, 1986; Zhong and Ng, 2010). Polysemy and homonymy of words can make it difficult for a statistical approach to generalize and infer well. Embeddings should fully represent all senses of an ambiguous word. This criterion becomes more difficult to satisfy as distributions of senses become more skewed, but a robust model must be able to overcome this.

Accurate and consistent representation of multifacetedness. This criterion addresses settings with large numbers of facets. It is based on the following linguistic phenomenon, a phenomenon that occurs frequently crosslinguistically

(Comrie, 1989). (i) Words have a large number of facets, including phonetic, morphological, syntactic, semantic and topical properties. (ii) Each facet by itself constitutes a small part of the overall information that a representation should capture about a word.

4 Experimental setup and results

We now design experiments to directly evaluate embeddings on the four criteria. We proceed as follows. First, we design a probabilistic context free grammar (PCFG) that generates a corpus that is a manifestation of the underlying phenomenon. Then we train our embedding models on the corpus. The embeddings obtained are then evaluated in a classification setting, in which we apply a linear SVM (Fan et al., 2008) to classify embeddings. Finally, we compare the classification results for different embedding models and analyze and summarize them.

Selecting embedding models. Since this paper is about developing a new evaluation methodology, the choice of models is not important as long as the models can serve to show that the proposed methodology reveals interesting differences with respect to the criteria.

On the highest level, we can distinguish two types of distributional representations. *Count vectors* (Sahlgren, 2006; Baroni and Lenci, 2010; Turney and Pantel, 2010) live in a high-dimensional vector space in which each dimension roughly corresponds to a (weighted) count of cooccurrence in a large corpus. *Learned vectors* are learned from large corpora using machine learning methods: unsupervised methods such as LSI (e.g., Deerwester et al. (1990), Levy and Goldberg (2014b)) and supervised methods such as neural networks (e.g., Mikolov et al. (2013)) and regression (e.g., Pennington et al. (2014)). Because of the recent popularity of learning-based methods, we consider one count-based and five learning-based distributional representation models.

The learning-based models are: (i) vLBL (henceforth: LBL) (vectorized log-bilinear language model) (Mnih and Kavukcuoglu, 2013), (ii) SkipGram (henceforth: SKIP) (skipgram bag-of-word model), (iii) CBOW (continuous bag-of-word model (Mikolov et al., 2013), (iv) Structured SkipGram (henceforth SSKIP), (Ling et al., 2015) and CWindow (henceforth CWIN) (continuous window model) (Ling et al., 2015). These models learn word embeddings for input and target spaces using neural network models.

1	$P(aVb\|S)$	=	1/4	
2	$P(bVa\|S)$	=	1/4	
3	$P(aWa\|S)$	=	1/8	
4	$P(aWb\|S)$	=	1/8	
5	$P(bWa\|S)$	=	1/8	
6	$P(bWb\|S)$	=	1/8	
7	$P(v_i\|V)$	=	1/5	$0 \leq i \leq 4$
8	$P(w_i\|W)$	=	1/5	$0 \leq i \leq 4$

Figure 1: Global conflation grammar. Words v_i occur in a subset of the contexts of words w_i, but the global count vector signatures are the same.

For a given context, represented by the input space representations of the left and right neighbors $\vec{v}_{i-1}$ and $\vec{v}_{i+1}$, LBL, CBOW and CWIN predict the target space $\vec{v}_i$ by combining the contexts. LBL combines $\vec{v}_{i-1}$ and $\vec{v}_{i+1}$ linearly with position dependent weights and CBOW (resp. CWIN) combines them by adding (resp. concatenation). SKIP and SSKIP predict the context words v_{i-1} or v_{i+1} given the input space $\vec{v}_i$. For SSKIP, context words are in different spaces depending on their position to the input word. In summary, CBOW and SKIP are learning embeddings using bag-of-word (BoW) models, but the other three, CWIN, SSKIP and LBL, are using position dependent models. We use `word2vec`[1] for SKIP and CBOW, `wang2vec`[2] for SSKIP and CWIN, and Lai et al. (2015)'s implementation[3] for LBL.

The count-based model is position-sensitive PPMI, Levy and Goldberg (2014a)'s explicit vector space representation model.[4] For a vocabulary of size V, the representation $\vec{w}$ of w is a vector of size $4V$, consisting of four parts corresponding to the relative positions $r \in \{-2, -1, 1, 2\}$ with respect to occurrences of w in the corpus. The entry for dimension word v in the part of $\vec{w}$ corresponding to relative position r is the PPMI (positive pointwise mutual information) weight of w and v for that relative position. The four parts of the vector are length normalized. In this paper, we use only two relative positions: $r \in \{-1, 1\}$, so each $\vec{w}$ has two parts, corresponding to immediate left and right neighbors.

[1] `code.google.com/archive/p/word2vec`
[2] `github.com/wlin12/wang2vec`
[3] `github.com/licstar/compare`
[4] `bitbucket.org/omerlevy/hyperwords`

4.1 Nonconflation

Grammar. The PCFG grammar shown in Figure 1 generates v_i words that occur in two types of contexts: a-b (line 1) and b-a (line 2); and w_i words that also occur in these two contexts (lines 4 and 5), but in addition occur in a-a (line 3) and b-b (line 6) contexts. As a result, the set of contexts in which v_i and w_i occur is different, but if we simply count the number of occurrences in the contexts, then v_i and w_i cannot be distinguished.

Dataset. We generated a corpus of 100,000 sentences. Words that can occur in a-a and b-b contexts constitute the positive class, all other words the negative class. The words v_3, v_4, w_3, w_4 were assigned to the test set, all other words to the training set.

Results. We learn representations of words by our six models and train one SVM per model; it takes a word representation as input and outputs +1 (word can occur in a-a/b-b) or -1 (it cannot). The SVMs trained on PPMI and CBOW representations assigned all four test set words to the negative class; in particular, w_3 and w_4 were incorrectly classified. Thus, the accuracy of classification for these models (50%) was not better than random. The SVMs trained on LBL, SSKIP, SSKIP and CWIN representations assigned all four test set words to the correct class: v_3 and v_4 were assigned to the negative class and w_3 and w_4 were assigned to the positive class.

Discussion. The property of embedding models that is relevant here is that PPMI is an *aggregation model*, which means it calculates aggregate statistics for each word and then computes the final word embedding from these aggregate statistics. In contrast, all our learning-based models are *iterative models*: they iterate over the corpus and each local context of a word is used as a training instance for learning its embedding.

For iterative models, it is common to use composition of words in the context, as in LBL, CBOW and CWIN. Non-compositional iterative models like SKIP and SSKIP are also popular. Aggregation models can also use composite features from context words, but these features are too sparse to be useful. The reason that the model of Agirre et al. (2009) is rarely used is precisely its inability to deal with sparseness. All widely used distributional models employ individual word occurrences as basic features.

The bad PPMI results are explained by the fact

1	$P(AVB\|S)$	=	1/2	
2	$P(CWD\|S)$	=	1/2	
3	$P(a_i\|A)$	=	1/10	$0 \leq i \leq 9$
4	$P(b_i\|B)$	=	1/10	$0 \leq i \leq 9$
5	$P(c_i\|C)$	=	1/10	$0 \leq i \leq 9$
6	$P(d_i\|D)$	=	1/10	$0 \leq i \leq 9$
7	$P(v_i\|V)$	=	1/10	$0 \leq i \leq 9$
8	$P(w_i\|W)$	=	1/10	$0 \leq i \leq 9$
9	$L' =$	$L(S)$		
10		$\cup \{a_i u_i b_i \| 0 \leq i \leq 9\}$		
11		$\cup \{c_i x_i d_i \| 0 \leq i \leq 9\}$		

Figure 2: In language L', frequent v_i and rare u_i occur in a-b contexts; frequent w_i and rare x_i occur in c-d contexts. Word representations should encode possible contexts (a-b vs. c-d) for both frequent and rare words.

that it is an aggregation model: the PPMI model cannot distinguish two words with the same global statistics – as is the case for, say, v_3 and w_3. The bad result of CBOW is probably connected to its weak (addition) composition of context, although it is an iterative compositional model. Simple representation of context words with iterative updating (through backpropagation in each training instance), can influence the embeddings in a way that SKIP and SSKIP get good results, although they are non-compositional.

As an example of conflation occurring in the English Wikipedia, consider this simple example. We replace all single digits by "7" in tokenization. We learn PPMI embeddings for the tokens and see that among the one hundred nearest neighbors of "7" are the days of the week, e.g., "Friday". As an example of a conflated feature consider the word "falls" occurring immediately to the right of the target word. The weekdays as well as single digits often have the immediate right neighbor "falls" in contexts like "Friday falls on a public holiday" and "2 out of 3 falls match" – tokenized as "7 out of 7 falls match" – in World Wrestling Entertainment (WWE). The left contexts of "Friday" and "7" are different in these contexts, but the PPMI model does not record this information in a way that would make the link to "falls" clear.

4.2 Robustness against sparseness

Grammar. The grammar shown in Figure 2 generates frequent v_i and rare u_i in a-b contexts (lines 1 and 9); and frequent w_i and rare x_i in c-d contexts (lines 2 and 10). The language generated by the PCFG on lines 1–8 is merged on lines 9–11 with the ten contexts $a_0u_0b_0 \ \ldots \ a_9u_9b_9$ (line 9)

and the ten contexts $c_0x_0d_0$ ... $c_9x_9d_9$ (line 10); that is, each of the u_i and x_i occurs exactly once in the merged language L', thus modeling the phenomenon of sparseness.

Dataset. We generated a corpus of 100,000 sentences using the PCFG (lines 1–8) and added the 20 rare sentences (lines 9–11). We label all words that can occur in c-d contexts as positive and all other words as negative. The singleton words u_i and x_i were assigned to the test set, all other words to the training set.

Results. After learning embeddings with different models, the SVM trained on PPMI representations assigned all twenty test words to the negative class. This is the correct decision for the ten u_i (since they cannot occur in a c-d context), but the incorrect decision for the x_i (since they can occur in a c-d context). Thus, the accuracy of classification was 50% and not better than random. The SVMs trained on learning-based representations classified all twenty test words correctly.

Discussion. Representations of rare words in the PPMI model are sparse. The PPMI representations of the u_i and x_i only contain two nonzero entries, one entry for an a_i or c_i (left context) and one entry for a b_i or d_i (right context). Given this sparseness, it is not surprising that representations are not a good basis for generalization and PPMI accuracy is random.

In contrast, learning-based models learn that the a_i, b_i, c_i and d_i form four different distributional classes. The final embeddings of the a_i after learning is completed are all close to each other and the same is true for the other three classes. Once the similarity of two words in the same distributional class (say, the similarity of a_5 and a_7) has been learned, the contexts for the u_i (resp. x_i) look essentially the same to embedding models as the contexts of the v_i (resp. w_i). Thus, the embeddings learned for the u_i will be similar to those learned for the v_i. This explains why learning-based representations achieve perfect classification accuracy.

This sparseness experiment highlights an important difference between count vectors and learned vectors. Count vector models are less robust in the face of sparseness and noise because they base their representations on individual contexts; the overall corpus distribution is only weakly taken into account, by way of PPMI weighting. In contrast, learned vector models make much better use of the overall corpus distribution and they can leverage second-order effects for learning improved representations. In our example, the second order effect is that the model first learns representations for the a_i, b_i, c_i and d_i and then uses these as a basis for inferring the similarity of u_i to v_i and of x_i to w_i.

1	$P(AV_1B\|S)$	=10/20	
2	$P(CW_1D\|S)$	=9/20	
3	$P(CW_2D\|S)$	$=\beta\cdot$1/20	
4	$P(AW_2B\|S)$	$=(1-\beta)\cdot$1/20	
5	$P(a_i\|A)$	=1/10	$0 \leq i \leq 9$
6	$P(b_i\|B)$	=1/10	$0 \leq i \leq 9$
7	$P(c_i\|C)$	=1/10	$0 \leq i \leq 9$
8	$P(d_i\|D)$	=1/10	$0 \leq i \leq 9$
9	$P(v_i\|V_1)$	=1/50	$0 \leq i \leq 49$
10	$P(w_i\|W_1)$	=1/45	$5 \leq i \leq 49$
11	$P(w_i\|W_2)$	=1/5	$0 \leq i \leq 4$

Figure 3: Ambiguity grammar. v_i and $w_5 \ldots w_{49}$ occur in a-b and c-d contexts only, respectively. $w_0 \ldots w_4$ are ambiguous and occur in both contexts.

4.3 Robustness against ambiguity

Grammar. The grammar in Figure 3 generates two types of contexts that we interpret as two different meanings: a-b contexts (lines 1,4) and c-d contexts (lines 2, 3). v_i occur only in a-b contexts (line 1), $w_5 \ldots w_{49}$ occur only in c-d contexts (line 2); thus, they are unambiguous. $w_0 \ldots w_4$ are ambiguous and occur with probability β in c-d contexts (line 3) and with probability $(1-\beta)$ in a-b contexts (lines 3, 4). The parameter β controls the skewedness of the sense distribution; e.g., the two senses are equiprobable for $\beta = 0.5$ and the second sense (line 4) is three times as probable as the first sense (line 3) for $\beta = 0.25$.

Dataset. The grammar specified in Figure 3 was used to generate a training corpus of 100,000 sentences. Label criterion: A word is labeled positive if it can occur in a c-d context, as negative otherwise. The test set consists of the five ambiguous words $w_0 \ldots w_4$. All other words are assigned to the training set.

Linear SVMs were trained for the binary classification task on the train set. 50 trials of this experiment were run for each of eleven values of β: $\beta = 2^{-\alpha}$ where $\alpha \in \{1.0, 1.1, 1.2, \ldots, 2.0\}$. Thus, for the smallest value of α, $\alpha = 1.0$, the two senses have the same frequency; for the largest value of α, $\alpha = 2.0$, the dominant sense is three times as frequent as the less frequent sense.

Results. Figure 4 shows accuracy of the classi-

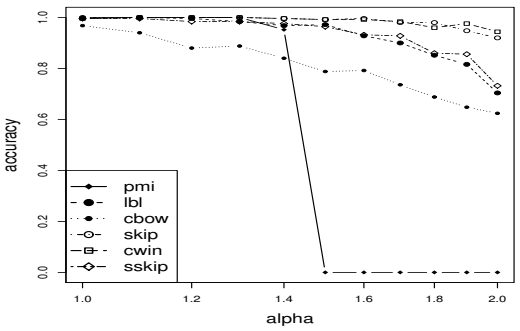

Figure 4: SVM classification results for the ambiguity dataset. X-axis: $\alpha = -\log_2 \beta$. Y-axis: classification accuracy:

fication on the test set: the proportion of correctly classified words out of a total of 250 (five words each in 50 trials).

All models perform well for balanced sense frequencies; e.g., for $\alpha = 1.0, \beta = 0.5$, the SVMs were all close to 100% accurate in predicting that the w_i can occur in a c-d context. PPMI accuracy falls steeply when α is increased from 1.4 to 1.5. It has a 100% error rate for $\alpha \geq 1.5$. Learning-based models perform better in the order CBOW (least robust), LBL, SSKIP, SKIP, CWIN (most robust). Even for $\alpha = 2.0$, CWIN and SKIP are still close to 100% accurate.

Discussion. The evaluation criterion we have used here is a classification task. The classifier attempts to answer a question that may occur in an application – can this word be used in this context? Thus, the evaluation criterion is: does the word representation contain a specific type of information that is needed for the application.

Another approach to ambiguity is to compute multiple representations for a word, one for each sense. We generally do not yet know what the sense of a word is when we want to use its word representation, so data-driven approaches like clustering have been used to create representations for different usage clusters of words that may capture some of its senses. For example, Reisinger and Mooney (2010) and Huang et al. (2012) cluster the contexts of each word and then learn a different representation for each cluster. The main motivation for this approach is the assumption that single-word distributional representations cannot represent all senses of a word well (Huang et al., 2012). However, Li and Jurafsky (2015) show that simply increasing the dimensionality of single-representation gets comparable results to using multiple-representation. Our results confirm that a single embedding can be robust against ambiguity, but also show the main challenge: skewness of sense distribution.

1	$P(NF_n\|S)$	=1/4	
2	$P(AF_a\|S)$	=1/4	
3	$P(NM_n\|S)$	=1/4	
4	$P(AM_f\|S)$	=1/4	
5	$P(n_i\|N)$	=1/5	$0 \leq i \leq 4$
6	$P(a_i\|A)$	=1/5	$0 \leq i \leq 4$
7	$P(x_i^{\text{nf}} U_i^{\text{nf}} \| F_n)$	=1/5	$0 \leq i \leq 4$
8	$P(f\|U_i^{\text{nf}})$	=1/2	
9	$P(\mu(U_i^{\text{nf}})\|U_i^{\text{nf}})$	=1/2	
10	$P(x_i^{\text{af}} U_i^{\text{af}} \| F_a)$	=1/5	$0 \leq i \leq 4$
11	$P(f\|U_i^{\text{af}})$	=1/2	
12	$P(\mu(U_i^{\text{af}})\|U_i^{\text{af}})$	=1/2	
13	$P(x_i^{\text{nm}} U_i^{\text{nm}} \| M_n)$	=1/5	$0 \leq i \leq 4$
14	$P(m\|U_i^{\text{nm}})$	=1/2	
15	$P(\mu(U_i^{\text{nm}})\|U_i^{\text{nm}})$	=1/2	
16	$P(x_i^{\text{am}} U_i^{\text{am}} \| M_f)$	=1/5	$0 \leq i \leq 4$
17	$P(m\|U_i^{\text{am}})$	=1/2	
18	$P(\mu(U_i^{\text{am}})\|U_i^{\text{am}})$	=1/2	

Figure 5: This grammar generates nouns ($x_i^{\text{n}\cdot}$) and adjectives ($x_i^{\text{a}\cdot}$) with masculine (x_i^{m}) and feminine (x_i^{f}) gender as well as paradigm features u_i. μ maps each U to one of $\{u_0 \dots u_4\}$. μ is randomly initialized and then kept fixed.

4.4 Accurate and consistent representation of multifacetedness

Grammar. The grammar shown in Figure 5 models two syntactic categories, nouns and adjectives, whose left context is highly predictable: it is one of five left context words n_i (resp. a_i) for nouns, see lines 1, 3, 5 (resp. for adjectives, see lines 2, 4, 6). There are two grammatical genders: feminine (corresponding to the two symbols F_n and F_a) and masculine (corresponding to the two symbols M_n and M_a). The four combinations of syntactic category and gender are equally probable (lines 1–4). In addition to *gender*, nouns and adjectives are distinguished with respect to *morphological paradigm*. Line 7 generates one of five feminine nouns (x_i^{nf}) and the corresponding paradigm marker U_i^{nf}. A noun has two equally probable right contexts: a context indicating its gender (line 8) and a context indicating its paradigm (line 9). μ is a function that maps each U to one of five morphological paradigms $\{u_0 \dots u_4\}$. μ is randomly initialized before a corpus is generated and kept fixed.

The function μ models the assignment of

paradigms to nouns and adjectives. Nouns and adjectives can have different (or the same) paradigms, but for a given noun or adjective the paradigm is fixed and does not change. Lines 7–9 generate gender and paradigm markers for feminine nouns, for which we use the symbols x_i^{nf}. Lines 10–18 cover the three other cases: masculine nouns (x_i^{nm}), feminine adjectives (x_i^{af}) and masculine adjectives (x_i^{am}).

Dataset. We perform 10 trials. In each trial, μ is initialized randomly and a corpus of 100,000 sentences is generated. The train set consists of the feminine nouns (x_i^{nf}, line 7) and the masculine nouns (x_i^{nm}, line 13). The test set consists of the feminine (x_i^{af}) and masculine (x_i^{am}) adjectives.

Results. Embeddings have been learned, SVMs are trained on the binary classification task feminine vs. masculine and evaluated on test. There was not a single error: accuracy of classifications is 100% for all embedding models.

Discussion. The facet gender is indicated directly by the distribution and easy to learn. For a noun or adjective x, we simply have to check whether f or m occurs to its right anywhere in the corpus. PPMI stores this information in two dimensions of the vectors and the SVM learns this fact perfectly. The encoding of "f or m occurs to the right" is less direct in the learning-based representation of x, but the experiment demonstrates that they also reliably encode it and the SVM reliably picks it up.

It would be possible to encode the facet in just one bit in a manually designed representation. While all representations are less compact than a one-bit representation – PPMI uses two real dimensions, learning-based models use an activation pattern over several dimensions – it is still true that most of the capacity of the embeddings is used for encoding facets other than gender: syntactic categories and paradigms. Note that there are five different instances each of feminine/masculine adjectives, feminine/masculine nouns and u_i words, but only two gender indicators: f and m. This is a typical scenario across languages: words are distinguished on a large number of morphological, grammatical, semantic and other dimensions and each of these dimensions corresponds to a small fraction of the overall knowledge we have about a given word.

Point-based tests do not directly evaluate specific facets of words. In similarity datasets, there is no individual test on facets – only full-space similarity is considered. There are test cases in analogy that hypothetically evaluate specific facets like gender of words, as in king-man+woman=queen. However, it does not consider the impact of other facets and assumes the only difference of "king" and "queen" is gender. A clear example that words usually differ on many facets, not just one, is the analogy: London:England $\sim$ Ankara:Turkey. *political-capital-of* applies to both, *cultural-capital-of* only to London:England since Istanbul is the cultural capital of Turkey.

To make our argument more clear, we designed an additional experiment that tries to evaluate GEN in our dataset based on similarity and analogy methods. In the *similarity evaluation*, we search for the nearest neighbor of each word and accuracy is the proportion of nearest neighbors that have the same gender as the search word. In the *analogy evaluation*, we randomly select triples of the form $<x_i^{c_1 g_1}, x_j^{c_1 g_2}, x_k^{c_2 g_2}>$ where $(c_1, c_2) \in \{(\text{noun}, \text{adjective}), (\text{adjective}, \text{noun})\}$ and $(g_1, g_2) \in \{(\text{masculine}, \text{feminine}), (\text{feminine}, \text{masculine})\}$. We then compute $\vec{s} = \vec{x}_i^{c_1 g_1} - \vec{x}_j^{c_1 g_2} + \vec{x}_k^{c_2 g_2}$ and identify the word whose vector is closest to $\vec{s}$ where the three vectors $\vec{x}_i^{c_1 g_1}$, $\vec{x}_j^{c_1 g_2}$, $\vec{x}_k^{c_2 g_2}$ are excluded. If the nearest neighbor of $\vec{s}$ is of type $\vec{x}_l^{c_2 g_1}$, then the search is successful; e.g., for $\vec{s} = \vec{x}_i^{\text{nf}} - \vec{x}_j^{\text{nm}} + \vec{x}_k^{\text{am}}$, the search is successful if the nearest neighbor is feminine. We did this evaluation on the same test set for PPMI and LBL embedding models. Error rates were 29% for PPMI and 25% for LBL (similarity) and 16% for PPMI and 14% for LBL (analogy). This high error, compared to 0% error for SVM classification, indicates it is not possible to determine the presence of a low entropy facet accurately and consistently when full-space similarity and analogy are used as test criteria.

5 Analysis

In this section, we first summarize and analyze the lessons we learned through experiments in Section 4. After that, we show how these lessons are supported by a real natural-language corpus.

5.1 Learned lessons

(i) Two words with clearly different context distributions should receive different representations. Aggregation models fail to do so by calculating

	all entities		head entities		tail entities	
	MLP	1NN	MLP	1NN	MLP	1NN
PPMI	61.6	44.0	69.2	63.8	43.0	28.5
LBL	63.5	51.7	72.7	66.4	44.1	32.8
CBOW	63.0	53.5	71.7	69.4	39.1	29.9
CWIN	66.1	53.0	73.5	68.6	46.8	31.4
SKIP	64.5	**57.1**	69.9	**71.5**	**49.8**	**34.0**
SSKIP	**66.2**	52.8	**73.9**	68.5	45.5	31.4

Table 1: Entity typing results using embeddings learned with different models.

global statistics.

(ii) Embedding learning can have different effectiveness for sparse vs. non-sparse events. Thus, models of representations should be evaluated with respect to their ability to deal with sparseness; evaluation data sets should include rare as well as frequent words.

(iii) Our results in Section 4.3 suggest that single-representation approaches can indeed represent different senses of a word. We did a classification task that roughly corresponds to the question: does this word have a particular meaning? A representation can fail on similarity judgement computations because less frequent senses occupy a small part of the capacity of the representation and therefore have little impact on full-space similarity values. Such a failure does not necessarily mean that a particular sense is not present in the representation and it does not necessarily mean that single-representation approaches perform poor on real-world tasks. However, we saw that even though single-representations do well on balanced senses, they can pose a challenge for ambiguous words with skewed senses.

(iv) Lexical information is complex and multifaceted. In point-based tests, all dimensions are considered together and their ability to evaluate specific facets or properties of a word is limited. The full-space similarity of a word may be highest to a word that has a different value on a low-entropy facet. Any good or bad result on these tasks is not sufficient to conclude that the representation is weak. The valid criterion of quality is whether information about the facet is consistently and accurately stored.

5.2 Extrinsic evaluation: entity typing

To support the case for sub-space evaluation and also to introduce a new extrinsic task that uses the embeddings directly in supervised classification, we address a *fine-grained entity typing* task.

Learning taxonomic properties or types of words has been used as an evaluation method for word embeddings (Rubinstein et al., 2015). Since available word typing datasets are quite small (cf. Baroni et al. (2014), Rubinstein et al. (2015)), entity typing can be a promising alternative, which enables to do supervised classification instead of unsupervised clustering. Entities, like other words, have many properties and therefore belong to several semantic types, e.g., "Barack Obama" is a POLITICIAN, AUTHOR and AWARD_WINNER. We perform entity typing by learning types of knowledge base entities from their embeddings; this requires looking at subspaces because each entity can belong to multiple types.

We adopt the setup of Yaghoobzadeh and Schütze (2015) who present a dataset of Freebase entities;[5] there are 102 types (e.g., POLITICIAN FOOD, LOCATION-CEMETERY) and most entities have several. More specifically, we use a multilayer-perceptron (MLP) with one hidden layer to classify entity embeddings to 102 FIGER types. To show the limit of point-based evaluation, we also experimentally test an entity typing model based on *cosine similarity* of entity embeddings. To each test entity, we assign all types of the entity closest to it in the train set. We call this approach 1NN (kNN for $k = 1$).[6]

We take part of ClueWeb, which is annotated with Freebase entities using automatic annotation of FACC1[7] (Gabrilovich et al., 2013), as our corpus. We then replace all mentions of entities with their Freebase identifier and learn embeddings of words and entities in the same space. Our corpus has around 6 million sentences with at least one annotated entity. We calculate embeddings using our different models. Our hyperparameters: for learning-based models: dim=100, neg=10, iterations=20, window=1, sub=10^{-3}; for PPMI: SVD-dim=100, neg=1, window=1, cds=0.75, sub=10^{-3}, eig=0.5. See (Levy et al., 2015) for more information about the meaning of hyperparameters.

Table 1 gives results on test for all (about 60,000 entities), head (freq $>$ 100; about 12,200 entities) and tail (freq $<$ 5; about 10,000 entities). The MLP models consistently outperform 1NN on

[5] cistern.cis.lmu.de/figment

[6] We tried other values of k, but results were not better.

[7] lemurproject.org/clueweb12/FACC1

all and tail entities. This supports our hypothesis that only part of the information about types that is present in the vectors can be determined by similarity-based methods that use the overall direction of vectors, i.e., full-space similarity.

There is little correlation between results of MLP and 1NN in all and head entities, and the correlation between their results in tail entities is high.[8] For example, for all entities, using 1NN, SKIP is 4.3% (4.1%) better, and using MLP is 1.7% (1.6%) worse than SSKIP (CWIN). The good performance of SKIP on 1NN using cosine similarity can be related to its objective function, which maximizes the cosine similarity of cooccuring token embeddings.

The important question is not similarity, but whether the information about a specific type exists in the entity embeddings or not. Our results confirm our previous observation that a classification by looking at subspaces is needed to answer this question. In contrast, based on full-space similarity, one can infer little about the quality of embeddings. Based on our results, SSKIP and CWIN embeddings contain more accurate and consistent information because MLP classifier gives better results for them. However, if we considered 1NN for comparison, SKIP and CBOW would be superior.

6 Conclusion and future work

We have introduced a new way of evaluating distributional representation models. As an alternative to the common evaluation tasks, we proposed to identify generic criteria that are important for an embedding model to represent properties of words accurately and consistently. We suggested four criteria based on fundamental characteristics of natural language and designed tests that evaluate models on the criteria. We developed this evaluation methodology using PCFG-generated corpora and applied it on a case study to compare different models of learning distributional representations.

While we showed important differences of the embedding models, the goal was not to do a comprehensive comparison of them. We proposed an innovative way of doing intrinsic evaluation of embeddings. Our evaluation method gave direct insight about the quality of embeddings. Additionally, while most intrinsic evaluations consider word vectors as points, we used classifiers that identify different small subspaces of the full space. This is an important desideratum when designing evaluation methods because of the multifacetedness of natural language words: they have a large number of properties, each of which only occupies a small proportion of the full-space capacity of the embedding.

Based on this paper, there are serveral lines of investigation we plan to conduct in the future. (i) We will attempt to support our results on artificially generated corpora by conducting experiments on *real natural language data*. (ii) We will study the *coverage of our four criteria* in evaluating word representations. (iii) We modeled the four criteria using separate PCFGs, but they could also be modeled by one single unified PCFG. One question that arises is then to what extent the four criteria are orthogonal and to what extent interdependent. A single unified grammar may make it harder to interpret the results, but may give additional and more fine-grained insights as to how the performance of embedding models is influenced by different fundamental properties of natural language and their interactions.

Finally, we have made the simplifying assumption in this paper that the best conceptual framework for thinking about embeddings is that the embedding space can be *decomposed into subspaces*: either into completely orthogonal subspaces or – less radically – into partially "overlapping" subspaces. Furthermore, we have made the assumption that the smoothness and robustness properties that are the main reasons why embeddings are used in NLP can be reduced to *similarities in subspaces*. See Rothe et al. (2016) and Rothe and Schütze (2016) for work that makes similar assumptions.

The fundamental assumptions here are decomposability and linearity. The smoothness properties could be much more complicated. However even if this was the case, then much of the general framework of what we have presented in this paper would still apply; e.g., the criterion that a particular facet be fully and correctly represented is as important as before. But the validity of the assumption that embedding spaces can be decomposed into "linear" subspaces should be investigated in the future.

Acknowledgments. This work was supported by DFG (SCHU 2246/8-2).

[8] The spearman correlation between MLP and 1NN for all=0.31, head=0.03, tail=0.75.

References

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA*, pages 19–27.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 238–247.

Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. Blackwell, 2nd edition.

D. A. Cruse. 1986. *Lexical Semantics*. Cambridge University Press, Cambridge, MA.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 873–882.

Arne Köhn. 2015. What?s in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, Lisbon, Portugal, September.

Siwei Lai, Kang Liu, Liheng Xu, and Jun Zhao. 2015. How to generate a good word embedding? *CoRR*, abs/1507.05523.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *CoNLL*.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal, September.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1299–1304.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*, pages 2265–2273.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *ACL*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense embeddings by orthogonal transformation. In *NAACL*.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 726–730.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, Lisbon, Portugal, September.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, Lisbon, Portugal, September.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res. (JAIR)*, 37:141–188.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725, Lisbon, Portugal, September.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*, pages 78–83.

Chapter 4

Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities

Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities

Yadollah Yaghoobzadeh and **Hinrich Schütze**
Center for Information and Language Processing
LMU Munich, Germany
yadollah@cis.lmu.de

Abstract

Entities are essential elements of natural language. Learning rich representations of entities is therefore important. In this paper, we present methods for learning multi-level representations of entities on three complementary levels: **character** (character patterns in entity names extracted, e.g., by neural networks), **word** (embeddings of words in entity names) and **entity** (entity embeddings). We investigate state-of-the-art learning methods on each level and find large differences, e.g., for deep learning models, traditional ngram features and the subword model of `fasttext` (Bojanowski et al., 2016) on the **character** level; for `word2vec` (Mikolov et al., 2013) on the **word** level; and for the order-aware model `wang2vec` (Ling et al., 2015a) on the **entity** level.

We confirm experimentally that each level of representation contributes complementary information and a joint representation of all three levels improves the existing embedding based baseline for fine-grained entity typing by a large margin. Additionally, we show that adding information from entity descriptions further improves multi-level representations of entities.

1 Introduction

Knowledge about entities is essential for understanding human language. This knowledge can be attributional (e.g., canFly, isEdible), type-based (e.g., isFood, isPolitician, isDisease) or relational (e.g, marriedTo, bornIn). Knowledge bases (KBs) are designed to store this information in a structured way, so that it can be queried easily. Examples of such KBs are Freebase (Bollacker et al., 2008), Wikipedia, Google knowledge graph and YAGO (Suchanek et al., 2007). For automatic updating and completing the entity knowledge, text resources such as news, user forums, textbooks or any other data in the form of text are important sources. Therefore, information extraction methods have been introduced to extract knowledge about entities from text. In this paper, we focus on the extraction of entity types, i.e., assigning types to – or *typing* – entities. Type information can help extraction of relations by applying constraints on relation arguments.

We address a problem setting in which the following are given: a KB with a set of entities E, a set of types T and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t; and a large corpus C in which mentions of E are annotated. In this setting, we address the task of *fine-grained entity typing*: we want to learn a probability function $S(e, t)$ for a pair of entity e and type t and based on $S(e, t)$ infer whether $m(e, t) = 1$ holds, i.e., whether entity e is a member of type t.

We address this problem by learning a multi-level representation for an entity that contains the information necessary for typing it. One important source is the *contexts in which the entity is used*. We can take the standard method of learning embeddings for words and extend it to learning embeddings for entities. This requires the use of an entity linker and can be implemented by replacing all occurrences of the entity by a unique token. We refer to entity embeddings as *entity-level representations*. Previously, entity embeddings have been learned mostly using bag-of-word models like `word2vec` (e.g., by Wang et al. (2014) and Yaghoobzadeh and Schütze (2015)). We show below that order information is critical for high-quality entity embeddings.

Entity-level representations are often uninfor-

mative for rare entities, so that using only entity embeddings is likely to produce poor results. In this paper, we use *entity names* as a source of information that is complementary to entity embeddings. We define an entity name as a noun phrase that is used to refer to an entity. We learn character and word level representations of entity names.

For the *character-level representation*, we adopt different character-level neural network architectures. Our intuition is that there is sub/cross word information, e.g., orthographic patterns, that is helpful to get better entity representations, especially for rare entities. A simple example is that a three-token sequence containing an initial like "P." surrounded by two capitalized words ("Rolph P. Kugl") is likely to refer to a person.

We compute the *word-level representation* as the sum of the embeddings of the words that make up the entity name. The sum of the embeddings accumulates evidence for a type/property over all constituents, e.g., a name containing "stadium", "lake" or "cemetery" is likely to refer to a location. In this paper, we compute our word level representation with two types of word embeddings: (i) using only contextual information of words in the corpus, e.g., by `word2vec` (Mikolov et al., 2013) and (ii) using subword as well as contextual information of words, e.g., by Facebook's recently released `fasttext` (Bojanowski et al., 2016).

In this paper, we integrate character-level and word-level with entity-level representations to improve the results of previous work on fine-grained typing of KB entities. We also show how descriptions of entities in a KB can be a complementary source of information to our multi-level representation to improve the results of entity typing, especially for rare entities.

Our main contributions in this paper are:

- We propose new methods for learning entity representations on three levels: character-level, word-level and entity-level.
- We show that these levels are complementary and a joint model that uses all three levels improves the state of the art on the task of fine-grained entity typing by a large margin.
- We experimentally show that an order dependent embedding is more informative than its bag-of-word counterpart for entity representation.

We release our dataset and source codes: `cistern.cis.lmu.de/figment2/`.

2 Related Work

Entity representation. Two main sources of information used for learning entity representation are: (i) links and descriptions in KB, (ii) name and contexts in corpora. We focus on name and contexts in corpora, but we also include (Wikipedia) descriptions. We represent entities on three levels: entity, word and character. Our entity-level representation is similar to work on relation extraction (Wang et al., 2014; Wang and Li, 2016), entity linking (Yamada et al., 2016; Fang et al., 2016), and entity typing (Yaghoobzadeh and Schütze, 2015). Our word-level representation with distributional word embeddings is similarly used to represent entities for entity linking (Sun et al., 2015) and relation extraction (Socher et al., 2013; Wang et al., 2014). Novel entity representation methods we introduce in this paper are representation based on `fasttext` (Bojanowski et al., 2016) subword embeddings, several character-level representations, "order-aware" entity-level embeddings and the combination of several different representations into one multi-level representation.

Character-subword level neural networks. Character-level convolutional neural networks (CNNs) are applied by dos Santos and Zadrozny (2014) to part of speech (POS) tagging, by dos Santos and Guimarães (2015), Ma and Hovy (2016), and Chiu and Nichols (2016) to named entity recognition (NER), by Zhang et al. (2015) and Zhang and LeCun (2015) to sentiment analysis and text categorization, and by Kim et al. (2016) to language modeling (LM). Character-level LSTM is applied by Ling et al. (2015b) to LM and POS tagging, by Lample et al. (2016) to NER, by Ballesteros et al. (2015) to parsing morphologically rich languages, and by Cao and Rei (2016) to learning word embeddings. Bojanowski et al. (2016) learn word embeddings by representing words with the average of their character ngrams (subwords) embeddings. Similarly, Chen et al. (2015) extends `word2vec` for Chinese with joint modeling with characters.

Fine-grained entity typing. Our task is to infer fine-grained types of KB entities. KB completion is an application of this task. Yaghoobzadeh and Schütze (2015)'s FIGMENT system addresses this task with only contextual information; they

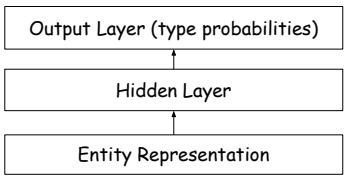

Figure 1: Schematic diagram of our architecture for entity classification. "Entity Representation" ($\vec{v}(e)$) is the (one-level or multi-level) vector representation of entity. Size of output layer is $|T|$.

do not use character-level and word-level features of entity names. Neelakantan and Chang (2015) and Xie et al. (2016) also address a similar task, but they rely on entity descriptions in KBs, which in many settings are not available. The problem of Fine-grained mention typing (FGMT) (Yosef et al., 2012; Ling and Weld, 2012; Yogatama et al., 2015; Del Corro et al., 2015; Shimaoka et al., 2016; Ren et al., 2016) is related to our task. FGMT classifies single *mentions* of named entities to their context dependent types whereas we attempt to identify all types of a KB *entity* from the aggregation of all its mentions. FGMT can still be evaluated in our task by aggregating the mention level decisions but as we will show in our experiments for one system, i.e., FIGER (Ling and Weld, 2012), our entity embedding based models are better in entity typing.

3 Fine-grained entity typing

Given (i) a KB with a set of entities E, (ii) a set of types T, and (iii) a large corpus C in which mentions of E are linked, we address the task of *fine-grained entity typing* (Yaghoobzadeh and Schütze, 2015): predict whether entity e is a member of type t or not. To do so, we use a set of training examples to learn $P(t|e)$: the probability that entity e has type t. These probabilities can be used to assign *new types* to entities covered in the KB as well as typing *unknown entities*.

We learn $P(t|e)$ with a general architecture; see Figure 1. The output layer has size $|T|$. Unit t of this layer outputs the probability for type t. "Entity Representation" ($\vec{v}(e)$) is the vector representation of entity e – we will describe in detail in the rest of this section what forms $\vec{v}(e)$ takes. We model $P(t|e)$ as a multi-label classification, and train a multilayer perceptron (MLP) with one hidden layer:

$$\big[P(t_1|e)\dots P(t_T|e)\big] = \sigma\Big(\mathbf{W}_{\text{out}} f\left(\mathbf{W}_{\text{in}}\vec{v}(e)\right)\Big) \tag{1}$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{h\times d}$ is the weight matrix from $\vec{v}(e) \in \mathbb{R}^d$ to the hidden layer with size h. f is the rectifier function. $\mathbf{W}_{\text{out}} \in \mathbb{R}^{|T|\times h}$ is the weight matrix from hidden layer to output layer of size $|T|$. σ is the sigmoid function. Our objective is binary cross entropy summed over types:

$$\sum_t -\Big(m_t \log p_t + (1-m_t)\log\left(1-p_t\right)\Big)$$

where m_t is the truth and p_t the prediction.

The key difficulty when trying to compute $P(t|e)$ is in learning a good representation for entity e. We make use of contexts and name of e to represent its feature vector on the three levels of entity, word and character.

3.1 Entity-level representation

Distributional representations or embeddings are commonly used for words. The underlying hypothesis is that words with similar meanings tend to occur in similar contexts (Harris, 1954) and therefore cooccur with similar context words. We can extend the distributional hypothesis to entities (cf. Wang et al. (2014), Yaghoobzadeh and Schütze (2015)): entities with similar meanings tend to have similar contexts. Thus, we can learn a d dimensional embedding $\vec{v}(e)$ of entity e from a corpus in which all mentions of the entity have been replaced by a special identifier. We refer to these entity vectors as the *entity level representation* (ELR).

In previous work, order information of context words (relative position of words in the contexts) was generally ignored and objectives similar to the SkipGram (henceforth: *SKIP*) model were used to learn $\vec{v}(e)$. However, the bag-of-word context is difficult to distinguish for pairs of types like (restaurant,food) and (author,book). This suggests that *using order aware embedding models is important for entities*. Therefore, we apply Ling et al. (2015a)'s extended version of SKIP, Structured SKIP (SSKIP). It incorporates the order of context words into the objective. We compare it with SKIP embeddings in our experiments.

3.2 Word-level representation

Words inside entity names are important sources of information for typing entities. We define the

word-level representation (WLR) as the *average of the embeddings of the words* that the entity name contains $\vec{v}(e) = 1/n \sum_{i=1}^{n} \vec{v}(w_i)$ where $\vec{v}(w_i)$ is the embedding of the i^{th} word of an entity name of length n. We opt for simple averaging since entity names often consist of a small number of words with clear semantics. Thus, averaging is a promising way of combining the information that each word contributes.

The word embedding, $\vec{w}$, itself can be learned from models with different granularity levels. Embedding models that consider words as atomic units in the corpus, e.g., SKIP and SSKIP, are word-level. On the other hand, embedding models that represent words with their character ngrams, e.g., `fasttext` (Bojanowski et al., 2016), are subword-level. Based on this, we consider and evaluate **word-level WLR (WWLR)** and **subword-level WLR (SWLR)** in this paper.[1]

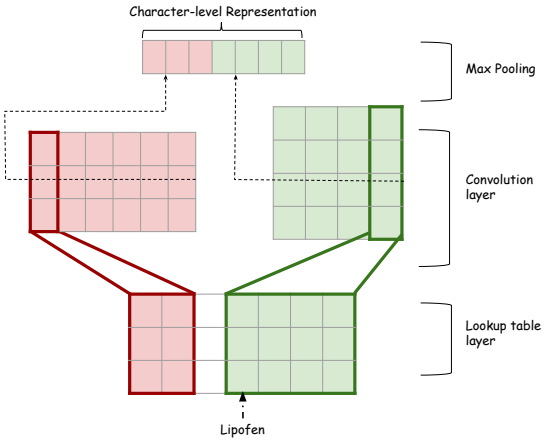

Figure 2: Example architecture for the character-level CNN with max pooling. The input is "Lipofen". Character embedding size is three. There are three filters of width 2 and four filters of width 4.

3.3 Character-level representation

For computing the *character level representation* (CLR), we design models that try to type an entity based on the sequence of characters of its name. Our hypothesis is that names of entities of a specific type often have similar character patterns. Entities of type ETHNICITY often end in "ish" and "ian", e.g., "Spanish" and "Russian". Entities of type MEDICINE often end in "en": "Lipofen", "acetaminophen". Also, some types tend to have specific cross-word shapes in their entities, e.g., PERSON names usually consist of two words, or MUSIC names are usually long, containing several words.

The first layer of the character-level models is a *lookup table* that maps each character to an embedding of size d_c. These embeddings capture similarities between characters, e.g., similarity in type of phoneme encoded (consonant/vowel) or similarity in case (lower/upper). The output of the lookup layer for an entity name is a matrix $C \in \mathbb{R}^{l \times d_c}$ where l is the maximum length of a name and all names are padded to length l. This length l includes special start/end characters that bracket the entity name.

We experiment with four architectures to produce character-level representations in this paper: FORWARD (direct forwarding of character embeddings), CNNs, LSTMs and BiLSTMs. The output of each architecture then takes the place of the entity representation $\vec{v}(e)$ in Figure 1.

FORWARD simply concatenates all rows of matrix C; thus, $\vec{v}(e) \in \mathbb{R}^{d_c * l}$.

The **CNN** uses k filters of different window widths w to narrowly convolve C. For each filter $H \in \mathbb{R}^{d_c \times w}$, the result of the convolution of H over matrix C is feature map $f \in \mathbb{R}^{l-w+1}$:

$$f[i] = \text{rectifier}(C_{[:,i:i+w-1]} \odot H + b)$$

where rectifier is the activation function, b is the bias, $C_{[:,i:i+w-1]}$ are the columns i to $i+w-1$ of C, $1 \leq w \leq 10$ are the window widths we consider and $\odot$ is the sum of element-wise multiplication. Max pooling then gives us one feature for each filter. The concatenation of all these features is our representation: $\vec{v}(e) \in \mathbb{R}^k$. An example CNN architecture is show in Figure 2.

The input to the **LSTM** is the character sequence in matrix C, i.e., $x_1, \ldots, x_l \in \mathbb{R}^{d_c}$. It generates the state sequence $h_1, ..., h_{l+1}$ and the output is the last state $\vec{v}(e) \in \mathbb{R}^{d_h}$.[2]

The **BiLSTM** consists of two LSTMs, one going forward, one going backward. The first state of the backward LSTM is initialized as h_{l+1}, the last state of the forward LSTM. The BiLSTM entity representation is the concatenation of last states of forward and backward LSTMs, i.e., $\vec{v}(e) \in \mathbb{R}^{2*d_h}$.

[1] Subword models have properties of both character-level models (subwords are character ngrams) and of word-level models (they do not cross boundaries between words). They probably could be put in either category, but in our context fit the word-level category better because we see the granularity level with respect to the entities and not words.

[2] We use Blocks (van Merriënboer et al., 2015).

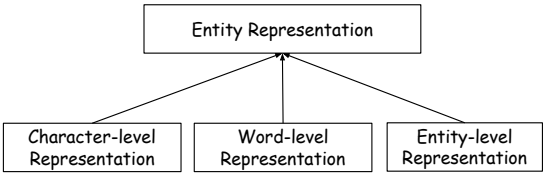

Figure 3: Multi-level representation

3.4 Multi-level representations

Our different levels of representations can give complementary information about entities.

WLR and CLR. Both WLR models, SWLR and WWLR, do not have access to the cross-word character ngrams of entity names while CLR models do. Also, CLR is task specific by training on the entity typing dataset while WLR is generic. On the other hand, WWLR and SWLR models have access to information that CLR ignores: the tokenization of entity names into words and embeddings of these words. It is clear that words are particularly important character sequences since they often correspond to linguistic units with clearly identifiable semantics – which is not true for most character sequences. For many entities, the words they contain are a better basis for typing than the character sequence. For example, even if "nectarine" and "compote" did not occur in any names in the training corpus, we can still learn good word embeddings from their non-entity occurrences. This then allows us to correctly type the entity "Aunt Mary's Nectarine Compote" as FOOD based on the sum of the word embeddings.

WLR/CLR and ELR. Representations from entity names, i.e., WLR and CLR, by themselves are limited because many classes of names can be used for different types of entities; e.g., person names do not contain hints as to whether they are referring to a politician or athlete. In contrast, the ELR embedding is based on an entity's contexts, which are often informative for each entity and can distinguish politicians from athletes. On the other hand, not all entities have sufficiently many informative contexts in the corpus. For these entities, their name can be a complementary source of information and character/word level representations can increase typing accuracy.

Thus, we introduce joint models that use combinations of the three levels. Each multi-level model concatenates several levels. We train the constituent embeddings as follows. WLR and ELR are computed as described above and are not changed during training. CLR – produced by one of the character-level networks described above – is initialized randomly and then tuned during training. Thus, it can focus on complementary information related to the task that is not already present in other levels. The schematic diagram of our multi-level representation is shown in Figure 3.

4 Experimental setup and results

4.1 Setup

Entity datasets and corpus. We address the task of fine-grained entity typing and use Yaghoobzadeh and Schütze (2015)'s FIGMENT dataset[3] for evaluation. The FIGMENT corpus is part of a version of ClueWeb in which Freebase entities are annotated using FACC1 (URL, 2016b; Gabrilovich et al., 2013). The FIGMENT entity datasets contain 200,000 Freebase entities that were mapped to 102 FIGER types (Ling and Weld, 2012). We use the same train (50%), dev (20%) and test (30%) partitions as Yaghoobzadeh and Schütze (2015) and extract the names from mentions of dataset entities in the corpus. We take the most frequent name for dev and test entities and three most frequent names for train (each one tagged with entity types).

Adding parent types to refine entity dataset. FIGMENT ignores that FIGER is a proper hierarchy of types; e.g., while HOSPITAL is a subtype of BUILDING according to FIGER, there are entities in FIGMENT that are hospitals, but not buildings.[4] Therefore, we modified the FIGMENT dataset by adding for each assigned type (e.g., HOSPITAL) its parents (e.g., BUILDING). This makes FIGMENT more consistent and eliminates spurious false negatives (BUILDING in the example).

We now describe our **baselines**: (i) BOW & NSL: hand-crafted features, (ii) FIGMENT (Yaghoobzadeh and Schütze, 2015) and (iii) adapted version of FIGER (Ling and Weld, 2012).

We implement the following two feature sets from the literature as a *hand-crafted baseline* for our character and word level models. (i) *BOW*: individual words of entity name (both as-is and lowercased); (ii) *NSL* (ngram-shape-length): shape and length of the entity name (cf. Ling and Weld (2012)), character n-grams, $1 \leq n \leq n_{\max}, n_{\max} = 5$ (we also tried $n_{\max} = 7$, but results were worse

[3] cistern.cis.lmu.de/figment/

[4] See github.com/xiaoling/figer for FIGER

on dev) and normalized character n-grams: lowercased, digits replaced by "7", punctuation replaced by ".". These features are represented as a sparse binary vector $\vec{v}(e)$ that is input to the architecture in Figure 1.

FIGMENT is the model for entity typing presented by Yaghoobzadeh and Schütze (2015). The authors only use entity-level representations for entities trained by SkipGram, so the FIGMENT baseline corresponds to the entity-level result shown as ELR(SKIP) in the tables.

The third baseline is using an existing mention-level entity typing system, *FIGER* (Ling and Weld, 2012). FIGER uses a wide variety of features on different levels (including parsing-based features) from contexts of entity mentions as well as the mentions themselves and returns a score for each mention-type instance in the corpus. We provide the ClueWeb/FACC1 segmentation of entities, so FIGER does not need to recognize entities.[5] We use the trained model provided by the authors and normalize FIGER scores using softmax to make them comparable for aggregation. We experimented with different aggregation functions (including maximum and k-largest-scores for a type), but we use the average of scores since it gave us the best result on dev. We call this baseline AGG-FIGER.

Distributional embeddings. For WWLR and ELR, we use SkipGram model in `word2vec` and SSkip model in `wang2vec` (Ling et al., 2015a) to learn embeddings for words, entities and types. To obtain embeddings for all three in the same space, we process ClueWeb/FACC1 as follows. For each sentence s, we add three copies: s itself, a copy of s in which each entity is replaced with its Freebase identifier (MID) and a copy in which each entity (not test entities though) is replaced with an ID indicating its notable type. The resulting corpus contains around 4 billion tokens and 1.5 billion types.

We run SKIP and SSkip with the same setup (200 dimensions, 10 negative samples, window size 5, word frequency threshold of 100)[6] on this corpus to learn embeddings for words, entities and FIGER types. Having entities and types in the same vector space, we can add another feature vector $\vec{v}(e) \in \mathbb{R}^{|T|}$ (referred to as TC below): for each entity, we compute cosine similarity of its entity vector with all type vectors.

For SWLR, we use `fasttext`[7] to learn word embeddings from the ClueWeb/FACC1 corpus. We use similar settings as our WWLR SKIP and SSkip embeddings and keep the defaults of other hyperparameters. Since the trained model of `fasttext` is applicable for new words, we apply the model to get embeddings for the filtered rare words as well.

model	hyperparameters
CLR(FF)	$d_c = 15, h_{mlp} = 600$
CLR(LSTM)	$d_c = 70, d_h = 70, h_{mlp} = 300$
CLR(BiLSTM)	$d_c = 50, d_h = 50, h_{mlp} = 200$
CLR(CNN)	$d_c = 10, w = [1, .., 8]$ $n = 100, h_{mlp} = 800$
CLR(NSL)	$h_{mlp} = 800$
BOW	$h_{mlp} = 200$
BOW+CLR(NSL)	$h_{mlp} = 300$
WWLR	$h_{mlp} = 400$
SWLR	$h_{mlp} = 400$
WWLR+CLR(CNN)	$w = [1, ..., 7]$ $d_c = 10, n = 50, h_{mlp} = 700$
SWLR+CLR(CNN)	$w = [1, ..., 7]$ $d_c = 10, n = 50, h_{mlp} = 700$
ELR(SKIP)	$h_{mlp} = 400$
ELR(SSKIP)	$h_{mlp} = 400$
ELR+CLR	$d_c = 10, w = [1, ..., 7]$ $n = 100, h_{mlp} = 700$
ELR+WWLR	$h_{mlp} = 600$
ELR+SWLR	$h_{mlp} = 600$
ELR+WWLR+CLR	$d_c = 10, w = [1, ..., 7]$ $n = 50, h_{mlp} = 700$
ELR+SWLR+CLR	$d_c = 10, w = [1, ..., 7]$ $n = 50, h_{mlp} = 700$
ELR+WWLR+CNN+TC	$d_c = 10, w = [1, ..., 7]$ $n = 50, h_{mlp} = 900$
ELR+SWLR+CNN+TC(MuLR)	$d_c = 10, w = [1, ..., 7]$ $n = 50, h_{mlp} = 900$
AVG-DES	$h_{mlp} = 400$
MuLR+AVG-DES	$d_c = 10, w = [1, ..., 7]$ $n = 50, h_{mlp} = 1000$

Table 1: Hyperparameters of different models. w is the filter size. n is the number of feature maps for each filter size. d_c is the character embedding size. d_h is the LSTM hidden state size. h_{mlp} is the number of hidden units in the MLP.

Our **hyperparameter values** are given in Table 1. The values are optimized on dev. We use AdaGrad and minibatch training. For each experiment, we select the best model on dev.

We use these **evaluation measures**: (i) accuracy: an entity is correct if all its types and no incorrect types are assigned to it; (ii) micro average F_1: F_1 of all type-entity assignment decisions; (iii) entity macro average F_1: F_1 of types assigned to an entity, averaged over entities; (iv) type macro average F_1: F_1 of entities assigned to a type, averaged over types.

The assignment decision is based on thresh-

[5] Mention typing is separated from recognition in FIGER model. So it can use our segmentation of entities.

[6] The threshold does not apply for MIDs.

[7] `github.com/facebookresearch/fastText`

olding the probability function $P(t|e)$. For each model and type, we select the threshold that maximizes F_1 of entities assigned to the type on dev.

4.2 Results

Table 2 gives results on the test entities for all (about 60,000 entities), head (frequency $>$ 100; about 12,200) and tail (frequency $<$ 5; about 10,000). *MFT* (line 1) is the most frequent type baseline that ranks types according to their frequency in the train entities. Each level of representation is separated with dashed lines, and – unless noted otherwise – the best of each level is joined in multi level representations.[8]

Character-level models are on lines 2-6. The order of systems is: CNN $>$ NSL $>$ BiLSTM $>$ LSTM $>$ FORWARD. The results show that complex neural networks are more effective than simple forwarding. BiLSTM works better than LSTM, confirming other related work. CNNs probably work better than LSTMs because there are few complex non-local dependencies in the sequence, but many important local features. CNNs with maxpooling can more straightforwardly capture local and position-independent features. CNN also beats NSL baseline; a possible reason is that CNN – an automatic method of feature learning – is more robust than hand engineered feature based NSL. We show more detailed results in Section 4.3.

Word-level models are on lines 7-10. BOW performs worse than WWLR because it cannot deal well with sparseness. SSKIP uses word order information in WWLR and performs better than SKIP. SWLR uses subword information and performs better than WWLR, especially for tail entities. Integrating subword information improves the quality of embeddings for rare words and mitigates the problem of unknown words.

Joint word-character level models are on lines 11-13. WWLR+CLR(CNN) and SWLR+CLR(CNN) beat the component models. This confirms our underlying assumption in designing the complementary multi-level models. BOW problem with rare words does not allow its joint model with NSL to work better than NSL. WWLR+CLR(CNN) works better than BOW+CLR(NSL) by 10% micro F_1, again due to the limits of BOW compared to WWLR. Interestingly WWLR+CLR works better than SWLR+CLR and this suggests that WWLR is indeed richer than SWLR when CLR mitigates its problem with rare/unknown words

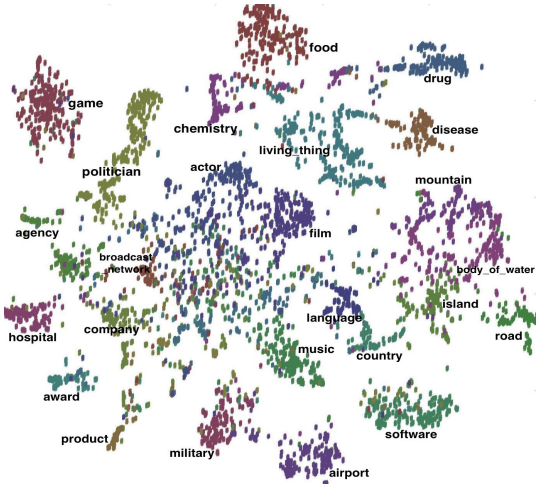

Figure 4: t-SNE result of entity-level representations

Entity-level models are on lines 14–15 and they are better than all previous models on lines 1–13. This shows the power of entity-level embeddings. In Figure 4, a t-SNE (Van der Maaten and Hinton, 2008) visualization of ELR(SKIP) embeddings using different colors for entity types shows that entities of the same type are clustered together. SSKIP works marginally better than SKIP for ELR, especially for tail entities, confirming our hypothesis that order information is important for a good distributional entity representation. This is also confirming the results of Yaghoobzadeh and Schütze (2016), where they also get better entity typing results with SSKIP compared to SKIP. They propose to use entity typing as an extrinsic evaluation for embedding models.

Joint entity, word, and character level models are on lines 16-23. The AGG-FIGER baseline works better than the systems on lines 1-13, but worse than ELRs. This is probably due to the fact that AGG-FIGER is optimized for mention typing and it is trained using distant supervision assumption. Parallel to our work, Yaghoobzadeh et al. (2017) optimize a mention typing model for our entity typing task by introducing multi instance learning algorithms, resulting comparable performance to ELR(SKIP). We will investigate their

[8]For accuracy measure: in the following ordered lists of sets, $A<B$ means that all members (row numbers in Table 2) of A are significantly worse than all members of B: $\{1\} < \{2\} < \{3,\ldots,11\} < \{12{,}13\} < \{14{,}15{,}16\} < \{17,\ldots,23\}$. Test of equal proportions, $\alpha < 0.05$. See Table 6 in the appendix for more details.

		all entities			head entities			tail entities		
		acc	mic	mac	acc	mic	mac	acc	mic	mac
1	MFT	.000	.041	.041	.000	.044	.044	.000	.038	.038
2	CLR(FORWARD)	.066	.379	.352	.067	.342	.369	.061	.374	.350
3	CLR(LSTM)	.121	.425	.396	.122	.433	.390	.116	.408	.391
4	CLR(BiLSTM)	.133	.440	.404	.129	.443	.394	.135	.428	.404
5	CLR(NSL)	.164	.484	.464	.157	.470	.443	.173	.483	.472
6	CLR(CNN)	.177	.494	.468	.171	.484	.450	.187	.489	.474
7	BOW	.113	.346	.379	.109	.323	.353	.120	.356	.396
8	WWLR(SKIP)	.214	.581	.531	.293	.660	.634	.173	.528	.478
9	WWLR(SSKIP)	.223	.584	.543	.306	.667	.642	.183	.533	.494
10	SWLR	.236	.590	.554	.301	.665	.632	.209	.551	.522
11	BOW+CLR(NSL)	.156	.487	.464	.157	.480	.452	.159	.485	.469
12	WWLR+CLR(CNN)	.257	.603	.568	.317	.668	.637	.235	.567	.538
13	SWLR+CLR(CNN)	.241	.594	.561	.295	.659	.628	.227	.560	.536
14	ELR(SKIP)	.488	.774	.741	.551	.834	.815	.337	.621	.560
15	ELR(SSKIP)	.515	.796	.763	.560	.839	.819	.394	.677	.619
16	AGG-FIGER	.320	.694	.660	.396	.762	.724	.220	.593	.568
17	ELR+CLR	.554	.816	.788	.580	.844	.825	.467	.733	.690
18	ELR+WWLR	.557	.819	.793	.582	.846	.827	.480	.749	.708
19	ELR+SWLR	.558	.820	.796	.584	.846	.829	.480	.751	.714
20	ELR+WWLR+CLR	.568	.823	.798	.590	.847	.829	.491	.755	.716
21	ELR+SWLR+CLR	.569	.824	.801	.590	.849	**.831**	.497	.760	.724
22	ELR+WWLR+CLR+TC	.572	.824	.801	.594	.849	**.831**	.499	.759	.722
23	ELR+SWLR+CLR+TC	**.575**	**.826**	**.802**	**.597**	**.851**	**.831**	**.508**	**.762**	**.727**

Table 2: Accuracy (acc), micro (mic) and macro (mac) F_1 on test for all, head and tail entities.

types:	all	head	tail
AGG-FIGER	.566	.702	.438
ELR	.621	.784	.480
MuLR	**.669**	**.811**	**.541**

Table 3: Type macro average F_1 on test for all, head and tail types. MuLR = ELR+SWLR+CLR+TC

	all	known?	
		yes	no
CLR(NSL)	.484	.521	.341
CLR(CNN)	.494	.524	.374
BOW	.346	.435	.065
SWLR	.590	.612	.499
BOW+NSL	.497	.535	.358
SWLR+CLR(CNN)	**.594**	**.616**	**.508**

Table 4: Micro F_1 on test of character, word level models for all, known ("known? yes") and unknown ("known? no") entities.

method in future.

Joining CLR with ELR (line 17) results in large improvements, especially for tail entities (5% micro F_1). This demonstrates that for rare entities, contextual information is often not sufficient for an informative representation, hence name features are important. This is also true for the joint models of WWLR/SWLR and ELR (lines 18-19). Joining WWLR works better than CLR, and SWLR is slightly better than WWLR. Joint models of WWLR/SWLR with ELR+CLR gives more improvements, and SWLR is again slightly better than WWLR. ELR+WWLR+CLR and ELR+SWLR+CLR, are better than their two-level counterparts, again confirming that these levels are complementary.

We get a further boost, especially for tail entities, by also including TC (type cosine) in the combinations (lines 22-23). This demonstrates the potential advantage of having a common representation space for entities and types. Our best model, ELR+SWLR+CLR+TC (line 22), which we refer to as MuLR in the other tables, beats our initial baselines (ELR and AGG-FIGER) by large margins, e.g., in tail entities improvements are more than 8% in micro F1.

Table 3 shows **type macro** F_1 for MuLR (ELR+SWLR+CLR+TC) and two baselines. There are 11 head types (those with $\geq$3000 train entities) and 36 tail types (those with $<$200 train entities). These results again confirm the superiority of our multi-level models over the baselines: AGG-FIGER and ELR, the best single-level model baseline.

4.3 Analysis

Unknown vs. known entities. To analyze the complementarity of character and word level representations, as well as more fine-grained comparison of our models and the baselines, we divide test entities into *known entities* – at least one word of the entity's name appears in a train entity – and *unknown entities* (the complement). There are 45,000 (resp. 15,000) known (resp. unknown) test entities.

Table 4 shows that the CNN works only slightly better (by 0.3%) than NSL on known entities, but works much better on unknown entities (by 3.3%), justifying our preference for deep learning CLR models. As expected, BOW works relatively well for known entities and really poorly for unknown entities. SWLR beats CLR models as well as BOW. The reason is that in our setup, word embeddings are induced on the entire corpus using an unsupervised algorithm. Thus, even for many words that did not occur in train, SWLR has ac-

cess to informative representations of words. The joint model, SWLR+CLR(CNN), is significantly better than BOW+CLR(NSL) again due to limits of BOW. SWLR+CLR(CNN) is better than SWLR in unknown entities.

Case study of LIVING-THING. To understand the interplay of different levels better, we perform a case study of the type LIVING-THING. Living beings that are not humans belong to this type.

WLRs incorrectly assign "Walter Leaf" (PERSON) and "Along Came A Spider" (MUSIC) to LIVING-THING because these names contain a word referring to a LIVING-THING ("leaf", "spider"), but the entity itself is not a LIVING-THING. In these cases, the averaging of embeddings that WLR performs is misleading. The CLR(CNN) types these two entities correctly because their names contain character ngram/shape patterns that are indicative of PERSON and MUSIC.

ELR incorrectly assigns "Zumpango" (CITY) and "Lake Kasumigaura" (LOCATION) to LIVING-THING because these entities are rare and words associated with living things (e.g., "wildlife") dominate in their contexts. However, CLR(CNN) and WLR enable the joint model to type the two entites correctly: "Zumpango" because of the informative suffix "-go" and "Lake Kasumigaura" because of the informative word "Lake".

While some of the **remaining errors** of our best system MuLR are due to the inherent difficulty of entity typing (e.g., it is difficult to correctly type a one-word entity that occurs once and whose name is not informative), many other errors are due to artifacts of our setup. First, ClueWeb/FACC1 is the result of an automatic entity linking system and any entity linking errors propagate to our models. Second, due to the incompleteness of Freebase (Yaghoobzadeh and Schütze, 2015), many entities in the FIGMENT dataset are incompletely annotated, resulting in correctly typed entities being evaluated as incorrect.

Adding another source: description-based embeddings. While in this paper, we focus on the contexts and names of entities, there is a textual source of information about entities in KBs which we can also make use of: descriptions of entities. We extract Wikipedia descriptions of FIGMENT entities filtering out the entities ($\sim$ 40,000 out of $\sim$ 200,000) without description.

We then build a simple entity representation by averaging the embeddings of the top k words (wrt tf-idf) of the description (henceforth, AVG-DES).[9] This representation is used as input in Figure 1 to train the MLP. We also train our best multi-level model as well as the joint of the two on this smaller dataset. Since the descriptions are coming from Wikipedia, we use 300-dimensional Glove (URL, 2016a) embeddings pretrained on Wikipdia+Gigaword to get more coverage of words. For MuLR, we still use the embeddings we trained before.

entities:	all	head	tail
AVG-DES	.773	.791	.745
MuLR	.825	.846	.757
MuLR+AVG-DES	**.873**	**.877**	**.852**

Table 5: Micro average F_1 results of MuLR and description based model and their joint.

Results are shown in Table 5. While for head entities, MuLR works marginally better, the difference is very small in tail entities. The joint model of the two (by concatenation of vectors) improves the micro F1, with clear boost for tail entities. This suggests that for tail entities, the contextual and name information is not enough by itself and some keywords from descriptions can be really helpful. Integrating more complex description-based embeddings, e.g., by using CNN (Xie et al., 2016), may improve the results further. We leave it for future work.

5 Conclusion

In this paper, we have introduced representations of entities on different levels: character, word and entity. The character level representation is learned from the entity name. The word level representation is computed from the embeddings of the words w_i in the entity name where the embedding of w_i is derived from the corpus contexts of w_i. The entity level representation of entity e_i is derived from the corpus contexts of e_i. Our experiments show that each of these levels contributes complementary information for the task of fine-grained typing of entities. The joint model of all three levels beats the state-of-the-art baseline by large margins. We further showed that extracting some keywords from Wikipedia descriptions of entities, when available, can considerably improve entity representations, especially for rare entities. We believe that our findings can be transferred to other tasks where entity representation matters.

[9] k = 20 gives the best results on dev.

Acknowledgments. This work was supported by DFG (SCHU 2246/8-2).

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 18–26, Berlin, Germany, August. Association for Computational Linguistics.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1236–1242.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878, Lisbon, Portugal, September. Association for Computational Linguistics.

Cícero Nogueira dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. *CoRR*, abs/1505.05008.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1818–1826.

Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 260–269, Berlin, Germany, August. Association for Computational Linguistics.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10:146–162.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2741–2749.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado, May–June. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, September. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August. Association for Computational Linguistics.

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

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525, Denver, Colorado, May–June. Association for Computational Linguistics.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1825–1834.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. pages 69–74, June.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1333–1339.

URL. 2016a. Glove project. `http://nlp.stanford.edu/projects/glove`.

URL. 2016b. Lemur project. `http://lemurproject.org/clueweb12/FACC1`.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619.

Zhigang Wang and Juan-Zi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1293–1299.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, Doha, Qatar, October. Association for Computational Linguistics.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2659–2665.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725, Lisbon, Portugal, September. Association for Computational Linguistics.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 236–246, Berlin, Germany, August. Association for Computational Linguistics.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise mitigation for neural entity typing and relation extraction. In *EACL*, Valencia, Spain.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. pages 250–259, August.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 291–296, Beijing, China, July. Association for Computational Linguistics.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *CoRR*, abs/1502.01710.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. pages 649–657.

A Supplementary Material

All entities

	Models	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
01	MFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	CLR(FORWARD)	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	CLR(LSTM)	*	*	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	CLR(BiLSTM)	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	CLR(CNN)	*	*	*	*	0	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
06	CLR(NSL)	*	*	*	*	0	0	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
07	BOW	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	WWLR(SkipG)	*	*	*	*	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
09	WWLR(SSkipG)	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
10	SWLR	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0	0	0	0
11	BOW+CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	WWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0	0
13	SWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0
14	ELR(SkipG)	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0
15	ELR(SSkipG)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0
16	AGG-FIGER	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0
17	ELR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
18	ELR+WWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
19	ELR+SWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
20	ELR+WWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
21	ELR+SWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
22	ELR+WWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
23	ELR+SWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0

Head entities

	Models	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
01	MFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	CLR(FORWARD)	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	CLR(LSTM)	*	*	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	CLR(BiLSTM)	*	*	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	CLR(CNN)	*	*	*	*	0	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
06	CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	BOW	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	WWLR(SkipG)	*	*	*	*	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
09	WWLR(SSkipG)	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
10	SWLR	*	*	*	*	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
11	BOW+CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	WWLR+CLR(CNN)	*	*	*	*	*	*	*	*	0	*	*	0	*	0	0	0	0	0	0	0	0	0	0
13	SWLR+CLR(CNN)	*	*	*	*	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
14	ELR(SkipG)	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0
15	ELR(SSkipG)	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0
16	AGG-FIGER	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0
17	ELR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
18	ELR+WWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
19	ELR+SWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
20	ELR+WWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
21	ELR+SWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
22	ELR+WWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
23	ELR+SWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0

Tail entities

	Models	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
01	MFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	CLR(FORWARD)	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	CLR(LSTM)	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	CLR(BiLSTM)	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	CLR(CNN)	*	*	*	*	0	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
06	CLR(NSL)	*	*	*	*	0	0	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
07	BOW	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	WWLR(SkipG)	*	*	*	*	0	0	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
09	WWLR(SSkipG)	*	*	*	*	0	0	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
10	SWLR	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0	0	0	0
11	BOW+CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	WWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	*	0	0	0	0	0	0
13	SWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0
14	ELR(SkipG)	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0
15	ELR(SSkipG)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0
16	AGG-FIGER	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0	0	0	0
17	ELR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
18	ELR+WWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
19	ELR+SWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
20	ELR+WWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0
21	ELR+SWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
22	ELR+WWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
23	ELR+SWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0

Table 6: Significance-test results for accuracy measure for all, head and tail entities. If the result for the model in a row is significantly larger than the result for the model in a column, then the value in the corresponding (row,column) is * and otherwise is 0.

Chapter 5

Noise Mitigation for Neural Entity Typing and Relation Extraction

Noise Mitigation for Neural Entity Typing and Relation Extraction

Yadollah Yaghoobzadeh* and **Heike Adel*** and **Hinrich Schütze**
* *These authors contributed equally to this work*
Center for Information and Language Processing
LMU Munich, Germany
yadollah|heike@cis.lmu.de

Abstract

In this paper, we address two different types of noise in information extraction models: noise from distant supervision and noise from pipeline input features. Our target tasks are entity typing and relation extraction. For the first noise type, we introduce multi-instance multi-label learning algorithms using neural network models, and apply them to fine-grained entity typing for the first time. Our model outperforms the state-of-the-art supervised approach which uses global embeddings of entities. For the second noise type, we propose ways to improve the integration of noisy entity type predictions into relation extraction. Our experiments show that probabilistic predictions are more robust than discrete predictions and that joint training of the two tasks performs best.

1 Introduction

Knowledge bases (KBs) are important resources for natural language processing tasks like question answering and entity linking. However, KBs are far from complete (e.g., Socher et al. (2013)). Therefore, methods for automatic knowledge base completion (KBC) are beneficial. Two subtasks of KBC are *entity typing (ET)* and *relation extraction (RE)*. We address both tasks in this paper.

As in other information extraction tasks, obtaining labeled training data for ET and RE is challenging. The challenge grows as labels become more fine-grained. Therefore, distant supervision (Mintz et al., 2009) is widely used. It reduces the need for manually created resources. Distant supervision assumes that if an entity has a type (resp. two entities have a relationship) in a KB, then all sentences mentioning that entity (resp. those two entities) express that type (resp. that relationship). However, that assumption is too strong and gives rise to many *noisy* labels. Different techniques to deal with that problem have been investigated. The main technique is multi-instance (MI) learning (Riedel et al., 2010). It relaxes the distant supervision assumption to the assumption that at least one instance of a bag (collection of all sentences containing the given entity/entity pair) expresses the type/relationship given in the KB. Multi-instance multi-label (MIML) learning is a generalization of MI in which one bag can have several labels (Surdeanu et al., 2012).

Most MI and MIML methods are based on hand crafted features. Recently, Zeng et al. (2015) introduced an end-to-end approach to MI learning based on neural networks. Their MI method takes the most confident instance as the prediction of the bag. Lin et al. (2016) further improved that method by taking other instances into account as well; they proposed MI learning based on selective attention as an alternative way of relaxing the impact of noisy labels on RE. In selective attention, a weighted average of instance representations is calculated first and then used to compute the prediction of a bag.

In this paper, we introduce two multi-label versions of MI. (i) *MIML-MAX* takes the maximum instance for each label. (ii) *MIML-ATT* applies, for each label, selective attention to the instances. We apply MIML-MAX and MIML-ATT to fine-grained ET. In contrast to RE, the ET task we consider contains a larger set of labels, with a variety of different granularities and hierarchical relationships. We show that MIML-ATT deals well with noise in corpus-level ET and improves or matches the results of a supervised model based on global embeddings of entities.

The second type of noise we address in this paper influences the integration of ET into RE. It has

been shown that adding entity types as features improves RE models (cf. Ling and Weld (2012), Liu et al. (2014)). However, noisy training data and difficulties of classification often cause wrong predictions of ET and, as a result, noisy inputs to RE. To address this, we propose a joint model of ET and RE and compare it with methods that integrate ET results in a strict pipeline. The joint model performs best. Among the pipeline models, we show that using probabilities instead of binary decisions better deals with noise (i.e., possible ET errors).

To sum up, our contributions are as follows. (i) We introduce new algorithms for MIML using neural networks. (ii) We apply MIML to fine-grained entity typing for the first time and show that it outperforms the state-of-the-art supervised method based on entity embeddings. (iii) We show that a novel way of integrating noisy entity type predictions into a relation extraction model and joint training of the two tasks lead to large improvements of RE performance.

We release code and data for future research.[1]

2 Related Work

Noise mitigation for distant supervision. Distant supervision can be used to train information extraction systems, e.g., in relation extraction (e.g., Mintz et al. (2009), Riedel et al. (2010), Hoffmann et al. (2011), Zeng et al. (2015)) and entity typing (e.g., Ling and Weld (2012), Yogatama et al. (2015), Dong et al. (2015)). To mitigate the noisy label problem, multi-instance (MI) learning has been introduced and applied in relation extraction (Riedel et al., 2010; Ritter et al., 2013). Surdeanu et al. (2012) introduced multi-instance multi-label (MIML) learning to extend MI learning for multi-label relation extraction. Those models are based on manually designed features. Zeng et al. (2015) and Lin et al. (2016) introduced MI learning methods for neural networks. We introduce MIML algorithms for neural networks. In contrast to most MI/MIML methods, which are applied in relation extraction, we apply MIML to the task of fine-grained entity typing. Ritter et al. (2013) applied MI on a Twitter dataset with ten types. Our dataset has a larger number of classes or types (namely 102) and input examples, compared to that Twitter dataset and also to the most widely used datasets for evaluating MI (cf. Riedel et al. (2010)). This makes our setup more challenging because of different dependencies and the multi-label nature of the problem. Also, there seems to be a difference between how entity relations and entity types are expressed in text. Our experiments support that hypothesis.

Knowledge base completion (KBC). Most KBC systems focus on identifying triples $R(e_1, r, e_2)$ missing from a KB (Nickel et al., 2012; Bordes et al., 2013; Weston et al., 2013; Socher et al., 2013; Jiang et al., 2012; Riedel et al., 2013; Wang et al., 2014). Work on entity typing or unary relations for KBC is more recent (Yao et al., 2013; Neelakantan and Chang, 2015; Yaghoobzadeh and Schütze, 2015; **?**). In this paper, we build a KBC system for unary and binary relations using contextual information of words and entities.

Named entity recognition (NER) and typing. NER systems (e.g., Finkel et al. (2005), Collobert et al. (2011)) used to consider only a small set of entity types. Recent work also addresses fine-grained NER (Yosef et al., 2012; Ling and Weld, 2012; Yogatama et al., 2015; Dong et al., 2015; Del Corro et al., 2015; **?**; Ren et al., 2016; Shimaoka et al., 2016). Some of this work (cf. Yogatama et al. (2015), Dong et al. (2015)) treats entity segment boundaries as given and classifies mentions into fine-grained types. We make a similar assumption, but in contrast to NER, we evaluate on the corpus-level entity typing task of Yaghoobzadeh and Schütze (2015); thus, we do not need test sentences annotated with context dependent entity types. This task was also used to evaluate embedding learning methods (**?**).

Entity types for relation extraction. Several studies have integrated entity type information into relation extraction – either coarse-grained (Hoffmann et al., 2011; GuoDong et al., 2005) or fine-grained (Liu et al., 2014; Du et al., 2015; Augenstein et al., 2015; Vlachos and Clark, 2014; Yao et al., 2010; Ling and Weld, 2012) entity types. In contrast to most of this work, but similar to Yao et al. (2010), we do not incorporate binary entity type values, but probabilistic outputs. Thus, we allow the relation extraction system to compensate for errors of entity typing. Additionally, we compare this approach to various other possibilities, to investigate which approach performs best. Yao et al. (2010) found that joint training of entity typing and relation extraction is better than a pipeline model; we show that this result also holds

[1] cistern.cis.lmu.de

for neural network models and when the number of entity types is large.

3 MIML Learning for Entity Typing

Entity typing (ET) is the task of finding, for each named entity, a set of types or classes that it belongs to, e.g., "author" and "politician" for "Obama". Our goal is corpus-level prediction of entity types. We use the entity-type information from a KB and annotated contexts of entities in a corpus to estimate $P(t|e)$, the probability that entity e has type t.

More specifically, consider an entity e and $B = \{c_1, c_2, ..., c_q\}$, the set of q contexts of e in the corpus. Each c_i is an instance of e and since e can have several labels, it is a multi-instance multi-label (MIML) learning problem. We address MIML using neural networks by representing each context as a vector $\vec{c_i} \in \mathbb{R}^h$, and learn $P(t|e)$ from the set of contexts of entity e. In the following, we first describe our MIML algorithms and then explain how $\vec{c_i}$ is computed.

Notations and definitions. Lowercase letters (e.g., e) refer to variables. Lowercase letters with an upper arrow (e.g., $\vec{e}$) are vectors. We define BCE, binary cross entropy, as follows where y is a binary variable and $\hat{y}$ is a real valued variable between 0 and 1.

$$\text{BCE}(y, \hat{y}) = -\Big(y \log(\hat{y}) + (1-y)(1-\log(\hat{y}))\Big)$$

3.1 Algorithms

Distant supervision. The basic way to estimate $P(t|e)$ is based on distant supervision with learning the type probability of each c_i individually, by making the assumption that each c_i expresses all labels of e. Therefore, we define the context-level probability function as:

$$P(t|c_i) = \sigma(\vec{w_t}\vec{c_i} + b_t) \tag{1}$$

where $\vec{w_t} \in \mathbb{R}^h$ is the output weight vector and b_t is the bias scalar for type t. The cost function is defined based on binary cross entropy:

$$L(\theta) = \sum_c \sum_t \text{BCE}(y_t, P(t|c)) \tag{2}$$

where y_t is 1 if entity e has type t otherwise 0. To compute $P(t|e)$ at prediction time, i.e., $P_{\text{pred}}(t|e)$, the context-level probabilities must be aggregated. Average is the usual way of doing that:

$$P_{\text{pred}}(t|e) = \frac{1}{q} \sum_{i=1}^{q} P(t|c_i) \tag{3}$$

Multi-instance multi-label. The distant supervision assumption is that *all* contexts of an entity with type t are contexts of t; e.g., we label all contexts mentioning "Barack Obama" with all of his types. Obviously, the labels are incorrect or *noisy* for some contexts. Multi-instance multi-label (MIML) learning addresses this problem. We apply MIML to fine-grained ET for the first time. Our assumption is: if entity e has type t, then there is at least one context of e in the corpus in which e occurs as type t. So, we apply this assumption during training with the following estimation of the type probability of an entity:

$$P(t|e) = \max_{1 \le i \le q} P(t|c_i) \tag{4}$$

which means we take the **maximum** probability of type t over all contexts of entity e as $P(t|e)$. We call this approach **MIML-MAX**.

MIML-MAX picks the most confident context for t, ignoring the probabilities of all the other contexts. Apart from missing information, this can be especially harmful if the entity annotations in the corpus are the result of an entity linking system. In that case, the most confident context might be wrongly linked to the entity. So, it can be beneficial to leverage all contexts into the final prediction, e.g., by **averaging** the type probabilities of all contexts of entity e:

$$P(t|e) = \frac{1}{q} \sum_{i=1}^{q} P(t|c_i) \tag{5}$$

We call this approach **MIML-AVG**. We also propose a combination of the maximum and average, which uses MIML-MAX (Eq. 4) in training and MIML-AVG (Eq. 5) in prediction. We call this approach **MIML-MAX-AVG**.

MIML-AVG treats every context equally which might be problematic since many contexts are irrelevant for a particular type. A better way is to weight the contexts according to their similarity to the types. Therefore, we propose using selective **attention** over contexts as follows and call this approach **MIML-ATT**. MIML-ATT is the multi-label version of the selective attention method proposed in Lin et al. (2016). To compute the type probability for e, we define:

$$P(t|e) = \sigma(\vec{w_t}\vec{a_t} + b_t) \tag{6}$$

where $\vec{w_t} \in \mathbb{R}^h$ is the output weight vector and b_t the bias scalar for type t, and $\vec{a_t}$ is the aggregated representation of all contexts c_i of e for type

Model	Train	Prediction
MIML-MAX	MAX	MAX
MIML-AVG	AVG	AVG
MIML-MAX-AVG	MAX	AVG
MIML-ATT	ATT	ATT

Table 1: Different MIML algorithms for entity typing, and the aggregation function they use to get corpus-level probabilities.

t, computed as follows:

$$\vec{a_t} = \sum_i \alpha_{i,t} \vec{c_i} \tag{7}$$

where $\alpha_{i,t}$ is the attention score of context c_i for type t and $\vec{a_t} \in \mathbb{R}^h$ can be interpreted as the representation of entity e for type t.

$\alpha_{i,t}$ is defined as:

$$\alpha_{i,t} = \frac{\exp(\vec{c_i}\mathbf{M}\vec{t})}{\sum_{j=1}^{q} \exp(\vec{c_j}\mathbf{M}\vec{t})} \tag{8}$$

where $\mathbf{M} \in \mathbb{R}^{h \times d_t}$ is a weight matrix that measures the similarity of $\vec{c}$ and $\vec{t}$. $\vec{t} \in \mathbb{R}^{d_t}$ is the representation of type t.

Table 1 summarizes the differences of our MIML methods with respect to the aggregation function they use to get corpus-level probabilities. For optimization of all MIML methods, we use the binary cross entropy loss function,

$$L(\theta) = \sum_e \sum_t \text{BCE}(y_t, P(t|e)) \tag{9}$$

In contrast to the loss function of distant supervision in Eq. 2, which iterates over all *contexts*, we iterate over all *entities* here.

3.2 Context Representation

To produce a high-quality context representation $\vec{c}$, we use convolutional neural networks (CNNs).

The first layer of the CNN is a *lookup table* that maps each word in c to an embedding of size d. The output of the lookup layer is a matrix $E \in \mathbb{R}^{d \times s}$ (the embedding layer), where s is the context size (a fixed number of words).

The CNN uses n filters of different window widths w to narrowly convolve E. For each of the n filters $H \in \mathbb{R}^{d \times w}$, the result of applying H to matrix E is a feature map $\vec{m} \in \mathbb{R}^{s-w+1}$:

$$\text{m}[i] = g(E_{:,i:i+w-1} \odot H) \tag{10}$$

where g is the $relu$ function, $\odot$ is the Frobenius product, $E_{:,i:i+w-1}$ are the columns i to $i+w-1$ of E and $1 \leq w \leq k$ are the window widths we consider. Max pooling then gives us one feature for each filter and the concatenation of those features is the CNN representation of c.

As it is shown in the entity typing part of Figure 1, we apply the CNN to the left and right context of the entity mention and the concatenation $\vec{\phi}(c) \in \mathbb{R}^{2n}$ is fed into a multi-layer perceptron (MLP) to get the final context representation $\vec{c} \in \mathbb{R}^h$:

$$\vec{c} = \tanh\left(\mathbf{W}_h \vec{\phi}(c)\right) \tag{11}$$

4 Type-aware Relation Extraction

Relation extraction (RE) is mostly defined as finding relations between pairs of entities, for instance, finding the relation "president-of" between "Obama" and "USA". Given a set of q contexts for an entity pair z, $B = \{c_1, c_2, ..., c_q\}$ in the corpus, we learn $P(r|z)$, which is the probability of relation r for z. We assume that each z has one relation $r(z)$. Each c_i is represented by a vector $\vec{c_i} \in \mathbb{R}^h$, which is our type-aware representation of context described in Section 4.1.

To learn $P(r|z)$, we use the multi-instance (MI) learning method of Zeng et al. (2015):

$$\begin{aligned} P(r|c_i) &= \text{softmax}\left(\mathbf{W}_{\text{out}} \vec{c_i}\right), \\ P(r|z) &= \max_{1 \leq i \leq q} P(r|c_i) \end{aligned} \tag{12}$$

where $P(r|c_i)$ is the probability of relation r for context c_i. The cost function we optimize is:

$$L(\theta) = -\sum_z \log P(r(z)|z)$$

4.1 Context Representation

Similar to our entity typing system, we apply CNNs to compute the context representation $\vec{\phi}(c)$. In particular, we use Adel et al. (2016)'s CNN. It uses an input representation designed for RE. Each sentence is split into three parts: left of the relation arguments, between the relation arguments and right of the relation arguments. The parts "overlap", i.e., the left (resp. right) argument is included in both left (resp. right) and middle parts. For each of the three parts, convolution and 3-max pooling (Kalchbrenner et al., 2014) is performed. The context representation $\vec{\phi}(c) \in \mathbb{R}^{3 \cdot 3 \cdot n}$ is the concatenation of the pooling results.

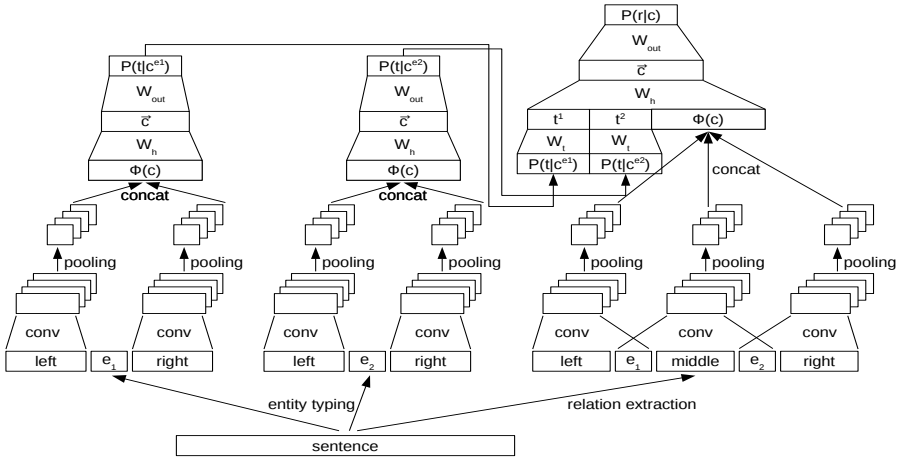

Figure 1: Our architecture for joint entity typing and relation extraction

4.1.1 Integration of Entity Types

We concatenate the entity type representations $\vec{t^1} \in \mathbb{R}^\tau$ and $\vec{t^2} \in \mathbb{R}^\tau$ of the relation arguments to the CNN representation of the context, $\vec{\phi}(c)$:

$$\vec{\phi}(c)' = [\vec{\phi}(c) : \vec{t^1} : \vec{t^2}] \quad (13)$$

Our context representation $\vec{c}$ is then:

$$\vec{c} = \tanh\left(\mathbf{W}_h \vec{\phi}(c)'\right) \quad (14)$$

where $\mathbf{W}_h \in \mathbb{R}^{h \times (3 \cdot 3 \cdot n + 2\tau)}$ is the weight matrix. This is also depicted in Figure 1, right column, third layer from the top: $\vec{t^1}$, $\vec{t^2}$, $\vec{\Phi}(c)$. We calculate $\vec{t^1}$ and $\vec{t^2}$ from the predictions of the entity typing model with the following transformation:

$$\vec{t^k} = f\Big(\mathbf{W_t}[P(t_1|c^{e_k}) \dots P(t_T|c^{e_k})]\Big) \quad (15)$$

where c^{e_k} is the context of e_k, $\mathbf{W_t} \in \mathbb{R}^{\tau \times T}$ is a weight matrix (learned from corpus or during training) and f is a function (identity or $\tanh$). With the transformation $\mathbf{W_t}$, the model can combine predictions for different types to learn better internal representations t^1 and t^2. The choices of $\mathbf{W_t}$ and f depend on the different representations we investigate and describe in the following.

(1) Pipeline: We integrate entity types into the RE model, using the output of ET in a pipeline model (see Eq. 15). We test the following representations of $\vec{t^k}$, $k \in \{1, 2\}$. **PREDICTED-HIDDEN**: $\mathbf{W_t}$ from Eq. 15 is learned during training and f is tanh. That means that a hidden layer learns representations based on the predictions $P(t_1|c^{e_k}) \dots P(t_T|c^{e_k})$. **BINARY-HIDDEN**: This is the binarization of the input of PREDICTED-HIDDEN, i.e., each probability estimate is converted to 0 or 1 (with a threshold of 0.5). **BINARY**: $\vec{t^k}$ is the binary vector itself (used by Ling and Weld (2012)). **WEIGHTED**: The columns of matrix $\mathbf{W_t}$ from Eq. 15 are the distributional embeddings of types trained on the corpus (see Section 5.1). f is the identity function.

(2) Joint model: As an alternative to the pipeline model, we investigate integrating entity typing into RE by jointly training both models. We use the architecture depicted in Figure 1. The key difference to the pipeline model PREDICTED-HIDDEN is that we learn $P(t|c)$ and $P(r|c)$ jointly, called **JOINT-TRAIN**. We compare JOINT-TRAIN to other models, including the pipeline models.

During training of JOINT-TRAIN, we compute the cost of the ET model for typing the first entity $L_1(\theta_T)$, the cost for typing the second entity $L_2(\theta_T)$ and the cost of the RE model for assigning a relation to the two entities $L(\theta_R)$. Then, we combine those costs with a weight γ which is tuned on the development set:

$$L(\theta) = \sum_z \Big(L_1(\theta_T) + L_2(\theta_T) + \gamma \cdot L(\theta_R)\Big),$$

$$L_i(\theta_T) = \sum_t \text{BCE}(y_t^{e_i}, P(t|c^{e_i})),$$

$$L(\theta_R) = -\log P(r(z)|z)$$

$P(r|z)$ is computed based on Eq. 12.

Note that based on this equation, the ET parameters are optimized on the contexts of the RE examples, which are a subset of all training examples of ET. However in the pipeline models, ET is trained on the whole training set used for typ-

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	NA (no relation)
LOC.LOC.containedby	

Table 2: Selected relations for relation extraction; PPL = people, GOV = governement

ing. Also note that in JOINT-TRAIN we do not use MIML for the ET part but a distant supervised cost function.

5 Experimental Data, Setup and Results

For entity typing, we use CF-FIGMENT (URL, 2016b), a dataset published by Yaghoobzadeh and Schütze (2015). CF-FIGMENT is derived from a version of ClueWeb (URL, 2016c) in which Freebase entities are annotated using FACC1 (URL, 2016d; Gabrilovich et al., 2013). CF-FIGMENT contains 200,000 Freebase entities that were mapped to 102 FIGER types (Ling and Weld, 2012), divided into train (50%), dev (20%) and test (30%); and a set of 4,300,000 sentences (contexts) containing those entities.

For relation extraction, we first select the ten most frequent relations (plus NA for no relation according to Freebase) of entity pairs in CF-FIGMENT. We ensure that the entity pairs have at least one context in CF-FIGMENT. This results in 5815, 3054 and 6889 unique entity pairs for train, dev and test.[2] Dev and test set sizes are 124,462 and 556,847 instances. For the train set, we take a subsample of 135,171 sentences. The entity and sentence sets of CF-FIGMENT were constructed to ensure that entities in the entity test set do not occur in the sentence train and dev sets; that is, a sentence was assigned to the train set only if all entities it contains are train entities.[1]

5.1 Word, Entity and Type Embeddings

We use 100-dimensional word embeddings to initialize the input layer of ET and RE. Embeddings are kept fixed during training. Since we need embeddings for words, entities and types in the same space, we process ClueWeb+FACC1 (corpus with entity information) as follows. For each sentence s, we add two copies: s itself, and a copy in which each entity is replaced with its notable type, the most important type according to Freebase. We process train, dev and test this way, but do not replace test entities with their notable type because the types of test entities are unknown in our application scenario. We run word2vec (Mikolov et al., 2013) on the resulting corpus to learn embeddings for words, entities and types. Note that our application scenario is that we are given an unannotated input corpus and our system then extracts entity types and relations from this input corpus to enhance the KB.

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

5.2 Entity Typing Experiments

Entity context setup. We use a window size of 5 on each side of the entity mentions. Following Yaghoobzadeh and Schütze (2015), we replace other entities occurring in the context with their Freebase notable type mapped to FIGER.

Models. Yaghoobzadeh and Schütze (2015) applied a multi-layer perceptron (**MLP**) architecture to create context representations. Therefore, we use an MLP baseline to compute the context representation $\vec{\phi}(c)$. The input to the MLP model is a concatenation of context word embeddings. As an alternative to MLP, we also train a **CNN** (see Section 3.2) to compute context representations. We run experiments with MLP and CNN, each trained with standard distant supervision and with MIML.

EntEmb and FIGMENT baselines. Following Yaghoobzadeh and Schütze (2015), we also learn entity embeddings and classify those embeddings to types, i.e., instead of distant supervision, we classify entities based on aggregated information represented in entity embeddings. An MLP with one hidden layer is used as classifier. We call that model EntEmb. We join the results of EntEmb with our best model (line 13 in Table 3), similar to the joint model (FIGMENT) in Yaghoobzadeh and Schütze (2015).

We use the same **evaluation measures** as Ling and Weld (2012), Yaghoobzadeh and Schütze (2015) and Neelakantan and Chang (2015) for entity typing: precision at 1 ($P@1$), which is the accuracy of picking the most confident type for each entity, micro average F_1 of all entity-type assignments and mean average precision (MAP) over types. We could make assignment decisions based on the standard criterion $p > \theta$, $\theta = 0.5$, but we found that tuning θ improves results. For each probabilistic classifier and each type, we set θ to the value that maximizes performance on dev.

	P@1 all	F_1 all	F_1 head	F_1 tail	MAP
1 MLP	74.3	69.1	74.8	52.5	42.1
2 MLP+MIML-MAX	74.7	59.2	50.7	46.8	41.3
3 MLP+MIML-AVG	77.2	70.6	74.9	56.2	45.0
4 MLP+MIML-MAX-AVG	75.2	71.2	76.4	56.0	47.1
5 MLP+MIML-ATT	81.0	72.0	76.9	59.1	48.8
6 CNN	78.4	72.2	77.3	56.3	47.6
7 CNN+MIML-MAX	78.6	62.2	53.5	49.7	46.6
8 CNN+MIML-AVG	80.8	73.5	77.7	59.2	50.4
9 CNN+MIML-MAX-AVG	79.9	74.3	79.2	59.8	53.3
10 CNN+MIML-ATT	83.4	75.1	79.4	62.2	55.2
11 EntEmb	80.8	73.3	79.9	57.4	56.6
12 FIGMENT	81.6	74.3	80.3	60.1	57.0
13 CNN+MIML-ATT+EntEmb	**85.4**	**78.2**	**83.3**	**66.2**	**64.8**

Table 3: P@1, Micro F_1 for all, head and tail entities and MAP results for entity typing.

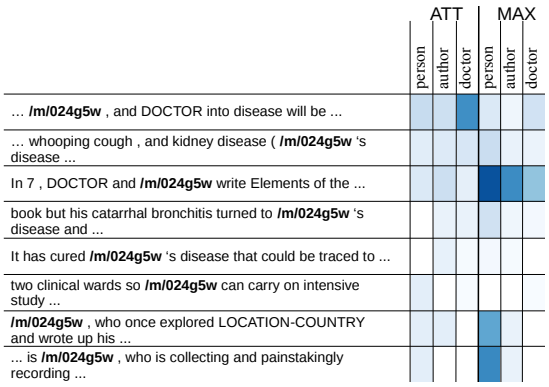

Figure 2: MIML-ATT and MIML-MAX scores for the example entity /m/024g5w.

Results. Table 3 shows results for P@1, micro F_1 and MAP. For F_1, we report separate results for all, head (frequency higher than 100) and tail (frequency less than 5) entities.

Discussion. The improvement of CNN (6) compared to MLP (1) is not surprising considering the effectiveness of CNNs in finding position independent local features, compared to the flat representation of MLP. Lines 2-5 and 7-10 show the results of different MIML algorithms for MLP and CNN, respectively. Considering micro F1 for all entities as the most importance measure, the trend is similar in both MLP and CNN for MIML methods: ATT $>$ MAX-AVG $>$ AVG $>$ MAX.

MAX is worse than even basic distant supervised models, especially for micro F1. MAX predictions are based on only one context of each entity (for each type), and the results suggest that this is harmful for entity typing. This is in contradiction with the previous results in RE (cf. Zeng et al. (2015)) and suggests that there might be a significant difference between expressing types of entities and relations between them in text. Related to this, MAX-AVG which averages the type probabilities at prediction time improves MAX by a large margin. Averaging the context probabilities seems to be a way to smooth the entity type probabilities. MAX-AVG models are also better than the corresponding models with AVG that train and predict with averaging. This is due to the fact that AVG gives equal weights to all context probabilities both in training and prediction. ATT uses weighted contexts in both training and prediction and that is probably the reason for its effectiveness over all other MIML algorithms. Overall, using attention (ATT) significantly improves the results of both MLP and CNN models.

CNN+MIML-ATT (10) performs comparable to EntEmb (11), with better micro F1 on all and tail entities and worse MAP and micro F1 on head entities. These two models have different properties, e.g., MIML is also able to type each mention of entities while EntEmb works only for corpus-level typing of entities. (See Yaghoobzadeh and Schütze (2015) for more differences) It is important to note that MIML can also be applied to any entity typing architecture or model that is trained by distant supervision. Due to the lack of large annotated corpora, distant supervision is currently the only viable approach to fine-grained entity typing; thus, our demonstration of the effectiveness of MIML is an important finding for entity typing.

Joining the results of CNN+MIML-ATT with EntEmb (line 13) gives large improvements over each of the single models. It is also consistently better (by more than 3% in all measures) than our baseline FIGMENT (12), which is basically MLP+EntEmb. This improvement is achieved by using CNN instead of MLP for context representation and integrating MIML-ATT. EntEmb is improved by **?**) by using entity names. We leave the integration of that model to future work.

Example. To show the behavior of MIML-MAX and MIML-ATT, we extract the scores that each method assigns to the labels for each context. A comparison for the example entity "Richard Bright" (MID: /m/024g5w) who is a PERSON, DOCTOR and AUTHOR is shown in Figure 2. Note that the weights from MIML-ATT (Eq. 8) sum to 1 for each label because of the applied softmax function while the scores from MIML-MAX (Eq. 1) do not. For both methods, the scores for the type PERSON are more equally distributed than for the other types which makes sense since the entity has the PERSON characteristics in every sentence. For

the other types, both models seem to be influenced by other entities occurring in the context (e.g., an occurrence with another DOCTOR could indicate that the entity is also a DOCTOR) but also by trigger words such as "write" or "book" for the type AUTHOR or "disease" for the type DOCTOR.

5.3 Relation Extraction Experiments

Models. In our experiments, we compare two state-of-the-art RE architectures: piecewise CNN (Zeng et al., 2015) and contextwise CNN (Adel et al., 2016). We use the publicly available implementation for the piecewise CNN (URL, 2016a) and our own implementation for the contextwise CNN. Both CNNs represent the input words with embeddings and split the contexts based on the positions of the relation arguments. The contextwise CNN splits the input before convolution, the piecewise CNN after convolution. Also, while the piecewise CNN applies a softmax layer directly after pooling, the contextwise CNN feeds the pooling results into a fully-connected hidden layer first. For both models, we use MI learning to mitigate the noise from distant supervision.

Results. The precision recall (PR) curves in Figure 3 show that the contextwise CNN outperforms the piecewise CNN on our RE dataset. We also compare them to a baseline model that does not learn context features but uses only the embeddings of the relation arguments as an input and feeds them into an MLP (similar to the EntEmb baseline for ET). The results confirm that the context features which the CNNs extract are very important, not only for ET but also for RE. Note that the 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 Ritter et al. (2013), we compute the area A under the PR curves which supports this trend (EntEmb: $A = 0.34$, piecewise CNN: $A = 0.48$, contextwise CNN: $A = 0.50$).

Pipeline vs. joint training. Since the contextwise CNN outperforms the piecewise CNN, we use the contextwise CNN for integrating entity types. Figure 4 shows that the performance on the RE dataset increases when we integrate entity type information into the CNN. The main trend of the PR curves and the areas under them shows the following order of model performances: JOINT-TRAIN > WEIGHTED > PREDICTED-HIDDEN > BINARY-HIDDEN > BINARY.

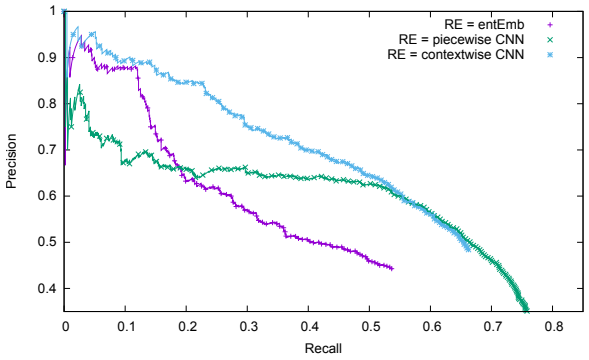

Figure 3: PR curves: relation extraction models

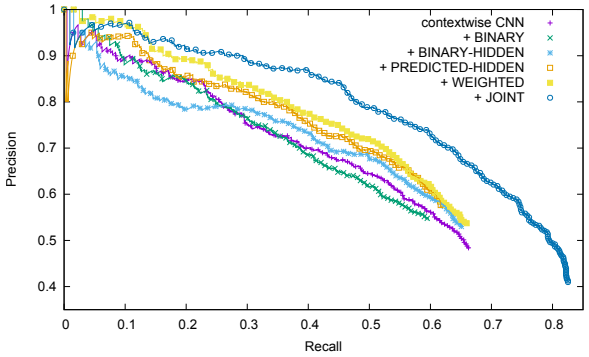

Figure 4: PR curves: type-aware relation extraction models

Discussion. The better performance of our approaches of integrating type predictions into the contextwise CNN (PREDICTED-HIDDEN, WEIGHTED) compared to baseline type integrations (BINARY, BINARY-HIDDEN) shows that probabilistic predictions of an entity typing system can be a valuable resource for RE. 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 preserve this information. Thus, using probabilistic representations, the RE system can compensate for noise in ET predictions.

WEIGHTED with access to the distributional type embeddings learned from the corpus works better than all other pipeline models. This shows that our type embeddings can be valuable for RE. JOINT-TRAIN performs better than all pipeline models, even though the ET part in the pipelines is trained on more data. The area of JOINT-TRAIN under the PR curve is $A = 0.66$. A plausible reason is the mutual dependencies of those two tasks which a joint model can better learn than a pipeline

model. We can also relate it to better noise mitigation of jointed ET, compared to isolated models.[3]

Analysis of joint training. In this paragraph, we investigate the joint training in more detail. In particular, we evaluate different variants of it by combining relation extraction with other entity typing approaches: EntEmb and FIGMENT. For joint training with ET-EntEmb, we do not use the context for predicting the types of the relation arguments but only their embeddings. Then, we feed those embeddings into an MLP which computes a representation that we use for the type prediction. This corresponds to the EntEmb model presented in Table 3 (line 11). For joint training with ET-FIGMENT, we compute two different cost functions for entity typing: one for typing based on entity embeddings (see ET-EntEmb above) and one for typing based on an MLP context model. This does not correspond exactly to the FIGMENT model from Table 3 (line 12) which combines an entity embedding and MLP context model as a postprocessing step but comes close. In addition to those two baseline ET models, we also train a version in which both entity typing and relation extraction use EntEmb as their only input features. Figure 5 shows the PR curves for those models. The curve for the model that uses only entity embedding features for both entity typing and relation extraction is much worse than the other curves. This emphasizes the importance of our context model for RE (see also Figure 3), also in combination with joint training. Similarly, the curve for the model with EntEmb as entity typing component has more precision variations than the curves for the other models which use context features for ET. Thus, joint training does not help per se but it is important which models are trained together. The areas under the PR curves show the following model trends: joint with ET-FIGMENT $\approx$ joint as in Figure 1 $>$ joint with ET-EntEmb $>$ joint with ET-EntEmb and RE-EntEmb.

Most improved relations. To identify which relations are improved the most when entity types are integrated, we compare the relation specific F_1 scores of CNN, CNN+WEIGHTED and CNN+JOINT-TRAIN. 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 F_1, resp.). JOINT-TRAIN has the most positive impact on PPL.deceased_PER.place_of_death, ORG.ORG.place_founded and GOV.GOV_agency.jurisdiction (from 36.13 to 67.10, 42.38 to 58.51 and 62.26 to 70.41 resp.).

[3] On the joint dataset, joint training improves MAP for entity typing by about 20% compared to the best isolated model.

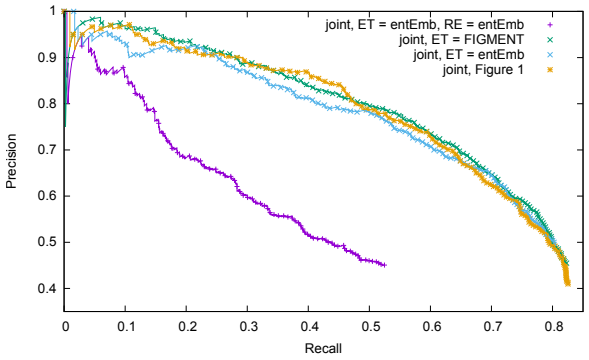

Figure 5: Variants of joint training

6 Conclusion

In this paper, we addressed different types of noise in two information extraction tasks: entity typing and relation extraction. We presented the first multi-instance multi-label methods for entity typing and showed that it helped to alleviate the noise from distant supervised labels. This is an important contribution because most of the current fine-grained entity typing systems are trained by distant supervision. Our best model sets a new state of the art in corpus-level entity typing. For relation extraction, we mitigated noise from using predicted entity types as features. We compared different pipeline approaches with each other and with our proposed joint type-relation extraction model. We observed that using type probabilities is more robust than binary predictions of types, and joint training gives the best results.

Acknowledgments

This work was supported by DFG (SCHU2246/8-2) and by a Google European Doctoral Fellowship granted to Heike Adel.

References

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

Diego, California, USA, June 12 - June 17, 2016. http://arxiv.org/abs/1603.05157.

Isabelle Augenstein, Andreas Vlachos, and Diana Maynard. 2015. Extracting relations between non-standard entities using distant supervision and imitation learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 747–757.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Irreflexive and hierarchical relations as translations. *CoRR*, abs/1304.7158.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1243–1249.

Lan Du, Anish Kumar, Mark Johnson, and Massimiliano Ciaramita. 2015. Using entity information from a knowledge base to improve relation extraction. In *Australasian Language Technology Association Workshop 2015*, pages 31–38.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*, pages 427–434. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. 2012. Link prediction in multi-relational graphs using additive models. In *Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data, Boston, USA, November 11, 2012*, pages 1–12.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany, August. Association for Computational Linguistics.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin, Ireland, August 23-29 2014*, pages 2107–2116.

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

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 515–525.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *World Wide Web Conference*, pages 271–280.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1825–1834.

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

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 74–84.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *TACL*, 1:367–378.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. *CoRR*, abs/1604.05525.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 455–465.

URL. 2016a. Ds_pcnns (piecewise cnn) code (kang liu). `http://lemurproject.org/clueweb12`.

URL. 2016b. Figment data set. `http://cistern.cis.lmu.de/figment`.

URL. 2016c. Lemur project: Clueweb. `http://lemurproject.org/clueweb12`.

URL. 2016d. Lemur project: Facc1. `http://lemurproject.org/clueweb12/FACC1`.

Andreas Vlachos and Stephen Clark. 2014. Application-driven relation extraction with limited distant supervision. In *Proceedings of the AHA! Workshop on Information Discovery in Text, Dublin, Ireland, August 23 2014*, pages 1–6.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1591–1601.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1366–1371.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 715–725.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1013–1023.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 79–84.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015,*

Lisbon, Portugal, September 17-21, 2015, pages 1753–1762.

Bibliography

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

Isabelle Augenstein, Andreas Vlachos, and Diana Maynard. Extracting relations between non-standard entities using distant supervision and imitation learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 747–757, 2015.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, 2015.

Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36:673–721, 2010.

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1798–1828, 2013.

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 *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human

knowledge. In *Proceedings of the International Conference on Management of Data*, pages 1247–1250, 2008.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Irreflexive and hierarchical relations as translations. In *Proceddings of the Workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*, 2013.

Jan A. Botha and Phil Blunsom. Compositional morphology for word representations and language modelling. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1899–1907, 2014.

Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 421–436. Springer, 2012.

Paul Buitelaar. Nlp in the ontology life-cycle. *eLearning4NLP Workshop*, 2007.

Yu Chen, You Ouyang, Wenjie Li, Dequan Zheng, and Tiejun Zhao. Using deep belief nets for chinese named entity categorization. In *Proceedings of the 2010 Named Entities Workshop*, pages 102–109, 2010.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878, 2015.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. A hybrid neural model for type classification of entity mentions. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1243–1249, 2015.

Cícero Nogueira dos Santos and Victor Guimarães. Boosting named entity recognition with neural character embeddings. *CoRR*, abs/1505.05008, 2015.

Cícero Nogueira dos Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1818–1826, 2014.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 626–634, 2015.

Lan Du, Anish Kumar, Mark Johnson, and Massimiliano Ciaramita. Using entity information from a knowledge base to improve relation extraction. In *Australasian Language Technology Association Workshop*, pages 31–38, 2015.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490, 2014.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 260–269, August 2016.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370, 2005.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. Context-dependent fine-grained entity type tagging. *CoRR*, abs/1412.1820, 2014.

Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

Sonal Gupta and Christopher D. Manning. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the 18th Conference on Computational Natural Language Learning*, pages 98–108, 2014.

GE Hinton. Distributed representations. *Technical Report at School of Computer Science at Carnegie Mellon University*, 1984.

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

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Associ-*

ation for Computational Linguistics: Human Language Technologies, pages 541–550, 2011.

Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. Link prediction in multi-relational graphs using additive models. In *Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data*, pages 1–12, 2012.

Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, 2015.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, 2014.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, pages 3294–3302, 2015.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. Named entity recognition with character-level models. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 180–183, 2003.

Arne Köhn. What's in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, 2015.

Siwei Lai, Kang Liu, Liheng Xu, and Jun Zhao. How to generate a good word embedding? *CoRR*, abs/1507.05523, 2015.

Yann Lecun and Yoshua Bengio. *Convolutional Networks for Images, Speech and Time Series*, pages 255–258. The MIT Press, 1995.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Effiicient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50, 1998.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, 2015.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2124–2133, 2016.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015a.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, 2015b.

Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *Proceedings of the 16th AAAI Conference on Artificial Intelligence*, 2012.

Xiao Ling, Sameer Singh, and Daniel Weld. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328, 2015c.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2107–2116, 2014.

Gideon S. Mann. Fine-grained proper noun ontologies for question answering. In *Proceedings of the 2002 Workshop on Building and Using Semantic Networks*, pages 1–7, 2002.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Paul McNamee and Hoa Trang Dang. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference*, pages 111–113, 2009.

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

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, 2013.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009.

Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of the 27th Annual Conference on Neural Information Processing*, pages 2265–2273, 2013.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30:3–26, 2007.

Arvind Neelakantan and Ming-Wei Chang. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525, 2015.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st World Wide Web Conference*, pages 271–280, 2012.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the 18th Conference on Computational Natural Language Learning*, 2014.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1 – 17, 1964.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378, 2016a.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pages 1825–1834, 2016b.

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 *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378, 2013.

Sascha Rothe and Hinrich Schütze. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. Ultradense word embeddings by orthogonal transformation. In *Proceddings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777, 2016.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 726–730, 2015.

Magnus Sahlgren. *The Word-Space Model*. PhD thesis, Stockholm University, 2006.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, 2015.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 69–74, 2016.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 926–934, 2013.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706, 2007.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1333–1339, 2015.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, 2012.

Michael Thelen and Ellen Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 214–221, 2002.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, 2015.

Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

BIBLIOGRAPHY

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Andreas Vlachos and Stephen Clark. Application-driven relation extraction with limited distant supervision. In *Proceedings of the AHA! Workshop on Information Discovery in Text*, pages 1–6, 2014.

Peng Wang, Jiaming Xu, Bo Xu, Cheng-Lin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 352–357, 2015.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1591–1601, 2014.

Zhigang Wang and Juan-Zi Li. Text-enhanced representation learning for knowledge graph. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1293–1299, 2016.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, 2013.

Yijun Xiao and Kyunghyun Cho. Efficient character-level document classification by combining convolution and recurrent layers. *CoRR*, abs/1602.00367, 2016.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the 30th Conference on Artificial Intelligence*, pages 2659–2665, 2016.

Yadollah Yaghoobzadeh and Hinrich Schütze. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725, 2015.

Yadollah Yaghoobzadeh and Hinrich Schütze. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 236–246, 2016.

Yadollah Yaghoobzadeh and Hinrich Schütze. Multi-level representations for fine-grained typing of knowledge base entities. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 578–589, 2017.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, 2016.

Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, 2010.

Limin Yao, Sebastian Riedel, and Andrew McCallum. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, pages 79–84, 2013.

Wenpeng Yin and Hinrich Schütze. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, 2015.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 291–296, 2015.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. HYENA: Hierarchical type classification for entity names. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1361–1370, 2012.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 2335–2344, 2014.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, 2015.

Xiang Zhang and Yann LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, 2015.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 29th Annual Conference on Neural Information Processing*, pages 649–657, 2015.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 267–272, 2015.

Zhi Zhong and Hwee Tou Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pages 78–83, 2010.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 427–434, 2005.

Curriculum Vitae

Education	
02/2014 – 07/2017	**Ph.D. Student** (CIS, LMU Munich, Germany) Research on Distributed Representations and Fine-grained Entity Typing
09/2010 – 01/2013	**MSc Student** (Sharif University of Technology, Iran) Specializations: Computer Engineering
09/2006 – 07/2010	**BSc Student** (University of Tehran, Iran) Specializations: Computer Engineering

Work Experience	
10/2016 - 01/2017	**Research Intern** (Intuit Inc., California, USA) Research on Ontology Learning
06/2015 – 09/2015	**Software Developer** (FANAP, Tehran, Iran) Developing and Maintaining an Insurance Software

Professional Activities	
Program Committee Member	ACL, EMNLP
Workshop Organization	SCLeM (Subword and Caracter LEvel Models in NLP) Workshop in EMNLP 2017, Copenhagen, Denmark, Sep. 2017