Journal of Artificial Intelligence Research 61 (2018) 835-862 Submitted 06/17; published 04/18

Corpus-Level Fine-Grained Entity Typing

Yadollah Yaghoobzadeh YADOLLAH.YAGHOOBZADEH@MICROSOFT.COM
Microsoft Research Montreal
Montreal, Canada &
Center for Information and Language Processing
LMU Munich, Munich, Germany

Heike Adel HEIKE.ADEL@IMS.UNI-STUTTGART.DE
Institute for Natural Language Processing
University of Stuttgart, Stuttgart, Germany &
Center for Information and Language Processing
LMU Munich, Munich, Germany

Hinrich Schütze INQUIRIES@CISLMU.ORG
Center for Information and Language Processing
LMU Munich, Munich, Germany

Abstract

Extracting information about entities remains an important research area. 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 computes scores based on global information of an entity and (ii) a context model that first evaluates the individual occurrences of an entity and then aggregates the scores.

Each of the two proposed models has 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 the name and contexts of entities on the three levels of entity, word, and character. We show that each level provides complementary information and a multi-level representation performs best. For the context model, we need to use distant supervision since there are no context-level labels available for entities. Distantly 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. We show the effectiveness of our models on a large entity typing dataset built from Freebase.

1. Introduction

Knowledge about entities is essential for natural language understanding (NLU). This knowledge includes facts about entities, such as their names, properties, relations and types. This data is usually stored in large-scale structures called knowledge bases (KBs) and therefore building and maintaining KBs is very important. Examples of such KBs are Freebase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008), YAGO (Suchanek, Kasneci, & Weikum, 2007) and Wikidata (Vrandečić & Krötzsch, 2014) . KB structure is usually 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.

Incompleteness is a crucial challenge for KBs because the world is changing – new entities are emerging, and existing entities are getting new properties. Therefore, there is always the 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. Most prior work tries to complete relations between entities. In contrast, the focus of this work is on the completion of entity types in KBs. For example, given a large corpus and the entity "Barack Obama" we need to find all his types including "person", "politician", and "author".

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 fine-grained 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 information about an entity from the corpus and its name, and then predicts the type based on that. We represent necessary information about the entity in a multi-level representation. This representation is global, meaning that it does not describe a specific context or mention of an entity but rather the aggregated information about it. One important source to compute the entity representations is the *contexts in which the entity is used*. We 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 (e.g., Wang, Zhang, Feng, & Chen, 2014; Yaghoobzadeh & Schütze, 2015; Yaghoobzadeh & Schütze, 2017). We refer to entity embeddings as *entity-level representations*. Entity-level representations are often uninformative for rare entities, so relying only on 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-level 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, Chen, Corrado, & Dean, 2013) and (ii) using subwords as well as contextual information of words, e.g., by Facebook's recently released FASTTEXT (Bojanowski, Grave, Joulin, & Mikolov, 2017).

The **context model** first scores each individual context of e as expressing type t or not. The final value of $m(e, t)$ is then computed based on the distribution of context scores. One difficulty of this model is that it is too expensive to label each entity context with its labels. Distant supervision (Mintz, Bills, Snow, & Jurafsky, 2009) is used to reduce the need for manually labelling contexts. Distant supervision assumes that if an entity has a type in the KB, then all contexts mentioning that entity express that type. However, this 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, Yao, & McCallum, 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) expresses the type given in the KB. Multi-instance multi-label (MIML) learning is a generalization of MI in which one bag can have several labels (Surdeanu, Tibshirani, Nallapati, & Manning, 2012).

Most MI and MIML methods are based on hand-crafted features. Recently, Zeng, Liu, Chen, and Zhao (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, Shen, Liu, Luan, and Sun (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 relation extraction. 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 four multi-label versions of MI. (i) *MIML-MAX* takes the maximum of instance scores for each label. (ii) *MIML-MEAN* takes the mean of instance scores for each label. (iii) *MIML-MAX-MEAN* takes the maximum and the mean of instance scores in training and testing, respectively. (iv) *MIML-ATT* applies, for each label, selective attention to the instances. We apply these MIML algorithms to fine-grained entity typing and show that MIML-ATT deals well with noise in this task and gives clear improvements for the distantly supervised models.

The global model is potentially more robust compared to the context model since it takes into account all the available information at once. In contrast, the context model has the advantage that it can correctly predict types for which there is only a small number of reliable contexts. For example, in a large corpus, it is likely to find a few reliable contexts indicating that "Barack Obama" is a best-selling author even though this evidence may be obscured in the global distribution because the vast majority of mentions of "Obama" does 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*.

The contributions in this paper include the following: (i) We address fine-grained entity typing by using text corpora for the application in knowledge base completion. (ii) We build a dataset for this task from Freebase entities and their fine-grained types. (iii) We introduce, implement and compare two types of models for the task, global and context models, and a joint model of them. (iv) We represent entities using novel distributed representations on the 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.

2. Related Work

Our task is *fine-grained entity typing*. Neelakantan and Chang (2015), Suzuki, Matsuda, Sekine, Okazaki, and Inui (2016) and Xie, Liu, Jia, Luan, and Sun (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, Bauer, Hoffart, Spaniol, & Weikum, 2012; Ling & Weld, 2012; Yogatama, Gillick, & Lazic, 2015; Del Corro, Abujabal, Gemulla, & Weikum, 2015; Shimaoka, Stenetorp, Inui, & Riedel, 2016; Ren, He, Qu, Voss, Ji, & Han, 2016; Rabinovich & Klein, 2017; van Erp & Vossen, 2017; Shimaoka, Stenetorp, Inui, & Riedel, 2017) 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 & Riloff, 2002; Gupta & 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) show 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, Tresp, & Kriegel, 2012; Bordes, Usunier, García-Durán, Weston, & Yakhnenko, 2013), while others use a combination of within-KB generalization and information extraction from text (Weston, Bordes, Yakhnenko, & Usunier, 2013; Socher, Chen, Manning, & Ng, 2013; Jiang, Tresp, Huang, & Nickel, 2012; Riedel, Yao, McCallum, & Marlin, 2013; Wang et al., 2014).

We also introduce methods for *noise mitigation in 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, Zhang, Ling, Zettlemoyer, & Weld, 2011; Zeng et al., 2015; Adel, Roth, & Schütze, 2016) and entity typing (e.g., Ling & Weld, 2012; Yogatama et al., 2015; Dong, Wei, Sun, Zhou, & Xu, 2015). To mitigate the noisy label problem, multi-instance (MI) learning has been introduced and applied in relation extraction (Riedel et al., 2010; Ritter, Zettlemoyer, Mausam, & Etzioni, 2013). Surdeanu et al. (2012) introduce 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) introduce 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) apply 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 in how entity relations and entity types are mentioned: expressing the entity relations is more likely to be explicit than entity types in many text genres. This means we usually need to look at many entity contexts for reliable type predictions. This can influence the choice of MIML algorithms since some of them just pick one context (instance) for prediction. Our experimental results confirm this hypothesis.

3. Motivation and Background

In this section, we provide motivation and background information for our work, including Freebase, incompleteness of knowledge bases, entity linking and FIGER types.

3.1 Freebase

Large scale knowledge bases (KBs) like Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and Wikidata (Vrandečić & Krötzsch, 2014) are designed to store world knowledge. KB structure is usually graph-based and with different schemas. 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., "Arnold Schwarzenegger" is a "person", "actor", "politician", "sports_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., the "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 a property defined for the type "politician".

3.2 Incompleteness of Knowledge Base

Even though Freebase is the largest publicly available KB of its kind, it still has significant coverage problems. For example, 78.5% of persons in Freebase do not have a *nationality* (Min, Grishman, Wan, Wang, & Gondek, 2013), or in our test set, 12% of persons do not have a finer grained type.

This is unavoidable partly because Freebase is user-generated and partly because the world changes constantly 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 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 is arguably the main repository for the type of knowledge represented in KBs, and thus it is reasonable to attempt completing them based on text. There is a significant body of work on corpus-based methods for extracting knowledge from text. However, most of the work has addressed relation extraction, and 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 our task, entity typing. Indeed, some recent work demonstrates large improvements when entity typing and linking are jointly modeled (Ling, Singh, & Weld, 2015b; Durrett & Klein, 2014). However, there are constraints that are important for high-performance entity linking but are of little relevance to entity typing. For example, there is large body of 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 only address entity typing in this paper and consider entity linking as an independent module that provides contexts of entities for FIGMENT. (A similar problem definition is used in relation extraction, cf., Zeng et al., 2015; Lin et al., 2016.) More specifically, we build FIGMENT on top of the output of an existing entity linking system and use FACC1 (Gabrilovich, Ringgaard, & Subramanya, 2013), an automatic Freebase annotation of ClueWeb (ClueWeb-URL, 2012). 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. However, there are types which are too fine-grained, such as "Vietnamese urban district". 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 & 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" (for a complete list of FIGER types, see Ling & Weld, 2012.). We use "type" to refer to FIGER types in the rest of the paper.

4. Global, Context, and Joint Models

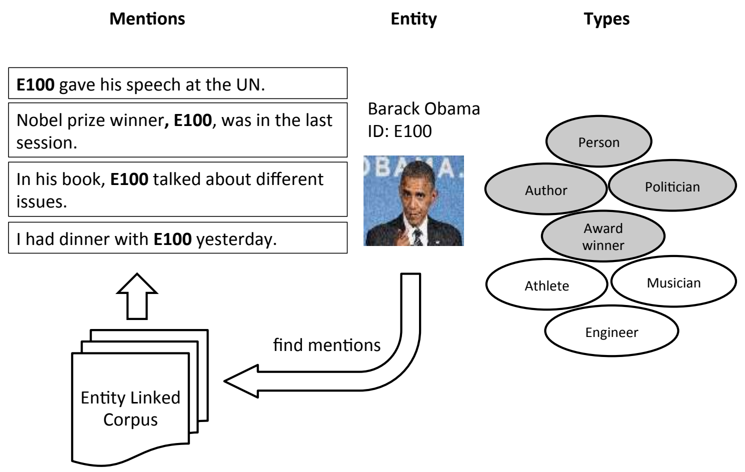
Mentions
Entity
Types
E100 gave his speech at the UN.
Nobel prize winner, E100, was in the last session.
In his book, E100 talked about different issues.
I had dinner with E100 yesterday.
Barack Obama
ID: E100
Person
Author
Politician
Award winner
Athlete
Musician
Engineer
find mentions
Entity Linked Corpus

Figure 1: An example for corpus-level fine-grained entity typing. Given Barack Obama with ID of E100 from a KB, and an entity linked corpus with some mentions of E100, the task is to predict the correct fine-grained types of E100 : "person", "politician", and "author", and "award-winner".

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 *corpus-level fine-grained entity typing*: predicting whether entity e is a member of type t or not. We show an example diagram of our task in Figure 1. 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 *new* or *unknown entities* – i.e., entities not covered by the KB. For new or unknown entities, an entity linking system would be necessary that identifies and clusters entity mentions. Examples for those systems are the ones participating in TAC KBP (McNamee & Dang, 2009).

In this section, we introduce our two models, global and context, as well as their combination (joint model). In Table 1, we provide a brief overview of the models. Details are given in the following sections.

Notations and definitions. Lowercase letters (e.g., e) refer to variables. Bold lowercase letters (e.g., $\mathbf{e}$) are vectors and bold uppercase letter (e.g., $\mathbf{W}$) are matrices. We define BCE, the binary cross entropy between two variables, as follows where y is a binary variable and $\hat{y}$ is a real valued variable between 0 and 1.

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

Global Model	Context Model
$P_{\text{GM}}(t\|e) = \mathit{score}(\mathbf{e})$	MIML-ATT: $P_{\text{CM}}(t\|e) = \mathit{score}(\mathbf{a_t})$ $\mathbf{a_t} = \sum_i \alpha_i \mathbf{c_i}$
Entity-level: $\mathbf{e} =$ distributed embedding of e	
Word-level: $\mathbf{e} = g($words of the name of $e)$	MIML-MAX/MEAN: $P_{\text{CM}}(t\|e) = \text{max/mean}_i P(t\|c_i)$ $P(t\|c_i) = \mathit{score}(\mathbf{c_i})$
Character-level: $\mathbf{e} = g($characters of the name of $e)$	$\mathbf{c_i} = g($words of $c_i)$
Joint Model $P(t\|e) = \frac{1}{2}\big(P_{\text{GM}}(t\|e) + P_{\text{CM}}(t\|e)\big)$	

Table 1: An overview of our global, context and joint models. g is a neural network function, e.g., a feed-forward neural network. e is an entity. c_i is the i-th context of e. α_i is a scalar. In our setup, we take the most frequent mention of e in the corpus as the name of e. MIML-ATT and MIML-MAX/MEAN are our different multi-instance multi-label models.

4.1 Global Model

In the global model, we learn $P(t|e)$ by first learning a distributed representation $\mathbf{e}$ of entity e. Then, a multi-layer perceptron (MLP) with one hidden layer is applied with the output layer of size $|T|$. The schematic diagram of the MLP is shown in Figure 2. Unit t of this layer outputs the probability for type t:

$$P_{\text{GM}}(t|e) = \sigma\Big(\mathbf{W}_{\text{out}} f\big(\mathbf{W}_{\text{in}}\mathbf{e}\big)\Big) \quad (1)$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{h \times d}$ is the weight matrix from $\mathbf{e} \in \mathbb{R}^d$ 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: $\sigma(x) = 1/(1 + e^{-x})$ that converts the value x to a value in $[0, 1]$. Our objective is binary cross entropy summed over types:

$$L = \sum_t \text{BCE}(\mathbf{y}_t, \mathbf{p}_t)$$

where $\mathbf{y}_t$ is the truth and $\mathbf{p}_t$ the prediction for type t.

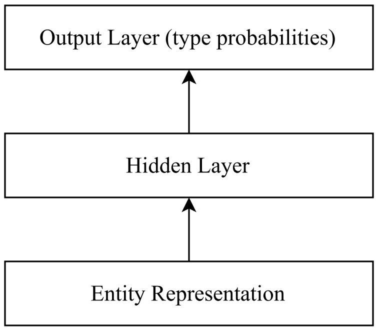

Figure 2: In the global model, a multi-layer perceptron is applied on the entity representation learned from contexts or/and name of the entity.

The key difficulty when computing $P(t|e)$ 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.

4.1.1 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): entities with similar meanings tend to have similar contexts. Thus, we can learn a d dimensional embedding $\mathbf{e}$ 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 $\mathbf{e}$. However, the bag-of-word context is difficult to distinguish for pairs of types that have similar context words like (“restaurant”, “food”) and (“author”, “book”). This suggests that *using order aware embedding models is important for entities*. Therefore, we apply Ling, Dyer, Black, and Trancoso’s (2015a) extended version of SKIP, Structured SKIP (SSKIP). It incorporates the order of context words into the objective and outperforms SKIP in entity typing (Yaghoobzadeh & Schütze, 2016; Yaghoobzadeh & Schütze, 2017).

4.1.2 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 constitute the entity name $\mathbf{e} = 1/n \sum_{i=1}^{n} \mathbf{w}_i$ where $\mathbf{w}_i$ is the embedding of the i^{th} word of an entity name of length n. We consider the canonical name of an entity in the KB to compute this representation. 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, $\mathbf{w}$, 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., 2017), are subword-level. Based on this, we consider and evaluate *word-level WLR (WWLR)* and *subword-level WLR (SWLR)* in this paper.[1]

4.1.3 CHARACTER-LEVEL REPRESENTATION

For computing the *character-level representation* (CLR), we design models that 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 or three 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 the type of phoneme encoded (consonant/vowel) or similarity in the case (lower/upper). The output of the lookup layer for an entity name is a matrix $\mathbf{C} \in \mathbb{R}^{l \times d_c}$ 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 two architectures to produce character-level representations in this paper: fully connected feed-forward (FF) and convolutional neural networks (CNNs).

FF simply concatenates all rows of matrix $\mathbf{C}$; thus, $\mathbf{e} \in \mathbb{R}^{d_c.l}$.

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

$$\mathbf{m}[i] = \mathrm{g}(\mathbf{C}_{[:,i:i+w-1]} \odot \mathbf{H} + b)$$

where g is the rectifier function ($g(x) = \max(0, x)$), b is the bias, $\mathbf{C}_{[:,i:i+w-1]}$ are the columns i to $i + w - 1$ of $\mathbf{C}$, $1 \leq w \leq k$ are the window widths we consider and $\odot$ is the Frobenius product. Max pooling then gives us one feature for each filter. The concatenation of all these features is our representation: $\mathbf{e} \in \mathbb{R}^n$. An example CNN architecture is shown in Figure 3.

4.1.4 MULTI-LEVEL REPRESENTATIONS

Our different levels of representations can provide 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

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 level of granularity with respect to the entities and not words.

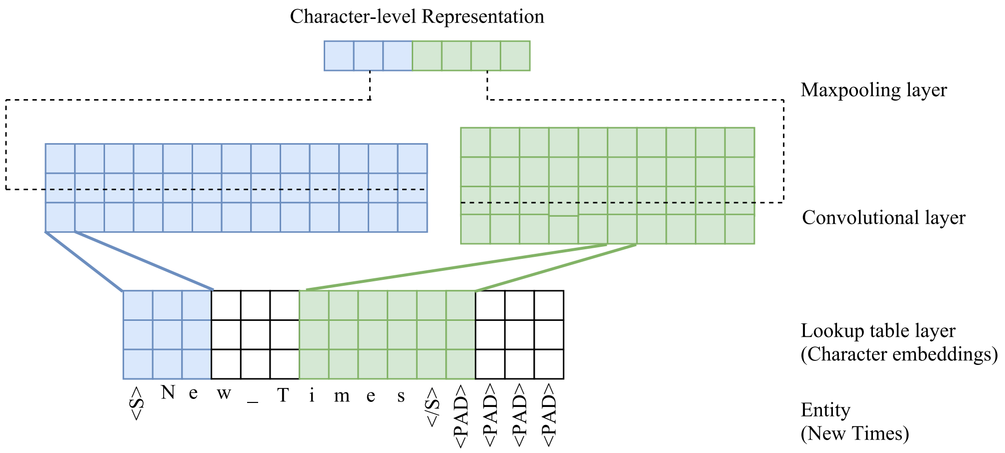

Figure 3: Example architecture for the character-level CNN with max pooling. The input is "New Times". Start and end symbols are appended to the input and it is padded by the pad symbol to a fixed size (here it is 15). Character embedding size is three. There are three filters of width 3 and four filters of width 6.

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, their constituting words 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 might allow 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 the contexts on an entity, 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 **multi-level** 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 the other levels. The schematic diagram of our multi-level representation is shown in Figure 4.

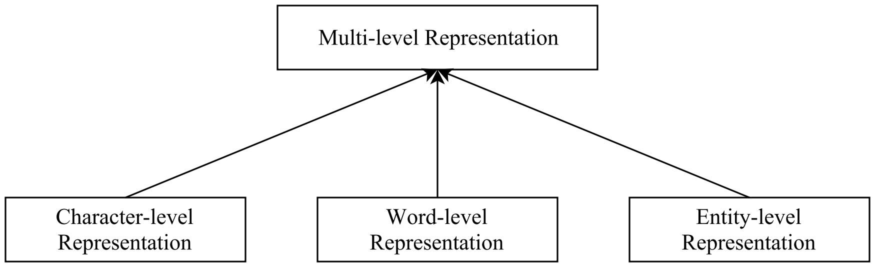
Multi-level Representation
Character-level Representation
Word-level Representation
Entity-level Representation

Figure 4: Multi-level representations of entities are the concatenation of a subset of character, word and entity-level representations. These representations are then given to the MLP in Figure 2.

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 anyone". 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. In the following, we describe our MIML algorithms that work on the contexts representations to compute $P(t|e)$.

4.2.1 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:

$$P(t|c_i) = \sigma(\mathbf{w}_t^T \mathbf{c}_i + b_t) \tag{2}$$

where $\mathbf{w}_t \in \mathbb{R}^h$ 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:

$$L(\theta) = \sum_c \sum_t \text{BCE}(\mathbf{y}_t, P(t|c)) \tag{3}$$

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

$$P_{\text{CM}}^{\text{pred}}(t|e) = \frac{1}{q} \sum_{i=1}^{q} P(t|c_i) \tag{4}$$

4.2.2 MULTI-INSTANCE MULTI-LABEL (MIML)

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:

$$P_{\text{CM}}(t|e) = \max_{1 \leq i \leq q} P(t|c_i) \tag{5}$$

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**.

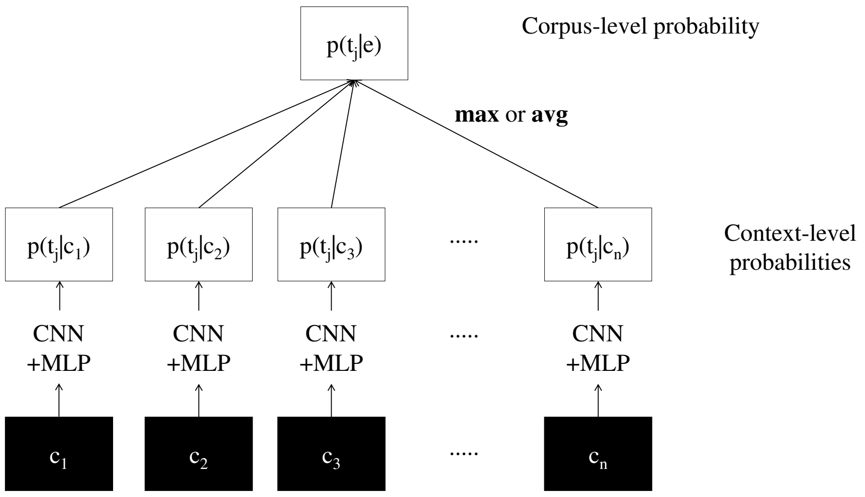

Figure 5: The context model with distant supervision, MIML-MAX, or MIML-AVG. It uses maximum or average to compute corpus-level type probabilities for an entity from its context-level probabilities. In the case of multi-instance multi-label learning (MIML-MAX or MIML-AVG), the aggregation function (max or avg) is applied in both training and prediction, otherwise (i.e., distant supervision) it is only applied at the prediction time.

Distant supervision makes the assumption that each instance is relevant. In MIML-MAX, we correct this by changing this assumption to "one instance is relevant". This means that MIML-MAX selects the most confident context for prediction of each type. 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. **Averaging** the type probabilities of all contexts of entity e is the simplest way of doing this (cf. Eq. 4):

$$P_{\text{CM}}(t|e) = \frac{1}{q} \sum_{i=1}^{q} P(t|c_i) \tag{6}$$

We call this approach **MIML-AVG**. We also propose a combination of the maximum and average, which uses MIML-MAX (Eq. 5) in training and MIML-AVG (Eq. 6) in prediction. We call this approach **MIML-MAX-AVG**. An illustration of these MIML approaches is depicted in Figure 5.

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 by Lin et al. (2016). To compute the type probability for e, we define:

$$P_{\text{CM}}(t|e) = \sigma(\mathbf{w_t}^T \mathbf{a_t} + b_t) \tag{7}$$

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

$$\mathbf{a_t} = \sum_i \alpha_{i,t} \mathbf{c_i} \tag{8}$$

where $\alpha_{i,t}$ is the attention score 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. $\alpha_{i,t}$ is defined as:

$$\alpha_{i,t} = \frac{\exp(\mathbf{c_i}^T \mathbf{M} \mathbf{t})}{\sum_{j=1}^{q} \exp(\mathbf{c_j}^T \mathbf{M} \mathbf{t})} \tag{9}$$

where $\mathbf{M} \in \mathbb{R}^{h \times d_t}$ is a weight matrix that measures the similarity of $\mathbf{c}$ and $\mathbf{t}$. $\mathbf{t} \in \mathbb{R}^{d_t}$ is the representation of type t. Figure 6 illustrates MIML-ATT.

Table 2 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,

$$L(\theta) = \sum_e \sum_t \text{BCE}(\mathbf{y}_t, P(t|e)) \tag{10}$$

In contrast to the loss function of distant supervision in Eq. 3, which iterates over all *contexts*, we iterate over all *entities* here.

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

Table 2: Different MIML algorithms for entity typing, and the aggregation function they use to get corpus-level probabilities.

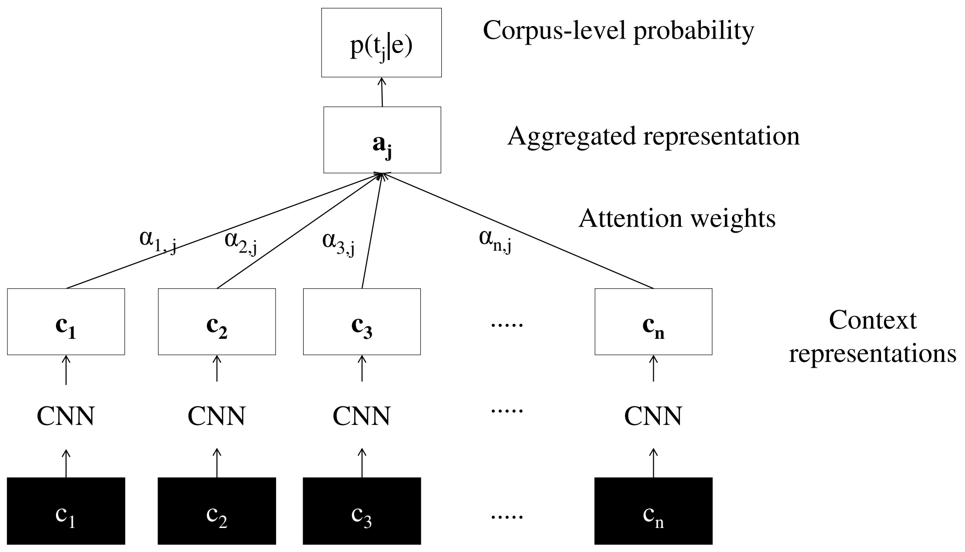

Figure 6: The context model with MIML-ATT. The corpus-level probability of a type is computed from type specific aggregated representation of the entity, which is computed using attention weights of the type on each of entity contexts.

4.2.3 CONTEXT REPRESENTATION

To produce a high-quality context representation **c**, we use two neural network architectures, fully connected feed-forward (FF) and convolutional neural networks (CNNs). The first layer of both architectures is a *lookup table* that maps each word in c to an embedding of size d. If there is another training entity in the context, we replace it with its notable type to get better generalization. The output of the lookup layer is a matrix $\mathbf{E} \in \mathbb{R}^{d \times s}$ (the embedding layer), where s is the context size (a fixed number of words). **FF** then simply concatenates all rows of matrix **E**; thus, $\phi(c) \in \mathbb{R}^{d*s}$.

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

$$\mathbf{m}[i] = g(\mathbf{E}_{:,i:i+w-1} \odot \mathbf{H}) \tag{11}$$

where g is the rectifier function, $\odot$ is the Frobenius product, $\mathbf{E}_{:,i:i+w-1}$ 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. We apply the CNN to the left and right context of the entity mention and the concatenation $\phi(c) \in \mathbb{R}^{2n}$ is the output of the CNN layer.

$\phi(c)$ is then fed into a multi-layer perceptron (MLP) for both architectures (FF and CNN) to get the final context representation $\mathbf{c} \in \mathbb{R}^{h}$:

$$\mathbf{c} = \tanh\Big(\mathbf{W}_h \phi(c)\Big) \tag{12}$$

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 contexts in which the entity occurs. This is also the way a human would ordinarily do entity typing for an unknown entity: they 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.

One notable weakness of CM is that a large proportion of contexts does not contain sufficient information to infer all types of the entity. For example, based on our distantly supervised training data, we label every context of "Obama" with "politician" and "author" and Obama's other types in the KB. Multi-instance learning algorithms mitigate this weakness to some degree, but at the extra cost of more model parameters.

The main strength of GM is that it bases its decisions on the entire evidence available in the corpus. This makes it more robust. Also, GM models the problem as a supervised task and its dataset consists of labeled entities with their types. This makes it more efficient compared to the distantly supervised approach used in CM.

The disadvantage of GM is that it is less likely to work well for non-dominant types of an entity that might be swamped by dominant types. For example, 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. Also for an unknown entity, the embedding learning model needs to be retrained to get the entity-level representation, which makes the GM application less efficient for new or emerging entities.

Since GM and CM models are complementary, we expect a combined model to work better. We test this hypothesis for the simplest possible joint model (JM), which averages the type probabilities of the two individual models for each entity as:

$$P_{\text{JM}}(t|e) = \frac{P_{\text{GM}}(t|e) + P_{\text{CM}}(t|e)}{2} \tag{13}$$

5. Experimental Setup and Results

In this section, we first describe the different setups we use to do our experiments. Next, we present the results of our models followed by further analysis.

5.1 Setup

In this section, we explain the datasets, evaluation metrics and other setups we follow to do our experiments. We also describe the baselines, which we compare our models against.

5.1.1 CORPUS

We select a subset of about 7.5 million web pages, taken from the first segment of ClueWeb12 (ClueWeb-URL, 2012), 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.

5.1.2 ENTITY DATASETS

We consider all entities in the corpus whose notable types can be mapped to one of the 113 FIGER types, based on the mapping provided by FIGER. 750,000 such entities form our set of entities. 10 out of 113 FIGER types have no entities in this set.[2] We then select a subset of 200,000 entities in a process which is explained by Yaghoobzadeh and Schütze (2015). We split the entities into train (50%), dev (20%) and test (30%) sets. The average and median number of FIGER types of the training entities are 1.8 and 2, respectively.[3]

5.1.3 CONTEXT SAMPLING

For the context model, 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 also 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 t.

Next, 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.

5.1.4 BASELINES

We extend our previous work in corpus-level fine-grained entity typing. JOINT-BASE1, JOINT-BASE2 and ELR+SWLR+CLR(CNN) correspond to the best models by Yaghoobzadeh and Schütze (2015), Yaghoobzadeh, Adel, and Schütze (2017) and Yaghoobzadeh and Schütze (2017), respectively. We also add some hand-crafted feature-based baselines.

We implement the following two feature sets from the literature as a *hand-crafted baseline* for our character-level and word-level GM 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 & Weld, 2012), character n-grams, $1 \leq n \leq n_{\max}, n_{\max} = 5$ (we also tried $n_{\max} = 7$, but results were worse on dev) and normalized character n-grams: lowercased, digits replaced by “7”, punctuation replaced by “.”. These features are represented as a sparse binary vector $\mathbf{e}$ that forms the input to the architecture in Figure 2.

The other baseline is using an existing mention-level entity typing system, *FIGER* (Ling & 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

2. 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.
3. The entity datasets are available at `http://cistern.cis.lmu.de/figment`.

entities, so FIGER does not need to recognize entities.[4] 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). The average of scores gave us the best result on dev. We call this baseline AGG-FIGER: aggregated version of FIGER (Ling & Weld, 2012).

5.1.5 EVALUATION METRICS

Evaluation of multi-label classification is more complicated than the common single-label setting because each example can belong to several labels at the same time. To address this, we use two types of metrics for evaluation: example-based and label-based. In the example-based measures, an average value is computed based on the evaluation values of each test example. In the label-based measures, the performance of a classifier is evaluated for each label and then averaged over all labels. Each measure can further be categorized into ranking or classification. In the ranking measures, the evaluation shows how well the models rank labels for examples (for the example-based measures), or, rank examples for labels (for the label-based measures).

Since our problem is multi-label classification, we adapt some of the measures from the literature. In our setting, entities are examples and types are the labels. The classification metrics measure 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. The decisions are then compared to the gold KB information, with the assumption that KB information is incomplete. The assignment decision is made based on thresholds, one per type, for each $P(t|e)$. We select the threshold that maximizes F_1 of entities assigned to the type on dev. In the following, we define the metrics.

Example-based metrics: Two ranking and three classification measures are considered for the example-based evaluation. The ranking measures are (i) precision at 1 (P@1): percentage of entities whose top-ranked type is correct; (ii) breakeven point (BEP, Boldrin & Levine, 2008): F_1 at the point in the ranked list at which precision and recall have the same value. The classification 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.

Label-based metrics: The ranking measures are (i) mean average precision (MAP): mean of the averaged precision of each type; (ii) average precision at k (P@k): average of each type's precision in the top k ranked entities. The classification measure is: (i) type macro average F_1: F_1 of entities assigned to a type, averaged over types.

5.1.6 DISTRIBUTIONAL EMBEDDINGS

For WWLR and ELR, we use the Structured SkipGram (SSKIP) model in WANG2VEC (Ling et al., 2015a) to learn the 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 million types.

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

We run SSKIP with the setup (100 dimensions, 10 negative samples, window size 5, and word frequency threshold of 100)[5] on this corpus to learn embeddings for words, entities and FIGER types. For SWLR, we use FASTTEXT (Bojanowski et al., 2017) 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.

5.1.7 Hyperparameter Values

Our hyperparameter values are optimized on dev. We use AdaGrad (Duchi, Hazan, & Singer, 2011) and minibatch training. For each experiment, we select the best model on dev. The values of our hyperparameters are shown in Table 7 in the appendix.

5.2 Results

We evaluate our different models using the mentioned measures. For the global model, we explore different entity representations including multi-level ones as described in Section 4.1. For the context model, we analyze the performance of the baseline AGG-FIGER and different models including the mentioned MIML methods on two architectures (FF and CNN) as described in Section 4.2. For the joint model, we show the combination of our best global and context models, as well as the combination of our baseline models.

Results for the **example-based measures** on the test entities for all (about 60,000 entities), head (frequency $>$ 100; about 12,200) and tail (frequency $<$ 5; about 10,000) are shown in Table 3. Each row represents one of the models. If not mentioned explicitly, the micro F_1 (Micro in Table 3) is the measure we talk about when comparing models.

Line 1 is the most frequent baseline, which assigns "person" to each entity. Lines 2-12 are different **global models** (GMs). For the character-level representation of entities, CNN (line 4) works clearly better than FF (line 3). This was expected as CNNs are good architectures to find position-independent local features for classification. NSL, the ngram baseline, works better than CNN in P@1 and BEP for all entities and macro F_1 for the tail entities. But for all other measures, CNN is better than NSL. CNN is learning the features automatically and this is another benefit of CNN over NSL. In word-level models (lines 5-8), SWLR (line 7) is better than WWLR (line 6) on the tail entities and worse on the head entities. Both are better than the BOW baseline, because BOW cannot deal well with sparseness. SWLR and WWLR are better than all CLR models (lines 2-4) because of their access to the word embeddings trained on the whole corpus. SWLR (line 7) has access to the subword information, and therefore has an embedding for each word, resulting in better embeddings for the tail entities. Since SWLR is better than WWLR for acc, we pick SWLR as our best WLR model. SWLR+CLR(CNN) (line 8) improves SWLR (line 7) by around 2%, implying that character-level representation add complementary information to the word-level models.

The entity-level representation, ELR (line 9), is the most important source of information for entities. Figure 7 provides a t-SNE (Van der Maaten & Hinton, 2008) visualization of ELR embeddings of a subset of the entities in our dataset. Different colors denote different entity types. The figure shows that entities of the same type are clustered together. The results confirm this: ELR is clearly better than character-level and word-level representations (line 9 $>$ lines 1-8). Adding CLR

5. The threshold does not apply for MIDs.

			all entities					head entities					tail entities				
			P@1	BEP	acc	mic	mac	P@1	BEP	acc	mic	mac	P@1	BEP	acc	mic	mac
global model (GM)	1	MFT	.101	.406	.000	.036	.036	.111	.406	.000	.040	.039	.097	.394	.000	.032	.032
	2	CLR(NSL)	**.643**	**.690**	.167	.447	.438	.615	.665	.154	.438	.425	.633	.696	**.180**	.439	**.437**
	3	CLR(FF)	.552	.618	.032	.345	.313	.530	.592	.032	.351	.313	.543	.621	.031	.330	.304
	4	CLR(CNN)	.628	.679	**.170**	**.471**	**.431**	**.598**	**.649**	**.166**	**.474**	**.423**	**.618**	**.685**	**.180**	**.458**	.429
	5	BOW	.560	.636	.080	.311	.370	.512	.591	.075	.301	.354	.566	.654	.088	.318	.376
	6	WWLR	.691	.727	.168	.553	.500	**.767**	.785	.246	.634	.610	.635	.684	.125	.497	.444
	7	SWLR	.702	.733	.178	.552	.525	.761	.782	.238	.625	.599	.651	.696	.150	.509	.488
	8	SWLR+CLR(CNN)	**.710**	**.754**	**.248**	**.578**	**.539**	.754	**.795**	**.312**	**.658**	**.618**	**.669**	**.729**	**.230**	**.529**	**.499**
	9	ELR	.851	.890	.494	.781	.740	.901	.922	.541	.831	.802	.732	.802	.381	.648	.581
	10	ELR+CLR(CNN)	.873	.905	.538	.804	.766	.906	.928	**.569**	.840	.812	.784	.841	.453	.713	.648
	11	ELR+SWLR	.877	.907	.532	.804	.769	.906	.856	.556	.836	.810	.802	.856	.466	.727	.668
	12	ELR+SWLR+CLR(CNN)	**.881**	**.911**	**.548**	**.812**	**.776**	**.907**	**.929**	.567	**.841**	**.813**	**.812**	**.862**	**.484**	**.738**	**.678**
context model (CM)	13	FF	.753	.791	.341	.697	.665	.781	.806	.415	.757	.722	.639	.705	.151	.529	.493
	14	FF+MIML-MAX	.757	.806	.230	.602	.562	.777	.809	.063	.518	.537	.660	.738	.210	.479	.345
	15	FF+MIML-MAX-AVG	.763	.807	.369	.721	.686	.774	.803	.442	.774	.745	.668	.741	.189	.568	.519
	16	FF+MIML-AVG	.779	.826	.369	.714	.685	.794	.830	.425	.760	.731	.674	.749	.204	.566	.531
	17	FF+MIML-ATT	**.820**	**.855**	**.403**	**.730**	**.701**	**.874**	**.889**	**.454**	**.781**	**.767**	**.681**	**.757**	**.283**	**.596**	**.553**
	18	CNN	.784	.820	.394	.722	.693	.818	.840	.461	.773	.747	.657	.726	.210	.563	.523
	19	CNN+MIML-MAX	.786	.825	.262	.622	.584	.811	.831	.084	.535	.561	.678	.752	.227	.497	.357
	20	CNN+MIML-MAX-AVG	.799	.834	.417	.743	.708	.818	.839	.484	.792	.839	.687	.757	.260	.598	.541
	21	CNN+MIML-AVG	.808	.847	.418	.735	.711	.829	.856	.472	.777	.757	.693	.763	.257	.592	.558
	22	CNN+MIML-ATT	**.837**	**.869**	**.460**	**.753**	**.730**	**.894**	**.903**	**.504**	**.796**	**.792**	**.699**	**.771**	**.335**	**.626**	**.584**
	23	AGG-FIGER	**.811**	**.847**	**.440**	**.740**	**.686**	**.843**	**.882**	**.530**	**.815**	**.763**	**.738**	**.784**	**.322**	**.627**	**.579**
	24	JOINT-BASE1	.857	.889	.507	.789	.746	.902	.920	.558	.836	.807	.735	.80	.380	.662	.591
	25	JOINT-BASE2	.876	.904	.534	.803	.769	**.923**	**.937**	.585	**.848**	**.830**	.756	.820	.415	.685	.622
	26	JOINT	**.885**	**.911**	**.563**	**.819**	**.785**	.912	.929	**.586**	**.848**	.820	**.812**	**.858**	**.487**	**.747**	**.694**

Table 3: Example-based measures: P@1, BEP, acc (accuracy), mic (micro F_1) and mac (macro F_1) on test for all, head and tail entities. Largest numbers in each column for each section separated with dotted lines are in bold font.

or SWLR to ELR (lines 10-11), improves ELR (line 9), especially for tail entities (around 7% in micro F_1). This demonstrates that for rare entities, contextual information is often not sufficient for an informative representation; hence, name features are helpful. Combination of all these three levels (line 12) is the best GM. This confirms our intuition that the levels are complementary.

The results of **context models** (CMs) are presented in lines 13-23. CNN (line 18) is better than FF (line 13); this result for context models mirrors the result we earlier obtained for global models (line 4 vs. line 3). Lines 14-17 and 19-22 show the results of different MIML algorithms for FF and CNN, respectively. The order of MIML methods is consistent in both FF and CNN: ATT $>$ MAX-AVG $>$ AVG $>$ MAX.

MAX (lines 14 and 19) is worse than its basic distantly supervised model version (lines 13 and 18). 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 is a difference between corpus evidence about the types of entities on the one hand and about relations between entities on the other. Related to this, MAX-AVG (lines 15 and 20) which averages the type probabilities at prediction time improves

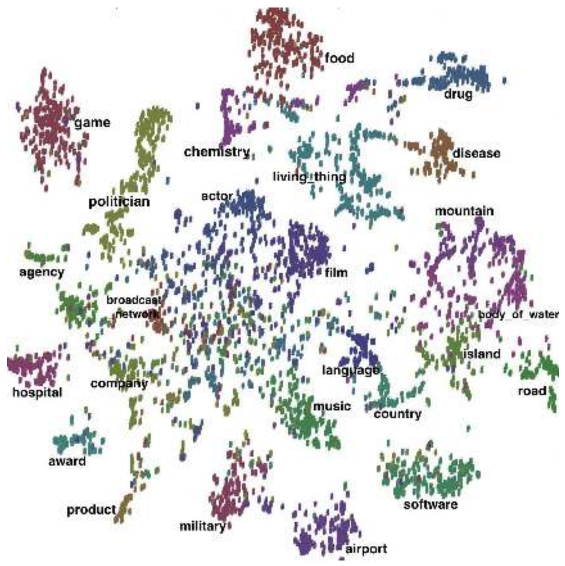

Figure 7: t-SNE result of entity-level representations.

MAX (lines 14 and 19) by a large margin. Averaging the context probabilities seems to be a way to smooth the entity type probabilities. MAX-AVG (lines 15 and 20) models are also better than the corresponding models with AVG (lines 16 and 21) which train and predict with averaging. This is due to the fact that AVG gives equal weights to all context probabilities in both training and prediction. ATT (lines 17 and 22) 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) consistently improves the results of both FF and CNN models.

In line 23, we show the results of AGG-FIGER. Compared to the neural network context models (lines 13-22), it is worse than CNN+MIML-ATT (line 22) in general, but better on head entities. AGG-FIGER (line 23) has access to the features extracted from the mentions of entities as well. This gives it more information than our context models have which only use the contexts of entities. AGG-FIGER is trained using distant supervision and its results could be improved by using our MIML methods. We leave this for future work. Nevertheless, all GM models with ELR (lines 9-12) are clearly better than CM models (lines 13-23) and this shows the effectiveness of entity-level embeddings for corpus-level entity typing.

Finally, lines 24-26 show the results for three joint models. JOINT-BASE1 (line 24) is the joint model of the GM model in line 9 (ELR) and the CM model in line 13 (FF) and, as we already mentioned, JOINT-BASE1 corresponds to our best model in Yaghoobzadeh and Schütze (2015). JOINT-BASE2 (line 25) is the joint model of the GM model in line 9 (ELR) and the CM model in line 22 (CNN+MIML-ATT). JOINT-BASE2 corresponds to our best model in Yaghoobzadeh et al. (2017). JOINT (line 26) is joining the best GM (line 12) and the best CM (line 22) and is the best in all of the measures, on the whole dataset, across all of the models. For the head entities, JOINT-BASE2 is slightly better. All joint models are better than their single counterparts. This confirms our intuition that GM and CM have complementary information. Compared to JOINT-BASE1, JOINT-BASE2 improves CM by using CNN instead of FF and applying MIML. Compared

				all types		head types		tail types	
			MAP	Macro F_1	P@50	Macro F_1	P@50	Macro F_1	P@50
global model	1	MFT	.018	.032	.016	.172	.084	.002	.001
	2	CLR(NSL)	**.277**	.320	.506	**.448**	.771	.203	.241
	3	CLR(FF)	.153	.194	.515	.374	.639	.115	.131
	4	CLR(CNN)	.274	**.325**	**.515**	.439	**.784**	**.252**	**.276**
	5	BOW	.246	.315	.504	**.427**	.759	.233	.272
	6	WWLR	.356	.395	.619	.520	.916	.308	.367
	7	SWLR	.374	.412	.627	.527	**.925**	**.339**	**.384**
	8	SWLR+CLR(CNN)	**.392**	**.420**	**.640**	**.530**	.908	.322	.375
	9	ELR	.631	.599	.788	.773	.988	.465	.527
	10	ELR+SWLR	.671	.632	.813	**.792**	**.993**	.500	.579
	11	ELR+CLR(CNN)	.679	.646	**.824**	.789	.992	**.524**	**.597**
	12	ELR+SWLR+CLR(CNN)	**.685**	**.650**	.820	.798	.989	.523	.593
context model	13	FF	.434	.468	.561	.688	.829	.332	.362
	14	FF+MIML-MAX	.432	.422	.667	.602	**.944**	.314	.398
	15	FF+MIML-MAX-AVG	.489	.510	.641	.704	.883	.363	.422
	16	FF+MIML-AVG	.470	.499	.595	.700	.853	.349	.400
	17	FF+MIML-ATT	**.514**	**.530**	**.678**	**.712**	.915	**.40**7	**.452**
	18	CNN	.478	.507	.603	. 709	.856	.367	.417
	19	CNN+MIML-MAX	.466	.450	.697	.618	**.967**	.358	.454
	20	CNN+MIML-MAX-AVG	.542	.552	.691	.725	.895	.425	.494
	21	CNN+MIML-AVG	.509	.534	.625	.723	.843	.394	.453
	22	CNN+MIML-ATT	**.561**	**.561**	**.723**	**.736**	.951	**.436**	**.496**
	23	AGG-FIGER	.59	.592	.756	.705	.909	.495	.570
	24	JOINT-BASE1	.641	.611	.794	.779	.961	.479	.547
	25	JOINT-BASE2	.685	.637	**.830**	.790	**.991**	.515	.597
	26	JOINT	**.703**	**.658**	.826	**.805**	.969	**.527**	**.607**

Table 4: Label-based measures for all, head and tail types. Largest numbers in each column for each section separated with dotted lines are in bold font.

to JOINT-BASE2, JOINT improves GM by adding word-level and character-level representations to the entity-level embeddings. The JOINT model is our FIGMENT system.

Table 4 shows the results for label-based measures for all (102 types), head (frequency > 3000; 15 types) and tail (frequency < 200; 36 types) types. Each row represents one of the models. If not mentioned explicitly, the macro F_1 for all types (macro in Table 4) is the measure we talk about when comparing models.

Overall, we see a similar trend as in Table 3. The only exception is that AGG-FIGER (line 23) is better here than CNN+MIML-ATT (line 22). JOINT (line 26) is also worse than JOINT-BASE2 (line 25) in P@50 measures in all and head types. The main points are still valid: (i) CNN is better for learning representations than FF; (ii) different levels of entity representations are complementary; (iii) ELR is very powerful; (iv) MIML is very effective in mitigating noise of distant supervision and MIML-ATT is the best algorithm in this regard; (v) CM and GM have complementary properties and their joint model performs the best.

5.3 Analysis

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 without description ($\sim$40,000 out of $\sim$200,000).

We then build a simple entity representation by averaging the embeddings of the top k words (w.r.t. tf-idf) of the description (henceforth, AVG-DES).[6] This representation is used similar to our embeddings in Section 4.1 to train a GM. We also train our best GM model ELR+SWLR+CLR(CNN) and our best CM model CNN+MIML-ATT, as well as a combination (concatenation) of GM and AVG-DESC (GM+ACG-DES), a joint model of CM and GM (CM+GM), and finally a joint model of all three models (GM+CM+AVG-DES) on this smaller dataset. Since the descriptions are coming from Wikipedia, we use 100-dimensional Glove (Pennington, Socher, & Manning, 2014) embeddings pretrained on Wikipdia+Gigaword to get a good coverage of words. For our CM and GM models, we still use the embeddings we trained before.

Results are shown in Table 5. We only show the micro F_1, the most represenative measure. GM works better than AVG-DES, again showing the power of global models in this task. The joint model, GM+CM, again improves over each single model. A more important result is that adding AVG-DESC to GM is very effective, especially for the tail entities for which the micro F_1 is improved by 9%. 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 this for future work. The best model here is the joint model (GM+CM+AVG-DES), with slight improvements over GM+AVG-DES.

	entities		
	all	head	tail
AVG-DES	.760	.786	.722
GM	.817	.842	.741
CM	.759	.755	.632
GM+CM	.827	.854	.748
GM+AVG-DES	.858	.870	.830
GM+CM+AVG-DES	**.865**	**.878**	**.834**

Table 5: Micro average F_1 results on the dataset of entities with Wikipedia description.

Comparison of character/word-level GM models on unknown vs. known entities. To do a more fine-grained comparison between GM models that are based on an entity's name and the feature-based baselines, we do a further analysis. 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 6 shows that NSL works worse than CNN on known entities (1.2%) but it is much worse on unknown entities (by 5.8%), justifying our preference for deep learning CLR models. As expected, BOW works relatively well for known entities and really poorly for unknown entities. Basically,

6. k = 20 gives the best results on dev.

BOW has no clue for the unknown entities. For WWLR and SWLR 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, they have access to informative representations of words. SWLR does not have OOVs and therefore works better than WWLR on the unknown entities. For the known entities, WWLR works better.

	all	known	unknown
CLR(NSL)	.447	.485	.309
CLR(CNN)	**.471**	**.497**	**.367**
BOW	.311	.470	.093
WWLR	**.553**	**.581**	.424
SWLR	.552	.574	**.468**

Table 6: Micro F_1 on the test entities of character and word-level models for all, known and unknown entities.

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, the entity annotations in Clueweb corpus, is correct. However, as mentioned before, it has a precision of only 80-85% and this causes errors. An example is the lunar crater "Buffon" in Freebase, a "location". Its predicted type is "author" 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 21,378 entities with type "person" in the test set and among them 2,600 entities (12%) do not have any finer grained type.

This is potentially due to the incompleteness of entity types in Freebase, because if a person does not have a more fine-grained type, she/he is probably not sufficiently famous to be in Freebase. To confirm this hypothesis, we examine the output of FIGMENT on this subset of entities. For 62% of the errors, 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, but missing in Freebase.

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 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.

6. Conclusion

We presented FIGMENT, a corpus-level system that uses textual information for fine-grained entity typing. We designed two scoring models for pairs of entities and types: a global model that computes type scores based on aggregated entity information and a context model that aggregates the type scores of individual entity contexts. We used embeddings of characters, words, entities and types to represent entity names and contexts. Our experimental results showed that the global and the context models provide complementary information for entity typing. As a result, a joint model performed the best.

Acknowledgments

We would like to thank the anonymous reviewers for their great comments. This work was partially supported by the European Research Council: ERC Advanced Grant #740516 NonSequeToR.

Appendix A. Hyperparameters

	model	hyperparameters
GM	CLR(FF)	$d_c = 15, h_{mlp} = 600$
	CLR(CNN)	$d_c = 10, w = [1, .., 8]$ $n = 100, h_{mlp} = 800$
	WWLR	$h_{mlp} = 400$
	SWLR	$h_{mlp} = 400$
	SWLR+CLR(CNN)	$w = [1, ..., 7]$ $d_c = 10, n = 50, h_{mlp} = 700$
	ELR	$h_{mlp} = 400$
	ELR+SWLR	$h_{mlp} = 600$
	ELR+CLR(CNN)	$d_c = 10, w = [1, ..., 7]$ $n = 100, h_{mlp} = 700$
	ELR+SWLR+CLR	$d_c = 10, w = [1, ..., 7]$ $n = 50, h_{mlp} = 700$
	FF	$h_{mlp} = 500, cs = 10$
	CNN	$w = [1, 2, 3, 4], n = 300, h_{mlp} = 600$
	AVG-DES	$h_{mlp} = 600$
	GM+AVG-DES	$d_c = 10, w = [1, ..., 8]$ $n = 100, h_{mlp} = 1500$
CM	FF	$cs = 10, h_{mlp} = 500$
	CNN	$cs = 10, h_{mlp} = 600, w = [1, 2, 3, 4]$

Table 7: Hyperparameters of different models. w is the CNN 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 output MLP. cs is the context size.

References

Adel, H., Roth, B., & Schütze, H. (2016). 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*, pp. 828–838.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Boldrin, M., & Levine, D. K. (2008). *Against intellectual monopoly*. Cambridge University Press Cambridge.

Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the International Conference on Management of Data*, pp. 1247–1250.

Bordes, A., Usunier, N., García-Durán, A., Weston, J., & Yakhnenko, O. (2013). Irreflexive and hierarchical relations as translations. In *Proceddings of the Workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*.

ClueWeb-URL (2012). Clueweb project.. `http://lemurproject.org/clueweb12/`.

Del Corro, L., Abujabal, A., Gemulla, R., & Weikum, G. (2015). Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 868–878.

Dong, L., Wei, F., Sun, H., Zhou, M., & Xu, K. (2015). A hybrid neural model for type classification of entity mentions. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pp. 1243–1249.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

Durrett, G., & Klein, D. (2014). A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, *2*, 477–490.

Gabrilovich, E., Ringgaard, M., & Subramanya, A. (2013). Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0)..

Gupta, S., & Manning, C. D. (2014). Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the 18th Conference on Computational Natural Language Learning*, pp. 98–108.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*, 146–162.

Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., & Weld, D. S. (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*, pp. 541–550.

Jiang, X., Tresp, V., Huang, Y., & Nickel, M. (2012). Link prediction in multi-relational graphs using additive models. In *Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data*, pp. 1–12.

Lin, Y., Shen, S., Liu, Z., Luan, H., & Sun, M. (2016). Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 2124–2133.

Ling, W., Dyer, C., Black, A. W., & Trancoso, I. (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*, pp. 1299–1304.

Ling, X., Singh, S., & Weld, D. (2015b). Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, *3*, 315–328.

Ling, X., & Weld, D. S. (2012). Fine-grained entity recognition. In *Proceedings of the 16th AAAI Conference on Artificial Intelligence*.

McNamee, P., & Dang, H. T. (2009). Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference*, pp. 111–113.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*.

Min, B., Grishman, R., Wan, L., Wang, C., & Gondek, D. (2013). 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*, pp. 777–782.

Mintz, M., Bills, S., Snow, R., & Jurafsky, D. (2009). 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*.

Neelakantan, A., & Chang, M.-W. (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*, pp. 515–525.

Nickel, M., Tresp, V., & Kriegel, H. (2012). Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st World Wide Web Conference*, pp. 271–280.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543.

Rabinovich, M., & Klein, D. (2017). Fine-grained entity typing with high-multiplicity assignments. *CoRR*, *abs/1704.07751*.

Ren, X., He, W., Qu, M., Voss, C. R., Ji, H., & Han, J. (2016). Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, pp. 1825–1834.

Riedel, S., Yao, L., & McCallum, A. (2010). Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pp. 148–163. Springer.

Riedel, S., Yao, L., McCallum, A., & Marlin, B. M. (2013). Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 74–84.

Ritter, A., Zettlemoyer, L., Mausam, & Etzioni, O. (2013). Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, *1*, 367–378.

Shimaoka, S., Stenetorp, P., Inui, K., & Riedel, S. (2016). An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pp. 69–74.

Shimaoka, S., Stenetorp, P., Inui, K., & Riedel, S. (2017). Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 1271–1280.

Socher, R., Chen, D., Manning, C. D., & Ng, A. Y. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pp. 926–934.

Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 697–706.

Surdeanu, M., Tibshirani, J., Nallapati, R., & Manning, C. D. (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*, pp. 455–465.

Suzuki, M., Matsuda, K., Sekine, S., Okazaki, N., & Inui, K. (2016). Fine-grained named entity classification with wikipedia article vectors. In *Proceedings of the 2016 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 483–486.

Thelen, M., & Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pp. 214–221.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*, 2579–2605.

van Erp, M., & Vossen, P. (2017). Multilingual fine-grained entity typing. In *Proceedings of the First International Conference of Language, Data, and Knowledge*, pp. 262–275.

Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Commun. ACM*, *57*(10), 78–85.

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1591–1601.

Weston, J., Bordes, A., Yakhnenko, O., & Usunier, N. (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*, pp. 1366–1371.

Xie, R., Liu, Z., Jia, J., Luan, H., & Sun, M. (2016). Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the 30th Conference on Artificial Intelligence*, pp. 2659–2665.

Yaghoobzadeh, Y., Adel, H., & Schütze, H. (2017). Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 1183–1194.

Yaghoobzadeh, Y., & Schütze, H. (2015). Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 715–725.

Yaghoobzadeh, Y., & Schütze, H. (2016). Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 236–246.

Yaghoobzadeh, Y., & Schütze, H. (2017). 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*, pp. 578–589.

Yogatama, D., Gillick, D., & Lazic, N. (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*, pp. 291–296.

Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M., & Weikum, G. (2012). HYENA: Hierarchical type classification for entity names. In *Proceedings of the 24th International Conference on Computational Linguistics*, pp. 1361–1370.

Zeng, D., Liu, K., Chen, Y., & Zhao, J. (2015). Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1753–1762.