Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Entity Disambiguation using Embeddings

Master's Thesis

2019 Bc. Lejla Metohajrová

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Entity Disambiguation using Embeddings

Master's Thesis

Study programme:	Computer Science
Field of study:	2508 Computer Science
Study department:	Department of Computer Science
Advisor:	Prof. Ing. Igor Farkaš, Dr.
Consultants:	Prof. Dr. Philippe Cudré-Mauroux
	Akansha Bhardwaj, MSc
	Paolo Rosso, MSc

Bratislava, 2019
Bc. Lejla Metohajrová

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Lejla Metohajrová
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Entity Disambiguation using Embeddings

Annotation: Named entity recognition (NER) and entity disambiguation (ED) are essential for semantic language understanding. ED automatically resolves references to entities in a given knowledge base. In this project, we will explore the current state-of-the-art for ED.

Aim:
1. Explore and replicate the current state-of-the-art for ED.
2. Improve the current state-of-the-art or implement it in a novel context.

Literature:
Ganea, O. E., & Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. arXiv preprint arXiv:1704.04920.
Le, P., & Titov, I. (2018). Improving entity linking by modeling latent relations between mentions. arXiv preprint arXiv:1804.10637.
Kolitsas, N., Ganea, O. E., & Hofmann, T. (2018). End-to-End Neural Entity Linking. arXiv preprint arXiv:1808.07699.

Supervisor:	prof. Ing. Igor Farkaš, Dr.
Consultant:	prof. Philippe Cudre-Mauroux, Dr.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	13.12.2018

Approved: 15.12.2018 prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

Student Supervisor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Lejla Metohajrová
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Entity Disambiguation using Embeddings
Rozpoznávanie entít pomocou vnorených vektorov

Anotácia: Rozpoznávanie pomenovaných entít a ich významu je dôležitou súčasťou sémantického porozumenia jazyka, kde automaticky rozpoznávame entity z danej databázy vedomostí v referenciách v texte. V tomto projekte budeme skúmať súčasné metódy pre rozpoznávanie pomenovaných entít.

Cieľ: 1. Preskúmajte a zreplikujte súčasné metódy pre rozpoznávanie pomenovaných entít.
2. Vylepšite súčasné metódy alebo ich implementujte v novom kontexte problému.

Literatúra:
Ganea, O. E., & Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. arXiv preprint arXiv:1704.04920.
Le, P., & Titov, I. (2018). Improving entity linking by modeling latent relations between mentions. arXiv preprint arXiv:1804.10637.
Kolitsas, N., Ganea, O. E., & Hofmann, T. (2018). End-to-End Neural Entity Linking. arXiv preprint arXiv:1808.07699.

Vedúci: prof. Ing. Igor Farkaš, Dr.
Konzultant: prof. Philippe Cudre-Mauroux, Dr.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 13.12.2018

Dátum schválenia: 15.12.2018 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Acknowledgements:

I would like to thank my supervisor Prof. Ing. Igor Farkaš, Dr. for his guidance and support throughout writing this thesis. I would also like to thank Prof. Dr. Philippe Cudré-Mauroux for the opportunity of working on this project. My sincere gratitude belongs to Akansha Bhardwaj, MSc and Paolo Rosso, MSc for their professional advice and continuous support.

Abstract

Short documents, such as posts on social media, lack the amount of context needed for deep learning. Modern systems for entity disambiguation (ED) for short documents often rely on annotated Knowledge Bases, extensive feature engineering and using simple machine learning models. We propose a novel approach to ED for short documents by leveraging representation learning and statistical information about entities from large amounts of unstructured data. The fact that even just our baseline model, that uses empirical probability from large unstructured data to select the top entity candidate, shows competitive performance to previous research is a clear evidence that such information is useful for this task. By experimenting with the current state-of-the-art approaches for ED for long documents, we develop a model that outperforms current systems on all available datasets for ED for short documents. Additionally, we publish a new dataset for ED with short documents based on Twitter posts labeled by earlier research.

Keywords: entity disambiguation, entity embeddings, word embeddings, natural language processing, social media

Abstrakt

Krátke dokumenty, ako napríklad príspevky na sociálnych sieťach, neobsahujú dostatočné množstvo kontextu, ktoré je potrebné pre učenie hlbokých neurónových sietí. Moderné systémy pre rozpoznávanie pomenovaných entít v krátkych dokumentoch sa často spoliehajú na anotované databázy znalostí, rozsiahle inžinierstvo vlastností dát a použitie jednoduchých modelov strojového učenia. V tejto práci navrhujeme nový prístup k rozpoznávaniu entít v krátkych dokumentoch, kde využívame učenie reprezentácií a štatistické informácie o entitách z veľkého množstva neštrukturovaných dát. Fakt, že už len náš základný model, ktorý používa empirickú pravdepodobnosť z veľkých neštrukturovaných dát na to, aby vybral najlepšieho kandidáta pre danú entitu, ukazuje výsledky porovnateľné s predchádzajúcim výskumom, je jasným dôkazom toho, že takéto informácie sú pre danú úlohu užitočné. Experimentovaním s najmodernejšími prístupmi pre rozpoznávanie entít v dlhých dokumentoch sme vyvinuli model, ktorý prekonáva súčasné systémy na všetkých prístupných datasetoch pre rozpoznávanie entít v krátkych dokumentoch. Navyše publikujeme aj nový dataset pre rozpoznávanie entít v krátkych dokumentoch, založený na príspevkoch z Twitteru, ktoré boli označené v skorších výskumoch.

Kľúčové slová: rozpoznávanie entít, vnorené vektory entít, vnorené vektory slov, spracovanie prirodzeného jazyka, sociálne siete

Contents

Introduction

With the increasing volume of textual content that can reach terabytes each day, there is a growing need for automatic methods of text summarization, semantic understanding and, eventually, question answering. The key step to these goals is entity linking that reveals the semantics of spans of text that refer to real-world entities. The main challenge arises from word ambiguities inherent to natural language: surface form *synonymy*, i.e., different spans of text referring to the same entity, and *homonymy*, i.e., the same name being shared by multiple entities.

The goal of entity disambiguation (ED) is to resolve potentially ambiguous mentions of entities to their canonical representations such as corresponding Wikipedia articles or referent entities in a Knowledge Base, like Wikidata (Vrandečić and Krötzsch, 2014), DBPedia (Auer et al., 2007) and YAGO (Suchanek et al., 2007).

Popular works usually try to link entities present in long documents with a lot of context, such as Wikipedia. Two types of information in a document is usually captured for ED: (i) contextual words surrounding an entity mention and (ii) other entity mentions that may or may not be in a relation with the current entity mention.

Due to a relatively small amount of such data, it is very difficult to disambiguate entities in short texts like posts on social media. The entities to be recognized are often complex and surrounded by highly subjective context. Previous research of ED for short texts exhausted many different approaches like extensive feature engineering (Waitelonis and Sack, 2016; Sakor et al., 2019), graph-based algorithms (Yang and Chang, 2016; Cao et al., 2015) and unsupervised disambiguation methods (Feng et al., 2018).

However, only a few papers exploit additional data other than short texts with little context. Structured information present in Knowledge Bases was shown to be useful for ED for short texts (Sakor et al., 2019). Useful information can be also obtained using semantic similarity methods trained over large amounts of unstructured data (Zhu and Iglesias, 2018).

We propose a novel approach to ED for short texts by leveraging representation learning and statistical information about entities from unstructured and context-rich data. We find that using complementary or related information can often lead to achieving state-of-the-art results in machine learning. This approach avoids using

any hand-crafted features and instead focuses on automating the process, which leads to a generalizable solution that could be extended for use in other domains and for other languages.

Chapter 1

Related Work and State of the Art

The field of artificial intelligence with focus on understanding natural language is referred to as *natural language processing* (NLP). There are many interesting problems in NLP such as automatic summarization, semantic search and question answering that rely on determining what entities appear in a given text.

Recognition and disambiguation of potentially ambiguous surface forms of entities in a natural language text into their canonical representations in a Knowledge Base (KB) is known as *entity linking*. It comprises two stages: *a) mention detection* and *b) entity disambiguation*. Some previous efforts focus on end-to-end entity linking (Kolitsas et al., 2018; Hoffart et al., 2011). In this thesis, however, we follow a series of research papers on entity disambiguation (ED) only, that assume already discovered entity mentions. Given a natural language text with detected *mentions* of interest, we try to answer which *entities* from a KB are the mentions referring to.

In the following section, we describe and define the task of entity disambiguation, in Section 1.2 we describe the methodology used in the related literature. Later, in Section 1.3 , we explain the current state-of-the-art research of ED and compare the approaches for long (Section 1.3.1) and short textual documents (Section 1.3.2).

1.1 Entity Disambiguation

News articles, blog posts and social media posts all contain references to real world entities recognized by public. For an example of entity disambiguation problem, consider the following sentence:

Woods is returning to the East Coast after winning the golf tournament in the UK.

The task of ED is to assign entities from a KB to the underlined entity mentions. Mentions can, generally, be ambiguous. The contextual words and other mentions from the same document (article, post) are usually used for disambiguating mentions

into correct corresponding entities. In Table 1.1, we show some entity candidates of the mentions from the above example and the correct entities depicted in bold.

Mention	Candidates
Woods	`Wood, Forest,` **`Tiger_Woods`**
East Coast	**`East_Coast_of_the_United_States,`** `National_Express_East_Coast, Eastern_states_of_Australia`
golf	**`Golf,`** `Volkswagen_Golf, Golf_(video_game)`
UK	**`United_Kingdom,`** `Ukrainian_language`

Table 1.1: Selected entity candidates of mentions from the example. The ground truth entities in context of the example are depicted in bold.

In this work, we resolve the mentions *jointly* across an entire document by combining *global information* from document-level entity relations with *local information* captured from mentions and their surrounding context.

It is worth noticing that previous mention detection might have yielded imperfect results. In our example, we could argue that the whole span *golf tournament* might have been detected instead of just *golf*. Correct detection could lead to disambiguating the mention into entities like `Masters_Tournament, US_Open_(golf)` and **`The_Open_Championship`** (which is the only major golf tournament held in the UK).

1.1.1 Definition

Let Δ be a Knowledge Base (KB), consisting of a set of entities $\mathcal{E}$, $\mathcal{V}$ a finite dictionary of phrases or names and $\mathcal{C}$ a context representation based on a corpus $\mathcal{D}$. Formally, we seek a mapping $F : (\mathcal{V}, \mathcal{C})^n \rightarrow \mathcal{E}^n$, that takes as input a sequence of linkable spans, *mentions* $\mathbf{m} = (m_1, ..., m_n) \in \mathcal{V}$ along with their context windows $\mathbf{c} = (c_1, ..., c_n) \in \mathcal{C}$ and produces a joint entity assignment $\mathbf{e} = (e_1, ..., e_n) \in \mathcal{E}$. Here n refers to the number of linkable spans in a document $D \in \mathcal{D}$ and each context window c_i consists of contextual words $c_i = w_1, ..., w_K$, where K is the size of the context window. This problem is known as *entity disambiguation* or *link generation* in the literature.

We can construct such mapping F in a probabilistic approach, by learning a conditional probability model $p(\mathbf{e}|\mathbf{m}, \mathbf{c})$ from data and then employing (approximate) probabilistic inference in order to find the maximum a posteriori (MAP) assignment, hence:

$$F(\mathbf{m}, \mathbf{c}) := \underset{\mathbf{e}' \in \mathcal{E}^n}{\operatorname{argmax}}\, p(\mathbf{e}'|\mathbf{m}, \mathbf{c}) \tag{1.1}$$

In the sequel, we describe how to estimate such a model from a corpus of entity-linked documents.

1.2 Related work

This section is devoted to a description of the common methodology established in previous related literature, as well as description of the current state-of-the-art publications. In the first part, we describe different formulations of the problem of ED and discuss common methodology and terminology. In Section 1.3.1 and Section 1.3.2, we discuss different challenges of ED based on the character of the data and address them in context of the current research.

Existing research often differs in the referent KB that is used as a vocabulary of entities. A common approach is to use KBs, such as Wikidata (Vrandečić and Krötzsch, 2014), DBPedia (Auer et al., 2007) and YAGO (Suchanek et al., 2007). These KBs, or Knowledge Graphs (KG), often contain relevant information about entities, such as description, aliases of the entity and RDF-like triplets (subject, predicate, object), that can be used for ED. Recent results show that it is possible to achieve good results on common ED datasets using only Wikidata without other training data (e.g. long documents with entity mentions, such as Wikipedia) (Delpeuch, 2019).

This approach is difficult to generalize to languages and domains which do not have KBs or experts available to hand-craft such features. We, in contrast, focus on automating the process using representation learning.

Using Wikipedia titles as references to entities is a task defined as *Wikification* (Ratinov et al., 2011). There is a vast research using Wikipedia as referent KB for entities. Wikipedia is an available corpus of entity-linked documents. The hyperlinks are considered ground truth annotations, the mention being the linked span of text and the truth entity being the Wikipedia page it refers to. One can extract two kinds of basic statistics from such a corpus: 1) counts of how often each entity was referred to by a specific name and 2) pairwise co-occurrence counts for entities in the same documents.

Note that, in this work, we only link mentions that have a valid gold KB entity. This setting is referred to as *InKBevaluation* (Röder et al., 2018). Thus, we treat mentions referring to entities outside of the KB as "non-linkable". This is in line with a few previous models (Yamada et al., 2016; Ganea and Hofmann, 2017; Kolitsas et al., 2018). We leave the interesting setting of discovering out-of-KB entities as future work.

1.2.1 Candidate Selection

Since there are usually many entities in a KB, it is absolutely necessary to choose a feasible set of potential candidates, eliminating options which are highly unlikely. For example, the word *"Apple"* has 52 different entity candidates on Wikipedia (equivalent to 52 entries on Apple's disambiguation page on Wikipedia) [1].

This preprocessing step is called *candidate selection*. Candidate selection is mostly achieved by performing a heuristic to choose the candidate set by statistics derived from the training data. In our work, we consider the top entity candidates based on an empirical entity-mention probability distribution $p(e|m)$ inspired by recent research (Ganea et al., 2016). This distribution is computed by averaging probabilities from two indexes built from mention-entity hyperlink count statistics from Wikipedia and a large Web corpus (Spitkovsky and Chang, 2012), and the YAGO (Hoffart et al., 2011) dictionary where each candidate receives a uniform probability distribution.

Sometimes the candidates can be further reduced to solving a domain-specific problem, for which only a subset of the general candidates are likely to appear within the context (Feng et al., 2018). For example, categories of the entity candidates can be used as additional information for eliminating unrelated candidates.

Thus, after this step, the task of a trainable model is reduced to choosing the best option among a smaller list of entity candidates $\mathcal{E}_{m_i} = (e'_{i,1}, ..., e'_{i,k})$ for each mention m_i. Here, k is a small pre-defined maximal number of entity candidates for each mention. In what follows, we will discuss two classes of approaches tackling the problem of choosing the best candidate: local and global modeling.

1.2.2 Local and Global Models

Local models consider the individual context of each entity mention in isolation in order to reduce the size of the decision space, utilizing clues such as the textual similarity between the context and each disambiguation entity candidate's Wikipedia page.

Let c_i be a local context of mention m_i and $\Psi(e'_i, c_i)$ be a local score function. A local model then tackles the problem by searching for the entity candidate e^*_i that optimizes the local score for each $i \in \{1, ..., n\}$ from a document D with n mentions, namely:

$$e^*_i = \underset{e'_i \in \mathcal{E}_{m_i}}{\operatorname{argmax}} \Psi(e'_i, c_i) \tag{1.2}$$

Usually earlier research depends only on local models (Mihalcea and Csomai, 2007; Milne and Witten, 2008), however, some recent work shines a new light on this approach (Lazic et al., 2015; Yamada et al., 2017).

[1] https://en.wikipedia.org/wiki/Apple_(disambiguation)

Global models attempt to jointly disambiguate all mentions in a document to arrive at a *coherent* set of disambiguated entities, based on the assumption that the underlying entities are correlated and consistent with the main topic of the document. This approach is also often referred to as *collective disambiguation* or *joint entity disambiguation*.

A global model, besides using local context within $\Psi(e'_i, c_i)$, takes into account *entity coherence*, captured by a coherence score function $\Phi(\mathbf{e}')$:

$$\mathbf{e}^* = \underset{\mathbf{e}' \in \mathcal{E}_{m_1} \times ... \times \mathcal{E}_{m_n}}{\operatorname{argmax}} \sum_{i=1}^{n} \Psi(e'_i, c_i) + \Phi(\mathbf{e}') \tag{1.3}$$

where $\mathbf{e}' = (e'_1, ..., e'_n)$ is a vector of entity candidates for mentions $(m_1, ..., m_n) \in D$ assigned for evaluation in a given iteration.

The coherence score function, in the simplest form, is a sum over all pairwise scores $\Phi(e'_i, e'_j)$ (Ratinov et al., 2011; Ganea et al., 2016; Globerson et al., 2016; Yamada et al., 2016) resulting in:

$$\mathbf{e}^* = \underset{\mathbf{e}' \in \mathcal{E}_{m_1} \times ... \times \mathcal{E}_{m_n}}{\operatorname{argmax}} \sum_{i=1}^{n} \Psi(e'_i, c_i) + \sum_{i \neq j}^{n} \Phi(e'_i, e'_j) \tag{1.4}$$

While this approach tends to result in superior accuracy, the main disadvantage is that the space of possible entity assignments grows combinatorially and the exact decoding of a global model is NP-hard (Wainwright et al., 2008). As a consequence, many approaches in this group rely on approximate inference mechanisms (Globerson et al., 2016; Ganea et al., 2016). Local models solve each mention independently. Therefore, they are, generally, computationally more effective than global models.

Previous work has investigated different approximation techniques, including: random graph walks (Guo and Barbosa, 2018), personalized PageRank (Pershina et al., 2015), intermention voting (Ferragina and Scaiella, 2010), graph pruning (Hoffart et al., 2011), integer linear programming (Cheng and Roth, 2013) and ranking SVMs (Ratinov et al., 2011). Some recent research explores even deep reinforcement learning for collective entity linking (Fang et al., 2019).

Another branch of research tries to utilize the weight of a minimum spanning tree to measure the coherence between entities and, instead of considering all the given mentions, a pair with the highest confidence is iteratively being selected at each step for decision making (Phan et al., 2018).

Recently, pretrained embeddings of words (Mikolov et al., 2013a; Pennington et al., 2014) and entities (He et al., 2013; Yamada et al., 2017; Ganea and Hofmann, 2017) have been used to model the relations between words and entities in the local context and relations between entities in the pairwise scores. These approaches often do not use any other hand-crafted features and are thus generalizable for other domains and problems.

Further research improves the techniques of collective disambiguation by modeling not only coherence, but multiple different latent relations between entity mentions. While previous works relied on hand-crafted features and relations present in KGs (Cheng and Roth, 2013), modern approach chooses automating the process by leveraging representation learning (Le and Titov, 2018).

The learned relations are represented by *relation embeddings*. This approach assumes K latent relations and for each entity mention pair (m_i, m_j) assigns a non-negative weight α_{ijq} to every relation $q \in (1, ..., K)$. The pairwise scores are then simply computed as a weighted sum of all relation-specific scores $\Phi_q(e'_i, e'_j)$. Formally:

$$\Phi(e'_i, e'_j) = \sum_{q=1}^{K} \alpha_{ijq} \Phi_q(e'_i, e'_j) \tag{1.5}$$

In this thesis, we follow the line of research emphasizing representation learning (Ganea and Hofmann, 2017; Le and Titov, 2018) due to achieving great results by elegant automated approach. We will describe our candidate selection process as well as our choice of local score function $\Psi(e'_i, c_i)$, pairwise score function $\Phi(e'_i, e'_j)$ and relation scores $\Phi_q(e'_i, e'_j)$ in Chapter 2.

1.3 State of The Art

The focus of state-of-the-art research is mostly on long documents, such as Wikipedia, with rich context and many entity-entity co-occurrences. Disambiguating entities in short texts like posts on social media suffers from issues like small amount of context that is highly subjective and only few entities are present in a document[2]. This makes ED for short documents a very challenging problem. In this section, we present the current state-of-the-art research for long and short documents and compare the underlying challenges behind the two problems.

1.3.1 Entity Disambiguation for Long Documents

Vast amounts of data provide the perfect conditions for training complex models of NLP tasks. Entity disambiguation is not an exception. Most of the previous research focuses on solving ED in long documents. The proposed solution in most cases starts with candidate selection based on entity-mention co-occurrence counts, such as we described in Section 1.2.1.

Some research of ED for long documents only uses local models by scoring entity candidates based on local context and possibly other additional information like Wikipedia pages of the entity candidates, information from KGs or word and entity

[2]A Twitter post (in ED data) usually contains ∼1.5 entity mentions

embeddings (e.g. Mihalcea and Csomai, 2007; Yamada et al., 2017). It is very common to use local models together with global models that also take into account information such as relations of the different entities in one document and often disambiguate all entities jointly within a document (Hoffart et al., 2011; Ganea and Hofmann, 2017; Le and Titov, 2018; Phan et al., 2018; Fang et al., 2019, etc.). The general approach of using local and global models is described in Section 1.2.2.

For long documents it can be also beneficial not to assume only a single topic, but rather multiple topics per document. Naturally, *topic modeling* can be used for entity disambiguation by attempting to harmonize the individual distribution of latent topics across candidate entities. Previous research explores applying Latent Dirichlet Allocation (LDA) (Houlsby and Ciaramita, 2014; Pilz and Paaß, 2011) and compare the resulting topic distribution of the input document to the topic distributions of the disambiguated entities' Wikipedia pages.

Despite the dominance of global models in recent years, current state of the art for ED for long documents reformulates the problem of ED as a type system of hundred categories of entities obtained from the Wikidata KG (Raiman and Raiman, 2018). In this work, each entity is assigned a unique ∼100-dimensional vector representing its membership in each category. The ED system then learns a mapping from a mention plus its context to the type representation of the ground truth entity. Thus, the system is a local model using the benefits of lower complexity. It would be interesting to test this approach for short documents. The system, however, assumes choosing a good specific set of categories which might not be fully automated. Therefore, we leave this as a future work.

1.3.2 Entity Disambiguation for Short Documents

Short documents like posts on social media and queries lack sufficient informative data to describe complex entities, therefore ED for short documents is a very challenging task. Due to this nature of the problem, few methods have been proposed to solve ED for short documents.

Previous research mostly relied on massive feature engineering and external handcrafted KBs (Cao et al., 2015; Habib and Van Keulen, 2016; Waitelonis and Sack, 2016; Sakor et al., 2019). Other approaches leveraged topic and category information choosing from entity candidates (Feng et al., 2018; Zhu and Iglesias, 2018).

Using additional data is highly beneficial for such task. Typically, previous methods (e.g. Sakor et al., 2019) only use KGs, such as Wikidata (Vrandečić and Krötzsch, 2014) that require human annotation and supervision. Only few approaches explored leveraging unstructured data for learning semantic representations of words and categories (Zhu and Iglesias, 2018).

In the next section, we propose a novel approach to ED for short documents by leveraging representation learning and statistical information about entities from unstructured and context-rich data. We emphasize on automation of the whole process, avoiding any hand-crafted features, which leads to a generalizable solution that could be extended for use in other domains and for other languages.

Apart from representation learning, we are also one of few that approach ED for short texts as collective disambiguation (Cao et al., 2015; Yang and Chang, 2016). Some previous works argued that modeling global dependencies for short documents can lead to poor performance (Feng et al., 2018) due to the misleading nature of subjective content in short documents like social media posts. However, we aim to overcome these difficulties by the aforementioned representation learning, making more informative decisions.

Chapter 2

Methodology

In this thesis, we aim to design a competitive system for entity disambiguation for short documents. The idea is to explore using additional information, such as entity statistic and representations of words and entities learned from a large corpus of long documents.

In this chapter, we present our model for entity disambiguation for short documents leveraging representation learning and statistical information from context-rich data. Our model is inspired by recent work in the field of ED for long documents (Ganea and Hofmann, 2017; Le and Titov, 2018). In the first section, we define our main objectives and our general approach, later we propose the implementation details.

2.1 Objectives

Let us start by proposing an empirical entity-mention probability distribution $p(e|m)$ based on Wikipedia corpus, such as the probability distribution used in previous related work (Ganea et al., 2016; Kolitsas et al., 2018; Le and Titov, 2018), for selecting a fixed number of entity candidates for each mention. The candidate selection process is described in Section 2.2.1. We solve the collective disambiguation task of selecting the best candidate based on the context and other entities within the same document by proposing a local and a global model in the same fashion used for long documents.

Representation learning is used to represent features like words and entities. Namely, we use pretrained Word2Vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) embedding models to represent contextual words. Instead of using sparse statistics about entity-entity co-occurences, we use entity embeddings bootstrapped from Word2Vec word embeddings, trained independently for each entity, using word-entity statistics from the canonical Wikipedia pages of the entities, and the context surrounding mentions of the entities in the Wikipedia corpus (Ganea and Hofmann, 2017). This approach embeds words and entities in the same low-dimensional vector space, which

gives us the ability to exploit geometric similarity between them. The training of entity embeddings is described in detail in Section 2.2.2.

Based on the assumption that only a few contextual words are informative for resolving an ambiguous mention, our local model uses neural attention mechanism over the local context to compute the local score $\Psi(e'_i, c_i)$ for each entity candidate (Ganea and Hofmann, 2017). A contextual word is considered relevant if it is strongly related to at least one of the entity candidates of a given mention. For short documents, we consider the whole document as the context of the mention (rather than a smaller context window) for producing the local scores. The details of the local model are described in Section 2.2.3

We define our global model as a fully-connected pairwise *conditional random field* (CRF) to model a joint probability distribution over all combinations of entity candidates per document $\mathbf{e}' \in \mathcal{E}_{m_1} \times ... \times \mathcal{E}_{m_n}$. We follow the multi-relational approach (Le and Titov, 2018), using relation scores $\Phi_q(e'_i, e'_j)$ to distinguish different relations between entities for computing the pairwise entity-entity scores $\Phi(e'_i, e'_j)$. For further details about our global model see Section 2.2.4.

Our belief is that further information, like the entity-mention probability distribution and the learned embedded representations could lead to interesting results in entity disambiguation for short documents.

2.2 Implementation Details

In this section, we present our model for entity disambiguation for short documents leveraging representation learning and statistical information from context-rich data.

2.2.1 Candidate Selection

For each mention to be disambiguated, we first select a set of potential candidates by considering the top 30 ranked entities based on the local mention-entity probability distribution $p(e|m)$ (Ganea et al., 2016). It is computed by averaging probabilities from two indices built from mention-entity hyperlink count statistics from Wikipedia and a large Web corpus (Spitkovsky and Chang, 2012), and the YAGO dictionary (Hoffart et al., 2011) where each candidate receives a uniform probability distribution.

Consequently, we further limit the number of candidates per mention by keeping only the top 4 entity candidates with the highest $p(e|m)$ and the top 3 entity candidates based on the local context-entity similarity $\sum_{w \in c} \mathbf{x}_e^\top \mathbf{x}_w$. These pruning heuristics result in a significantly improved running time at an insignificant accuracy loss. We have chosen these numbers based on previous research (Ganea and Hofmann, 2017; Le and Titov, 2018).

Following previous works (Yamada et al., 2016; Ganea and Hofmann, 2017; Le and Titov, 2018), we considered only the mentions that have entities in the KB (in our case, the mention-entity map $p(e|m)$).

2.2.2 Entity Embeddings

In this section, we describe the training of entity embeddings that we inherit from previous research (Ganea and Hofmann, 2017). The entity embeddings share the same vector space with popular word embeddings (Mikolov et al., 2013a; Pennington et al., 2014) that are pretrained in our setting. This allows us to take geometric and, by definition, also semantic similarity measures between words and entities.

These vector representations are a key component to avoid hand-engineered features, multiple disambiguation steps, or the need for additional *ad hoc* heuristics when solving the ED task. Apart from that, a great advantage of this approach is that it works well for rare entities with few mentions and new entities can always easily be added in an incremental manner, which is important in practice.

Let $\mathbf{x} : \mathcal{W} \to \mathbb{R}^d$ be a pretrained word embedding map encoding the meaning of words $e \in \mathcal{W}$. For this purpose,Word2Vec (Mikolov et al., 2013a) pretrained on the Google News dataset is used. We describe below, how to extend this map to entities $\mathcal{E}$, i.e. $\mathbf{x} : \mathcal{E} \to \mathbb{R}^d$.

Let us assume a generative model in which words that co-occur with an entity e are sampled from a conditional distribution $p(w|e)$ when they are generated. The distribution is approximated empirically, by the word-entity co-occurrence counts $\#(w, e)$ from two sources: (i) the entity's canonical Wikipedia page, and (ii) the windows of fixed size surrounding mentions (hyperlinks) of the entity in Wikipedia. For the generative model, this is the *positive* distribution of words w related to entity e. Furthermore, we assume that the *negative* words (i.e. unrelated to e) are drawn from a word prior probability distribution $p(w)$. Smoothed unigram distribution is used for this approximation.

For training, a max-margin objective is used to infer the optimal embedding $\mathbf{x}_e$ of entity e. The idea is that vectors of positive words $w^+ \sim p(w|e)$ are closer to the entity vector $\mathbf{x}_e$ than vectors of random (negative) words $w^- \sim p(w)$. Let $\gamma > 0$ be a margin parameter and $[\cdot]_+$ the ReLU function (see Figure 3.4). The objective is then defined as:

$$\begin{aligned} J(\mathbf{z}; e) &:= \mathbb{E}_{w^+|e}\mathbb{E}_{w^-}[h(\mathbf{z}; w^+, w^-)] \\ h(\mathbf{z}; w, v) &:= [\gamma - \langle \mathbf{z}, \mathbf{x}_w \rangle]_+ \\ \mathbf{x}_e &:= \operatorname*{argmin}_{\mathbf{z}:||\mathbf{z}||=1} J(\mathbf{z}; e) \end{aligned} \tag{2.1}$$

The training was done by previous work (Ganea and Hofmann, 2017) using AdaGrad

optimizer (see Section 3.2.3).

2.2.3 Local Model with Neural Attention

We implement the local neural attention mechanism based by recent research (Ganea and Hofmann, 2017) that uses word and entity embeddings to represent the input features. The insight behind the attention mechanism is that only a few context words are informative for resolving an ambiguous mention. It has been exploited before that focusing only on those words helps to reduce noise and improves disambiguation (Lazic et al., 2015). Here, we assume that a context word may be relevant, if it is strongly related to at least one of the entity candidates of a given mention.

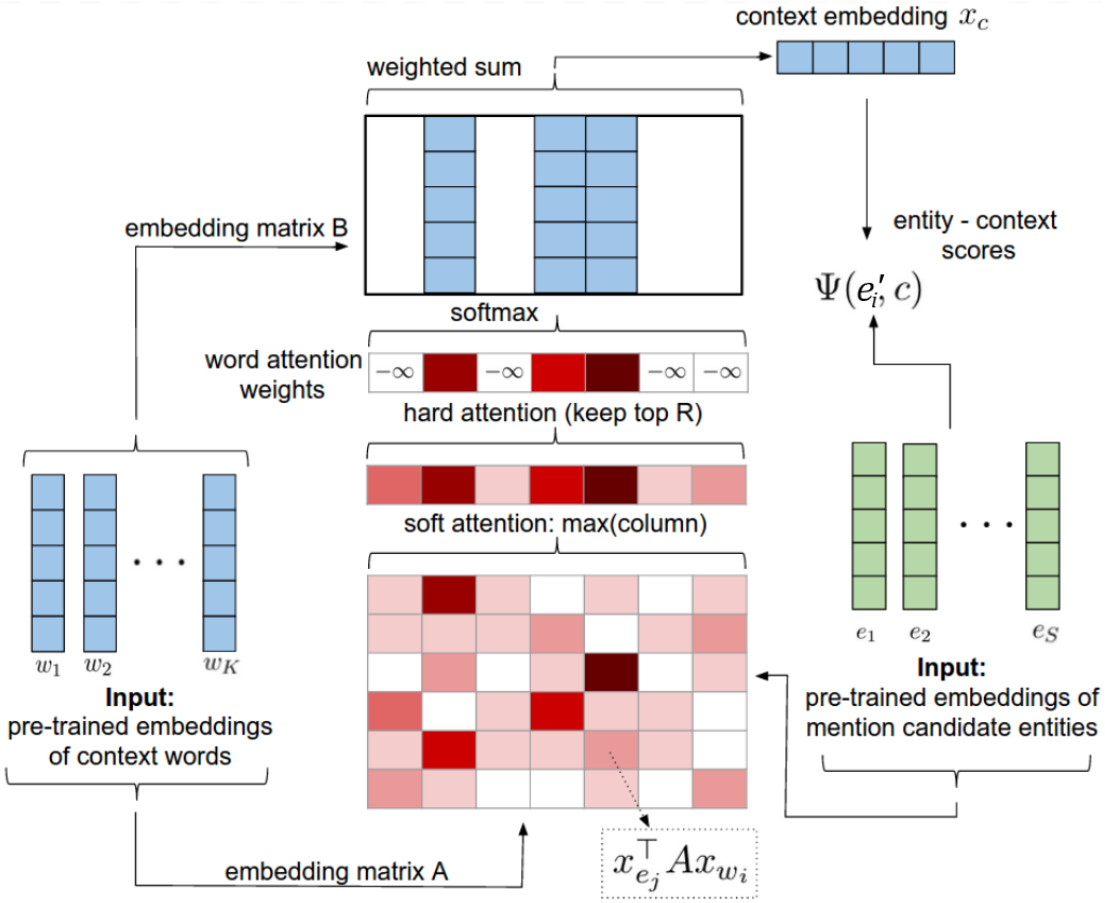

Figure 2.1: Local model with neural attention (Ganea and Hofmann, 2017). The inputs are contextual words $c = (w_1, ..., w_K)$ (on the left, blue) and candidate entities $\mathcal{E}_m = e'_1, ..., e'_S$ (on the right, green) of one mention m. The outputs are local entity candidate scores $\Psi(e'_i, c)$ for each $e'_i \in \mathcal{E}_m$. All parts are differentiable and trainable with backpropagation.

The local model, depicted in Figure 2.1, computes an entity-context score for each candidate $e' \in \mathcal{E}_m$ based on the K-word local context $c = w_1, ..., w_K$ surrounding mention m. It is a composition of differentiable functions, thus it is smooth from input to output, allowing us to easily compute gradients and backpropagate through the

model. Each word $w \in c$ and entity candidate $e' \in \mathcal{E}_m$ is mapped to its embedding, x_w and $x_{e'}$ respectively, via the pretrained map x (see Section 2.2.2). We then compute an unnormalized support score for each word in the context as follows:

$$u(w) = \max_{e' \in \mathcal{E}_m} x_{e'}^\top \mathbf{A} x_w \tag{2.2}$$

where $\mathbf{A}$ is a parameterized diagonal matrix. The weight is high if the word is strongly related to at least one candidate entity. We apply (hard) pruning to the top $R \leq K$ words with the highest weights:

$$\overline{c} = \{w \in c | u(w) \in topR(u)\} \tag{2.3}$$

and apply a softmax function (see Equation 3.7) on these weights:

$$\beta(w) = \begin{cases} \frac{\exp u(w)}{\sum_{v \in \overline{c}} \exp u(v)} & \text{if } w \in \overline{c} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

The final β-weighted context-based local score of an entity candidate e' with context window c is then defined as:

$$\Psi(e', c) = \sum_{w \in \overline{c}} \beta(w) x_{e'}^\top \mathbf{B} x_w \tag{2.5}$$

where $\mathbf{B}$ is another trainable diagonal matrix.

2.2.4 Multi-relational Global Model

For collective entity disambiguation, we define a global model, based on Equation 1.4, to find the optimal entity-mention assignment vector $\mathbf{e}^*$ over $\mathcal{E}_{m_1} \times ... \times \mathcal{E}_{m_n} \ni \mathbf{e}'$.

Pairwise Scores

First, we will describe how to compute the pairwise score function $\Phi(e'_i, e'_j)$ and the relation scores $\Phi_q(e'_i, e'_j)$ for K latent relations (Le and Titov, 2018). The pairwise scores $\Phi(e'_i, e'_j)$ are computed as a sum of all relation-specific scores $\Phi_q(e'_i, e'_j)$ for each $q \in (1, ..., K)$, weighted by a non-negative weight α_{ijq} (see Equation 1.5). Each relation q is represented by a trainable diagonal matrix $\mathbf{R}_q \in \mathbb{R}^{d \times d}$ and its relation score is given by:

$$\Phi_q(e'_i, e'_j) = e_i'^\top \mathbf{R}_q e'_j \tag{2.6}$$

The weight α_{ijq} is a normalized score:

$$\alpha_{ijq} = \frac{1}{Z_{ijq}} \exp\left(\frac{f^\top(m_i, c_i) \mathbf{D}_q f(m_j, c_j)}{\sqrt{d}}\right) \tag{2.7}$$

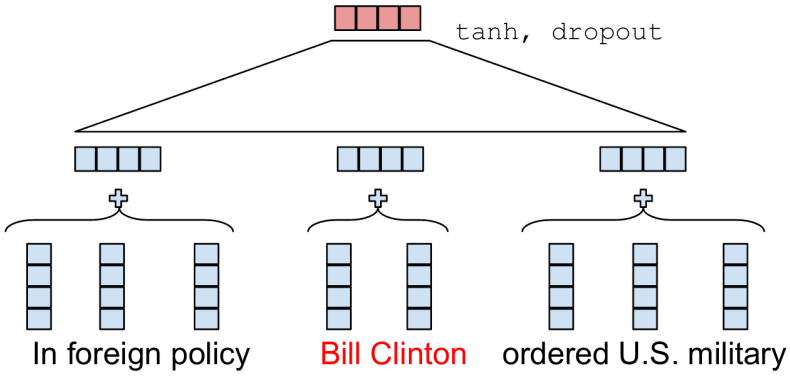

Figure 2.2: Mapping function $f(m_i, c_i)$, a single-layer neural network using hyperbolic tangent (Eq. 3.4) activation function and dropout (see Section 3.2.4). The input is the concatenation of the average embedding of words in the left context, the mention and the right context respectively (Le and Titov, 2018).

where Z_{ijq} is a normalization factor, $f : (\mathcal{V}, \mathcal{C})^n \rightarrow \mathbb{R}^d$ is a mapping function, and $\mathbf{D}_q \in \mathbb{R}^{d \times d}$ is a trainable diagonal matrix. We learn the mapping f by modeling a single-layer neural network depicted in Figure 2.2.

To define the normalization factor Z_{ijq}, we use *ment-norm* (Le and Titov, 2018), normalization over mentions:

$$Z_{ijq} = \sum_{\substack{j'=1 \\ j' \neq i}}^{n} \exp\left(\frac{f^\top(m_i, c_i)\mathbf{D}_q f(m'_j, c'_j)}{\sqrt{d}}\right) \tag{2.8}$$

so that $\sum_{\substack{j'=1 \\ j' \neq i}}^{n} \alpha_{ij'q} = 1$.

The intuition behind ment-norm is that for each relation q and mention m_i, we are looking for mentions related to m_i with relation q. For each pair of m_i and m_j we can distinguish two cases: (i) α_{ijq} is small for all q: m_i and m_j are not related under any relation, (ii) α_{ijq} is large for one or more q: there are one or more relations which are predicted for m_i and m_j.

Ment-norm is a type of attention mechanism that is in line with multi-head attention approach (Vaswani et al., 2017). For each mention m_i and each q, we can interpret α_{ijq} as the probability of choosing a mention m_j among the set of mentions in the document. Each mention m_i will have maximally K mentions (heads) to focus on.

Since there is no further transformation of the output of the attention mechanism, we need to handle cases where all K relations are not applicable for a given mention. Therefore, a padding mention m_{pad} linked to a padding entity e_{pad} is added to each document. Then, the model can use the padding mention to damp the probability mass that other mentions receive.

The final pairwise scores can be computed as a normalized weighted sum of all relation scores:

$$\Phi(e'_i, e'_j) = \sum_{q=1}^{K} \alpha_{ijq} e'^{\top}_i \mathbf{R}_q e'_j \tag{2.9}$$

CRF Model, Learning and Inference

The global model for finding an optimal entity candidate is defined as a fully-connected conditional random field, with the local scores as the unary factor $\Psi(e_i) = \Psi(e_i, c_i)$ and the pairwise scores as defined above:

$$\text{CRF}(\mathbf{e}|\mathbf{m}, \mathbf{c}) = \exp\Big(\sum_{i=1}^{n} \Psi(e_i) + \sum_{i<j} \Phi(e_i, e_j)\Big) \tag{2.10}$$

The goal of global ED prediction is to perform maximum-a-posteriori (MAP) estimation to find the assignments of entities $\mathbf{e}^*$ that maximize $\text{CRF}(\mathbf{e}^*|\mathbf{m}, \mathbf{c})$. Training and prediction of this CRF is NP-hard. Therefore, we use *truncated fitting* of max-product *loopy belief propagation* (LBP) to a fixed number of message passing iterations (Ganea and Hofmann, 2017).

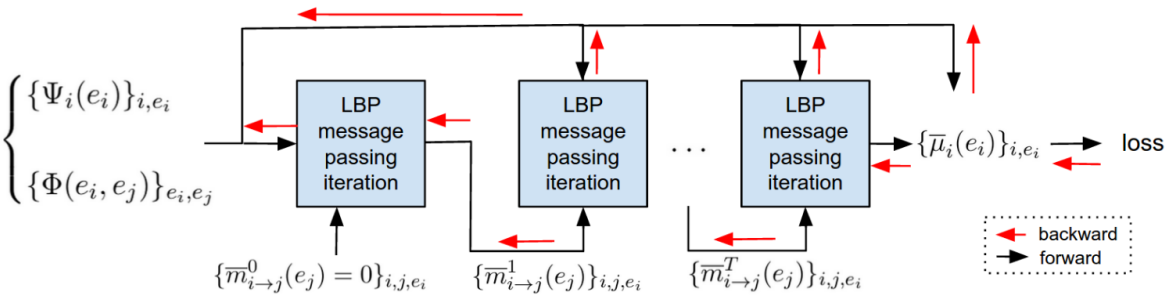

Figure 2.3: Global model: unrolled LBP deep network that is end-to-end differentiable and trainable (Ganea and Hofmann, 2017).

The architecture of the global model with T message passing iterations is shown in Figure 2.3. In each iteration t, mention m_i votes for entity candidate $e \in \mathcal{E}_{m_j}$ using a normalized log-message $\overline{m}^t_{i\to j}(e)$. First, we define the CRF potentials as:

$$m^{t+1}_{i\to j}(e) = \max_{e' \in \mathcal{E}_{m_i}} \{\Psi(e') + \Phi(e, e') + \sum_{k \neq j} \overline{m}^t_{k\to i}(e')\} \tag{2.11}$$

Then, the normalized log-message is computed as:

$$\overline{m}^t_{i\to j}(e) = \log[\delta \cdot \text{softmax}(m^t_{i\to j}(e)) + (1-\delta) \cdot \exp(\overline{m}^{t-1}_{i\to j}(e))] \tag{2.12}$$

where $\delta \in (0, 1]$ is a damping factor. The messages at first iteration (layer) are set to zero. After T iterations (layers), the beliefs (marginals) are computed as:

$$\mu_i(e) = \Psi(e) + \sum_{k \neq i} \overline{m}^T_{k\to i}(e) \tag{2.13}$$

$$\overline{\mu}_i(e) = \frac{\exp\left[\mu_i(e)\right]}{\sum_{e' \in \mathcal{E}_{m_i}} \exp\left[\mu_i(e)\right]} \tag{2.14}$$

We compute the final score using a two-layer neural network g and the entity-mention conditional probability map $p(e|m_i)$:

$$\rho_i(e) = g(\overline{\mu}_i(e), p(e|m_i)) \tag{2.15}$$

We define a margin-based objective and minimize the following ranking loss:

$$L(\theta) = \sum_{D \in \mathcal{D}} \sum_{m_i \in D} \sum_{e \in \mathcal{E}_{m_i}} h(m_i, e) \tag{2.16}$$

$$h(m_i, e) = [\gamma - \rho_i(e_i^*) + \rho_i(e)]_+ \tag{2.17}$$

where $\theta = \{\mathbf{A}, \mathbf{B}, \mathbf{R}, \mathbf{D}$ and the weights of f and $g\}$ are the training parameters, $\mathcal{D}$ is the training corpus and e_i^* is the ground truth entity. *Adam* is used as an optimizer (see Section 3.2.3).

For ment-norm, the padding mention is treated like any other mention. We add $f_{pad} = f(m_{pad}, c_{pad})$ and $e_{pad} \in \mathbb{R}^d$, an embedding of e_{pad}, to the model parameter list, and tune them while training the model.

In order to encourage the models to explore different relations, we add the following regularization term to the loss function in Equation 2.16 (Le and Titov, 2018):

$$\lambda_1 \sum_{i,j} \text{dist}(\mathbf{R}_i, \mathbf{R}_j) + \lambda_2 \sum_{i,j} \text{dist}(\mathbf{D}_i, \mathbf{D}_j) \tag{2.18}$$

where λ_1, λ_2 are set to -10^{-7} and:

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \left|\left| \frac{\mathbf{x}}{||\mathbf{x}||_2} - \frac{\mathbf{y}}{||\mathbf{y}||_2} \right|\right|_2 \tag{2.19}$$

The operation $||.||_2$ represents the Euclidean norm.

Chapter 3

Technical Background

In this chapter, we give an overview of machine learning algorithms, approaches and popular hacks that are commonly used for entity disambiguation and a large variety of other machine learning problems. In the following section, we describe representation learning and discuss popular word embeddings, in Section 3.2, we outline simple neural networks and popular tuning techniques, and fianlly, in Section 3.3, we characterize selected graphical models in machine learning that are used in our research.

3.1 Feature Representation in NLP

In NLP, previous approaches (Cucerzan, 2007; Milne and Witten, 2008; Ratinov et al., 2011) used high-dimensional but sparse vectors to represent the input features. These were usually used by linear machine learning models such as *support vector machines* and *logistic regression* (Goldberg, 2015). Nowadays, usually non-linear *neural network* models, trained over dense feature vectors are applied to problems of NLP.

The former techniques used so called *one-hot* representations of the input features, where features are specific linguistic inputs like a word or a part-of-speech tag ("VERB", "NOUN", ...). The dimensionality of these sparse feature vectors is the same as the number of distinct features.

In the later approachs a feature is embedded into a low-dimensional vector space (low-dimensional in comparison to the one-hot representation; usually a few hundreds dimensions), thus these dense vectors are called *embeddings*. In Figure 3.1, we show a comparison of these sparse and dense feature representations.

One of the advantages of these dense vector representations is computational speed. The frameworks used for neural networks work slow with high-dimensional sparse vectors. However, the main advantage of using representation learning, such as embeddings, is generalization. Unlike one-hot feature representation, data with similar features will have similar vector embeddings, therefore we can measure relatedness

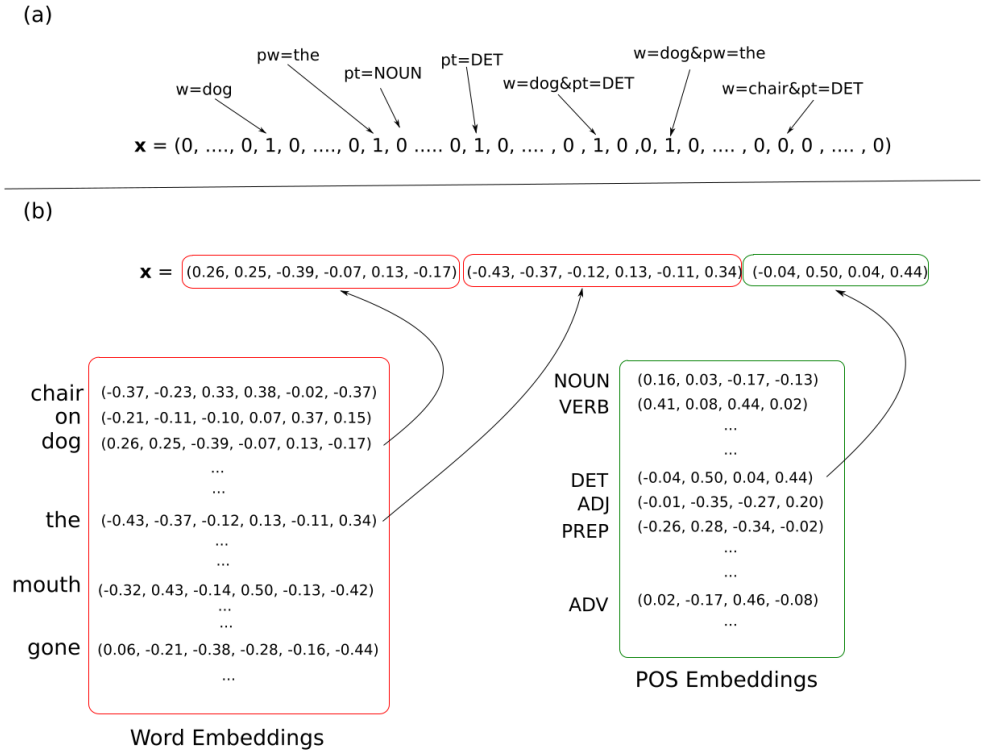

Figure 3.1: Sparse vs. dense feature representations (Goldberg, 2015). Two encodings of the same information: (a) Sparse feature vector, where each dimension represents a binary feature and feature combinations receive their own dimensions. (b) Dense embeddings-based feature vector, where each core feature is represented as a vector and feature combinations do not have an explicit encoding.

(distance) of the data in this vector space. In addition, relationships of words are present as vector offsets, so that in the embedding space, all pairs of words sharing a particular relation are related by the same constant offset. This is illustrated in Figure 3.2.

Despite the fact that one-hot representation might have some advantages, like interpretability and the fact that it is easy to use for small data, representation learning repeatedly shows state-of-the-art results in various areas across NLP (e.g. Celikyilmaz et al., 2018; Heinzerling and Strube, 2019; Akbik et al., 2019).

3.1.1 Word Embeddings

From mere one-hot encoding, TF-IDF (term frequency – inverse document frequency), to neural based architectures like Word2Vec (Mikolov et al., 2013a), GloVe (Pennington et al., 2014) to current state of the art like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), today's word embeddings have evolved from merely storing a binary status to capturing syntactic relationship and context.

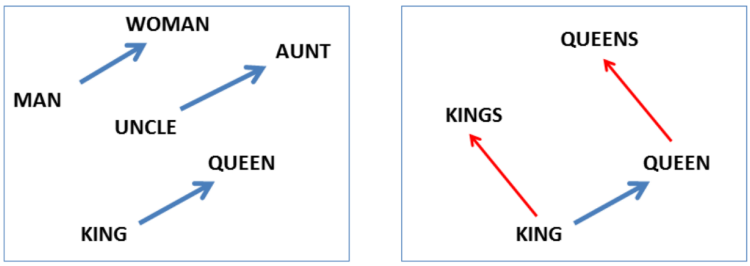

Figure 3.2: Relation offsets in embeddings (Mikolov et al., 2013b). Left panel shows vector offsets for three word pairs illustrating the gender relation. Right panel shows a different projection, and the singular/plural relation for two words. In higher-dimensional space, multiple relations can be embedded for a single word.

Word2Vec and GloVe

Word2Vec is a predictive embedding model that only takes into account the local context to predict the word vectors. GloVe, on the other hand, uses predictive neural methods to capture co-occurence information from the global context. Word2Vec and GloVe do not take into account the word order, but rather treat the context as a *bag of words*. GloVe vectors are faster to train, however, neither GloVe or Word2Vec has been shown to provide generally better results than the other one.

ELMo and BERT

ELMo and BERT generate embeddings for a word based on the context it appears in, therefore they produce slightly different embeddings for each of the word's occurrence. For example, the word *'play'* has multiple meanings, such as the verb *'to play'* or a noun for a theatre production. In standard word embeddings such as Glove or Word2Vec, each instance of the word *'play'* would have the same representation. This enables NLP models to better disambiguate between the correct meaning of a given word. BERT uses attention transformers instead of bidirectonal RNNs (recurrent neural networks), like ELMo does, to encode context. After its release, it enabled near instant state-of-the-art results for many downstream tasks.

A practical implication of the difference between these two types of embeddings is that we can use Word2Vec and GloVe vectors pretrained on a large corpus directly, using only a database with the vectors for the words. There is no need for the model itself that was used to train these vectors. However, in the case of ELMo and BERT, since they are context dependent, we need the model that was used to train the embeddings even after training, since the models generate the vectors for a word based on the context. Using ELMo or BERT embeddings for our task could potentially yield state-of-the-art results. Due to the time and computational difficulty, we leave this for future work.

3.2 Feed-Forward Neural Network

Artificial *neural networks* (NNs) are computational models of machine learning inspired by biological neural networks that can be trained (using various methods of machine learning) to perform different tasks. There are multiple types of NNs, the simplest of them are *feed-forward neural networks*.

A *perceptron*, an artificial neuron, recieves one or more weigthed inputs and sums them in order to produce output. The perceptron outputs a non-zero value only if the weighted sum exceeds a certain *threshold*. For that reason *a bias unit* (usually a value of 1) is often added to the input layer. The weighted sum (linear combination) is then passed to a non-linear *activation function* (see Equation 3.6) to produce a non-linear output.

A feed-forward neural network is composed of layers of neurons. The input layer contains the input features, equivalently, the output layer is the output of the network. A network that only consists of input and output layer is a *single-layer perceptron*, while NN model consisting of multiple layers is *a multi-layer perceptron* (MLP). The layers between the input and the output layer, are called *hidden layers*. A typical MLP with one hidden layer is shown in Figure 3.3.

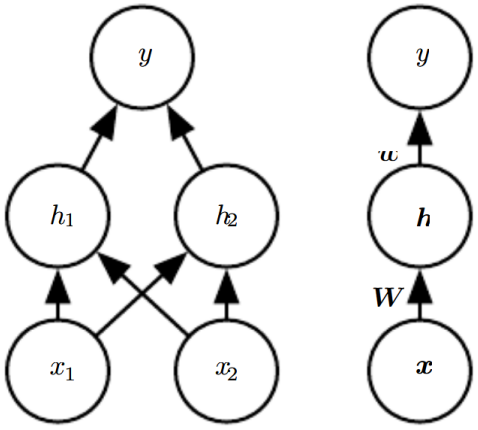

Figure 3.3: An example of a feed-forward network, drawn in two different styles (Goodfellow et al., 2016). Specifically, this is a multi-layer perceptron with an input vector $\boldsymbol{x}$*, one hidden layer* $\boldsymbol{h}$ *and a single output y. Here, we indicate that a matrix* $\boldsymbol{W}$ *describes the mapping from* $\boldsymbol{x}$ *to* $\boldsymbol{h}$*, and a vector* $\boldsymbol{w}$ *describes the mapping from* $\boldsymbol{h}$ *to y. On the left, every unit is drawn separately as a single node, while on the right side, each node represents the whole layer. This style is much more compact.*

To train a NN, we need to find such weights (parameters) of the linear combination, so that the output is close to the target value. A properly trained NN has the ability of *generalization*, which means it can predict the output of an input vector that was not present in the training data with a small error. The NN is trained after multiple *epochs*

(loops) of *forward propagation* and backward propagation of error, *the backpropagation*, over the training data.

3.2.1 Forward propagation

To evaluate the output of a neural network, we use *forward propagation*. Working with a MLP, we need to compute the activation of the layers in the direction from the input to the output layer. We will demonstrate the forward propagation on the example network in Figure 3.3. The activation of the k^{th} unit in the hidden layer is the output of the activation function f that is given the linear combination of the input vector $\mathbf{x}$ and the k^{th} vector from the weight matrix $\mathbf{V}$ as the argument:

$$h_k = f(\sum_{j=1}^{n} \mathbf{V}_{kj} x_j) \tag{3.1}$$

Similarly, the activation of the i^{th} unit in the output layer is given by the output of the activation function of the linear combination of the hidden layer and the weight matrix $\mathbf{W}$:

$$y_i = f(\sum_{k=1}^{q} \mathbf{W}_{ik} h_k) \tag{3.2}$$

Activation Function

There are various types of activation functions used in neural networks. A common type of activation function is the *logistic function* (or the *sigmoid function*) which transforms the output to the range $(0, 1)$:

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{3.3}$$

Hyperbolic tangent is also an S-shaped function, but it transforms the output into the range $(-1, 1)$:

$$tanh(x) = \frac{\exp(2x) - 1}{\exp(2x) + 1} \tag{3.4}$$

An approximation of the hyperbolic tangent *a hard tangent* is an alternative and fast to compute activation function:

$$hardtanh(x) = \begin{cases} -1 & x < -1 \\ 1 & x > 1 \\ x & \text{otherwise} \end{cases} \tag{3.5}$$

Nowadays, widely used activation function is *rectified linear unit* (ReLU) (Glorot et al., 2011) depicted in Figure 3.4, which is a simple and computationally inexpensive activation function that performs very well on various tasks. For great results,

it is often used together with the dropout regularization technique that is described in Section 3.2.4:

$$ReLU(x) = \max(0, x) = \begin{cases} 0 & x < 0 \\ x & \text{otherwise} \end{cases} \tag{3.6}$$

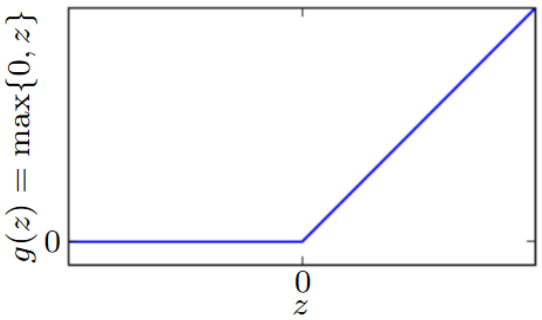

Figure 3.4: The rectified linear activation function.

The function remains very close to linear, however, in the sense that it is a piecewise linear function with two linear pieces. Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. They also preserve many of the properties that make linear models generalize well. A common principle throughout computer science is that we can build complicated systems from minimal components. Much as a Turing machine's memory needs only to be able to store 0 or 1 states, we can build a universal function approximator from rectified linear functions affine transformation from an input vector to an output scalar (Goodfellow et al., 2016).

Output Transformation

An activation function can also be applied to the output layer of a network; in that case, we are talking about the *output transformation*. One common output transformation is the *softmax* function:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{k} \exp(x_j)}; \quad X = (x_1, ..., x_k) \tag{3.7}$$

The output vector of the softmax function is a discrete probability distribution over k possible output classes, summing up to one.

3.2.2 Backpropagation

The backward propagation of errors, i.e. the backpropagation (Rumelhart et al., 1988), is a learning algorithm used for training NNs. Initially, the weights of a neural network are set randomly. Based on the desired output (target), we compute the error

of a NN by comparing it to the output and then update all weights in the network using backpropagation.

We can compute the error of the i^{th} neuron from the output layer using formula in Equation 3.8, where y_i is the output of the NN (see Eq. 3.2), d_i is the desired output (label) and f_i' is the derivative of the activation function with respect to its argument:

$$\delta_i = (d_i - y_i) f_i' \tag{3.8}$$

At first, we update the weights closer to the output layer by adding a small value to each of them. The update is computed by the formula:

$$W_{ik}(t+1) = W_{ik}(t) - \alpha \delta_i h_k \tag{3.9}$$

where W_{ik} is the weight between the k^{th} neuron in the hidden layer and the i^{th} neuron in the output layer, α is *the learning rate* parameter ($0 < \alpha < 1$), δ_i is the error of the i^{th} neuron, computed by the formula in Equation 3.8 and h_k is the k^{th} neuron in the hidden layer.

Then, we back propagate the error through the network. We compute the error of the hidden layer, as follows

$$\delta_k = (\sum_i W_{ik} \delta_i) f_k' \tag{3.10}$$

Equivalently, we update the weights between the hidden and the input layer:

$$V_{kj}(t+1) = V_{kj}(t) + \alpha \delta_k x_j \tag{3.11}$$

Overfitting

The process is repeated many times (depending on the task) and the neural network is trained. The training should be stopped before the NN (the model) starts dropping in accuracy (or other metric) on our validation (development) data. Otherwise, *overfitting* may occur. In case of overfitting, the model learns the specifics of the training data very well, but looses the ability to generalize, and thus the performance on unseen data gets worse.

Loss Function

In Equation 3.8, we would usually use a *loss funcion*, instead of a simple difference. There are many loss functions to consider while training a neural network. Each one is useful for a different type of problem. Here, we give an overview of a few common loss functions.

Mean squared error (MSE), the simplest form of a loss function, is measured as the average of the squared difference between predictions and true labels:

$$MSE = \frac{\sum_{i=1}^{n}(d_i - y_i)^2}{n} \tag{3.12}$$

where n is the number of training examples, d_i is the desired output of the i^{th} example and y_i is the prediction for i^{th} training example. MSE is usually used for *regression* problems. A common loss function for a (binary) *classification* problem is the *cross entropy* loss:

$$CrossEntropy = (1 - d_i)\log(1 - y_i) - d_i \log(y_i) \tag{3.13}$$

In some settings, we are not given supervision in terms of labels, but rather as pairs of correct x^+ and incorrect x^- items, and our goal is to score the correct items above incorrect ones. In such cases, we can use a *margin-based ranking loss*:

$$RankingLoss = [\gamma - NN(x^+) + NN(x^-)]_+ \tag{3.14}$$

where γ is the margin parameter, $NN(x)$ is the prediction for input x and $[\cdot]_+$ is the ReLU function.

3.2.3 Optimization Techniques

Gradient descent is a way to minimize an objective function $J(\theta)$ with parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\Delta J(\theta)$ with respect to the parameters. The learning rate α determines the size of the steps we take to reach a (local) minimum.

Choosing a proper learning rate can be difficult. A learning rate that is too small leads to very slow convergence and can get stuck in an undesirable local minimum, while too large learning rate can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge. It is often beneficial to adjust the learning rate during training, e.g. start with higher learning rate and decrease it. To deal with this challenge, there is a variety of different optimization algorithms that we will outline below.

Momentum is a method that helps accelerate gradient descent in the relevant direction and dampens oscillations. It does this by adding a fraction γ of the update vector of the past time step to the current update vector:

$$v_{t+1} = \mu v_t + \alpha L(W_t) \tag{3.15}$$

$$W_{t+1} = W_t + v_{t+1} \tag{3.16}$$

AdaGrad (Duchi et al., 2011) is an algorithm for gradient-based optimization that adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger

updates (i.e. high learning rates) for parameters associated with infrequent features. For this reason, it is well-suited for dealing with sparse data. For example, AdaGrad was used to train GloVe word embeddings (Pennington et al., 2014), as infrequent words require much larger updates than frequent ones.

Adagrad scales the learning rate for each parameter by dividing current gradient in update rule by the sum of previous gradients for that parameter:

$$g_{t+1} = g_t + L(W_t)^2 \tag{3.17}$$

$$W_{t+1} = W_t - \frac{\alpha}{\sqrt{g_{t+1}} + \epsilon} L(W_t)^2 \tag{3.18}$$

where $L((W_t)$ is the loss w.r.t. the weights W_t and ϵ is a smoothing term that avoids division by zero. As a result, when the gradient is very large, alpha is reduced and vice-versa.

The main weakness of AdaGrad is its accumulation of the squared gradients in the denominator. Since every added term is positive, the accumulated sum keeps growing during training. This causes the learning rate to shrink until it becomes useless. *AdaDelta* (Zeiler, 2012) and RMSprop (Tieleman and Hinton, 2012) are both extensions of AdaGrad that seeks to overcome this issue. Instead of accumulating all past squared gradients, AdaDelta restricts the window of accumulated past gradients to some fixed size w. In RMSprop the g_t term is calculated by exponentially decaying average and not the sum of gradients:

$$g_{t+1} = \gamma_2 g_t + (1 - \gamma_2) L(W_t)^2 \tag{3.19}$$

where $0 \leq \gamma_2 < 1$ is the decay rate. Here, g_t is called the *second order moment* of L. Additionally, a first order moment m_t can also be introduced:

$$m_{t+1} = \gamma_1 m_t + (1 - \gamma_1) L(W_t) \tag{3.20}$$

Adam (Kingma and Ba, 2014) uses both, the first order moment m_t and second order moment g_t. As m_t and g_t are initialized as vectors of zeros, the authors of Adam observe that they are biased towards zero. They counteract these biases by computing the first and the second order moment estimates:

$$\hat{m}_t = \frac{m_t}{1 - \gamma_1^t} \tag{3.21}$$

$$\hat{g}_t = \frac{g_t}{1 - \gamma_2^t} \tag{3.22}$$

$$\tag{3.23}$$

The final update is computed as:

$$W_{t+1} = W_t - \frac{\alpha \hat{m}_{t+1}}{\sqrt{\hat{g}_{t+1}} + \epsilon} \tag{3.24}$$

3.2.4 Regularization and Dropout

A common problem in machine learning is *overfitting*. In neural networks, overfitting results in low generalization, so that the model does not perform well on unseen data, but the performance improves on the training data. This is an adverse effect and we can mitigate it to some extent by regularization techniques.

A common regularization method is *L2 regularization*, placing a squared penalty on parameters with large values by adding the term $\frac{1}{2}\gamma||\theta||^2$ to the objective function to be minimized, where θ is the set of model parameters, $||\cdot||^2$ is the squared L2 norm (sum of squares of the values), and γ is a hyperparameter controlling the amount of regularization.

An alternative regularization method is *dropout* (Hinton et al., 2012). This method is designed to prevent the network from learning to rely on specific weights. Dropout is a very simple regularization technique that showed to be very effective. The basic idea is to randomly drop (set the values to zero) some portion (usually a half) of the neurons in the network, or in a specific layer, for each training example. It has been shown that a MLP with dropout applied on every layer is equivalent to *bayesian model averaging* (Gal and Ghahramani, 2016).

3.3 Graphical Models in Machine Learning

Probabilistic graphical model is a graph that captures the conditional dependencies between random variables. Each node of the graph is associated with a random variable, and the edges in the graph are used to encode relations between the random variables. There is a huge variety of probabilistic graphical models used for different tasks in machine learning. In this thesis, however, we will only talk about *conditional random fields* (CRFs).

3.3.1 Conditional Random Field

CRF is an undirected conditional probabilistic graphical model, often used for labeling sequential data, such as text in natural language processing and biological sequences of nucleotides or amino acids. Formaly, $G = (V, E)$ is an undirected graph modeling a conditional distribution $p(\mathbf{Y}|\mathbf{X})$, where $\mathbf{X}$ and $\mathbf{Y}$ are the random variables of observations and labels respectively. There is a node $v \in V$, corresponding to each of the random variables Y_v of $\mathbf{Y}$ that obey the *Markov property* with respect to G:

$$p(Y_v|\mathbf{X}, Y_w; v \neq w) = p(Y_v|\mathbf{X}, Y_w; v \sim w) \tag{3.25}$$

where $v \sim w$ indicates that v and w are neighbours in G.

The graphical structure of a conditional random field may be used to factorize the joint distribution over elements Y_v of $\mathbf{Y}$ into a normalized product of strictly positive, real-valued *potential functions*, derived from the notion of conditional independence[1]. In other words, we can define the CRF($\mathbf{Y}|\mathbf{X}$) model to be a normalized product of potential functions:

$$\exp\big(\sum_k \mu_k \Psi_k(Y_i, \mathbf{X}, i) + \sum_j \lambda_j \Phi_j(Y_{i-1}, Y_i, \mathbf{X}, i)\big) \tag{3.26}$$

where $\Phi_j(Y_{i-1}, Y_i, \mathbf{X}, i)$ is a transition feature function of the entire observation sequence and the labels at positions i and $i-1$, $\Psi_k(Y_i, \mathbf{X}, i)$ is a state feature function of the label at position i and the observation sequence, and μ_k and λ_j are parameters to be estimated from the training data.

The parameters are usually learned by solving *maximum likelihood estimation* using gradient-based methods. If the graph is a tree, message passing algorithms yield exact solutions for inference. Nonetheless, for general graphs, the problem of exact inference in CRF is intractable. Therefore, several algorithms can be used to obtain approximate solutions. The most common is *loopy belief propagation* (LBP).

Belief propagation (BP) is a message passing algorithm for performing inference on graphical models. It calculates the marginal distribution for each unobserved node Y_v, conditional on any observed nodes $\mathbf{X}$. Although it was originally designed for acyclic graphical models, BP can be used in general (cyclic) graphs with loops (therefore LBP). The presence of loops does not guarantee convergence, however, in most of the practical cases it does converge.

[1] A normalization factor is introduced to ensure that the product of potential functions is a valid probability distribution over the random variables represented by vertices in G.

Chapter 4

Experiments

In this chapter, we first describe standard datasets for ED for long and short documents and how we use them. Then, we report and discuss our experiments and their results. We set up the experiments in the same way as in previous research (Ganea and Hofmann, 2017; Le and Titov, 2018), run each model 5 times and report the average and the 95% confidence interval of the standard micro F1 score.

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.1}$$

The micro F1 score is simply the F1 averaged over all mentions.

4.1 Datasets

We followed two lines of research, ED for long and for short documents. In our experiments, we train the models in two different settings, on long and short documents, and evaluate the performance of each model on all of the datasets.

Our entity-mention probability map $p(e|m)$ is based on the data from Wikipedia (Feb, 2014) and it is used for candidate selection, as well as by our global model in both settings. We use Word2Vec embeddings (Mikolov et al., 2013a) pretrained[1] on the Google News dataset for our local model in Equation 2.2, GloVe (Pennington et al., 2014) embeddings trained on 840B tokens for computing the mapping function f in Equation 2.7 and Equation 2.8 and pretrained entity embeddings (Ganea and Hofmann, 2017) described in Section 2.2.2 based on the same version of Wikipedia (Feb, 2014) corpus. In this way, our KB of entities is consistent with the provided entity embeddings.

[1]Published by Word2Vec authors: `http://bit.ly/1R9Wsqr`

Furthermore, it is worth noticing that all of the embeddings we use share the same vector space with the dimension $d = 300$, allowing us to capture geometric similarities between all of them.

4.1.1 Standard Benchmark: Long Documents

For training, in the setting for long documents, we use the AIDA-train split from AIDA-CoNLL dataset (Hoffart et al., 2011) and the AIDA-A for model validation. For evaluation, we use the AIDA-B split and five other out-of-domain datasets, namely, MSNBC, AQUAINT, ACE2004, WNED-WIKI and WNED-CWEB (Guo and Barbosa, 2018)[2]. The statistics of these datasets are provided in Table 4.1. For both, long and short documents, we considered only mentions that have ground truth entities in the KB.

Dataset	**#Mentions**	**#Docs**	**Mentions per Doc**
AIDA-train	18448	946	19.5
AIDA-A (val)	4791	216	22.1
AIDA-B (test)	4485	231	19.4
MSNBC	656	20	32.8
AQUAINT	727	50	14.5
ACE2004	257	57	4.5
WNED-CWEB	11154	320	34.8
WNED-WIKI	6792	345	19.7

Table 4.1: Statistics of ED datasets with long documents.

4.1.2 Short Documents: Twitter Posts

Since there is not a stable benchmark, we explore different datasets with short documents for ED. Probably the most used in the literature is *Microposts2016* (Rizzo et al., 2016) containing posts from Twitter. However, using the current candidate selection system, we experienced difficulties with obtaining the ground truth entities from this dataset. We were only label to link ∼54% of the ground truth entities to our KB. The reason behind this is that our KB is based on Wikipedia from February 2014, while the Twitter posts (tweets) were collected in December 2015. Major topics of the tweets were, simply, yet unknown or not popular enough in 2014. For example, entities referring to *Star Wars: The Force Awakens* movie (2015) or *Donald Trump* as the president (elected in 2016) will be unknown to our KB.

[2]Available at `https://bit.ly/2gnSBLg`

For this reason, in our data, where we only keep the mentions with ground truth entities, there are much less mentions than the original data. We also only keep the documents (tweets) with at least one linkable mention. It is worth mentioning that the pretrained entity embeddings we use are also based on the same version of Wikipedia, therefore it would be necessary to train new embeddings if we added new entities to our KB. Hence, we leave the evaluation of the full Microposts2016 dataset for future work.

Apart from using (cleaned) Microposts2016, we create a new dataset for ED with short documents by processing Brian (Locke, 2009), Mena (Habib and Van Keulen, 2012) and Microposts2014 (Cano et al., 2014) datasets[3]. The resulting dataset Tw[4] is traditionally split into training, validation and testing set by splitting each of the three datasets in ratio 80 : 10 : 10 respectively, concatenating the datasets and shuffling examples within each part.

The new dataset Tw contains entity candidates generated using our $p(e|m)$. The final numbers of the cleaned Microposts2016*, Brian, Mena, Microposts2014 and Tw datasets are shown in Table 4.2.

Dataset	**#Mentions**	**#Docs**	**Mentions per Doc**
Microposts2016*-train	4905	3333	1.5
Microposts2016*-dev	148	97	1.5
Microposts2016*-test	368	286	1.3
Brian	1585	1603	1
Mena	510	162	3.1
Microposts2014	3819	2339	1.6
Tw-train	3662	2058	1.8
Tw-val	457	418	1.1
Tw-test	457	421	1.1

Table 4.2: Statistics of cleaned Microposts2016 data that is linkable to our KB, Brian, Mena and Microposts2014 and our processed dataset Tw.

In our experiments, we train our models for ED for short documents in two different settings. One setting uses Microposts2016*-train for training and Microposts2016*-val for model validaton and the other one uses Tw-train and Tw-val respectively.

[3]Collected and published at `https://github.com/badiehm/TwitterNEED`. (Habib and Van Keulen, 2016)

[4]Available at `https://github.com/lej-la/Twitter_ED`

*Microposts2016 data with only ∼54% of mentions

4.2 Training Details and Hyperparameters

We mentioned before three different settings of evaluation of our models based on the training and validation datasets, one setting for long documents and two for short documents. There are three different models that we experimented with in all three settings. The first model with multiple relations (*mul-rel*) was described in Chapter 2. We tried using another model with only a single relation (*sgl-rel*) as well, rewriting Equation 2.9 for computing the pairwise score between two entity candidates e'_i and e'_j to:

$$\Phi(e'_i, e'_j) = e'^{\top}_i \mathbf{R} e'_j \tag{4.2}$$

with only one trainable diagonal matrix $\mathbf{R}$. We further experimented with a normalized version of a single-relational model, inspired by previous research (Ganea and Hofmann, 2017). The resulting pairwise score for the single-relational normalized model (*sgl-norm*) is computed as:

$$\Phi(e'_i, e'_j) = \frac{1}{n-1} e'^{\top}_i \mathbf{R} e'_j \tag{4.3}$$

Please, note that Equation 4.3 is a special case of Equation 2.9 with $K = 1$.

The size of the context window used in the local model is $K = 100$ and $R = 25$ top contextual words are being kept by the hard attention filter in Equation 2.3. For the global model, we use the following parameter values: $\gamma = 0.01$ (see Equation 2.17), the number of LBP loops is 10, the dropout rate for f was set to 0.3, the window size of local contexts c_i for the pairwise score functions is 6. We initialize $diag(\mathbf{R}_q)$ and $diag(\mathbf{D}_q)$ by sampling from $\mathcal{N}(0, 0.1)$ for all q, except that $diag(\mathbf{R}_1)$ was sampled from $\mathcal{N}(1, 0.1)$. For the *mul-rel* model, the number of relations was $K = 3$, while the *sgl-rel* and *sgl-norm* models only use one relation. These hyperparameters were inferred by previous research (Ganea and Hofmann, 2017; Le and Titov, 2018) and tweaking them could offer a potential space for improvement of our results. Since hyperparameter tuning is expensive, we leave this for future work.

Besides these three models, we also evaluated a baseline model, that only uses $p(e|m)$ to pick the top entity candidate for each mention, on all datasets.

4.3 Results

In this section, we compare the results of our models in 3 different settings and we discuss the results of our experiments. Furthermore, we compare our best models to results reported by related research.

4.3.1 AIDA Corpus

Here, we evaluate our models trained in the setting for long documents. The training dataset in this case is AIDA-train and the validation dataset is AIDA-A. We evaluate the models on all datasets (with both, long and short documents).

Dataset	mul-rel	sgl-rel	sgl-norm	BASELINE
AIDA-A	**90.66 ± 0.2**	83.87 ± 1.8	90.35 ± 0.1	73.73
AIDA-B	**91.66 ± 0.2**	83.79 ± 3.1	91.64 ± 0.2	71.79
MSNBC	93.38 ± 0.6	91.91 ± 1.4	**93.70 ± 0.8**	89.67
AQUAINT	86.92 ± 0.9	86.41 ± 0.8	**88.30 ± 1.4**	84.48
ACE2004	**88.73 ± 0.4**	87.32 ± 0.4	88.40 ± 2.1	87.32
WNED-CWEB	77.47 ± 0.4	74.60 ± 0.8	**77.77 ± 0.2**	69.74
WNED-WIKI	**76.84 ± 0.5**	73.30 ± 1.8	76.54 ± 0.9	63.96
Tw-train	69.83 ± 2.2	75.10 ± 2.0	71.91 ± 10.9	**80.12**
Tw-val	63.57 ± 2.4	68.36 ± 5.0	65.65 ± 13.8	**80.96**
Tw-test	63.84 ± 4.0	67.09 ± 4.6	64.55 ± 12.7	**77.24**
Micro2016*-train	72.03 ± 3.3	75.77 ± 2.0	72.72 ± 13.2	**81.57**
Micro2016*-dev	56.25 ± 13.1	72.30 ± 7.9	50.68 ± 25.2	**90.54**
Micro2016*-test	54.42 ± 5.1	70.43 ± 3.1	56.34 ± 30.1	**83.15**
Micro2014	57.55 ± 2.3	64.96 ± 1.7	60.10 ± 11.5	**69.05**
Mena	79.89 ± 2.2	81.80 ± 1.3	80.86 ± 4.0	**83.32**
Brian	60.69 ± 1.0	59.28 ± 0.3	**60.97 ± 3.6**	59.64

Table 4.3: Micro F1 scores of models trained on AIDA corpus (long documents). The best scores for each dataset are depicted in bold.

Please, note that the model *mul-rel* trained on AIDA corresponds to the best model of Le and Titov (2018) and the model *sgl-norm* trained on AIDA corresponds to the best model of Ganea and Hofmann (2017).

The results in Table 4.3 show, that both *mul-rel* and *sgl-norm* models have very similar performance. The scores of all models trained on AIDA corpus are quite high for long documents, with respect to the baseline. However, such models perform very poorly on short documents (worse than our $p(e|m)$ baseline) and have very wide 95% confidence intervals. This behaviour is expected, since datasets with short documents lack the amount of context and have lower number of mentions per document.

4.3.2 Tw Corpus

In this setting, we use our new dataset Tw, that contains Twitter posts, to train (Tw-train) and validate (Tw-val) our models. We evaluate the models on all datasets (with both, long and short documents).

Based on the results in Table 4.4, we claim that our normalized single-relational model (*sgl-norm*) performs the best for all datasets, out of the models we trained in this setting. Furthermore, we can see that not only the model performs well on short documents, but also works reasonably well for solving ED for long documents, even though the 95% confidence interval are wider for long documents. This might not be very intuitive, since the training data is much simpler, with shorter context and fewer mentions per document than the testing data with long documents. To our knowledge, we are the first to observe such phenomenon, and we will discuss this further in Section 4.4.

Dataset	mul-rel	sgl-rel	sgl-norm	BASELINE
Tw-train	**87.23 ± 0.7**	84.42 ± 1.4	87.16 ± 0.9	80.12
Tw-val	85.34 ± 0.5	82.58 ± 1.2	**85.82 ± 1.0**	80.96
Tw-test	84.86 ± 0.9	81.14 ± 1.0	**84.99 ± 0.8**	77.24
Micro2016*-train	85.05 ± 0.5	83.86 ± 0.7	**85.56 ± 0.9**	81.57
Micro2016*-dev	90.41 ± 1.4	88.51 ± 5.0	**91.62 ± 1.4**	90.54
Micro2016*-test	84.62 ± 0.6	81.20 ± 4.8	**85.49 ± 1.6**	83.15
Micro2014	74.46 ± 0.5	72.15 ± 0.8	**74.56 ± 0.8**	69.05
Mena	**84.54 ± 1.0**	84.14 ± 0.3	84.42 ± 0.8	83.32
Brian	66.71 ± 0.2	63.28 ± 1.9	**66.74 ± 0.5**	59.64
AIDA-A	81.30 ± 4.7	77.66 ± 1.6	**83.62 ± 6.1**	73.73
AIDA-B	80.86 ± 5.7	77.49 ± 1.9	**84.36 ± 7.7**	71.79
MSNBC	91.81 ± 2.3	91.54 ± 0.4	**92.30 ± 1.9**	89.67
AQUAINT	88.48 ± 4.8	88.28 ± 0.7	**89.29 ± 4.3**	84.48
ACE2004	88.21 ± 0.9	88.13 ± 0.5	**88.37 ± 1.1**	87.32
WNED-CWEB	74.49 ± 4.7	72.93 ± 0.6	**75.51 ± 4.7**	69.74
WNED-WIKI	71.16 ± 6.9	68.16 ± 0.8	**72.22 ± 6.6**	63.96

Table 4.4: Micro F1 scores of models trained on Tw corpus (short documents). The best scores for each dataset are depicted in bold.

Our experiments suggest that using the multi-relational approach might not be as beneficial as claimed in previous research (Le and Titov, 2018), for neither long nor short documents. The results of this approach (*mul-rel*) were not significantly higher for long documents than the normalized model with a single relation (*sgl-norm*) and

were worse for short documents.

4.3.3 Microposts2016 Corpus

Finally, we perform experiments on our models trained on the cleaned Microposts2016* data. Namely, we train the models using the Microposts2016*-train dataset and validate them using Microposts2016*-dev. The results are shown in Table 4.5.

Dataset	mul-rel	sgl-rel	sgl-norm	BASELINE
Micro2016*-train	**85.09 ± 2.8**	83.69 ± 2.8	84.36 ± 3.9	81.57
Micro2016*-dev	91.22 ± 1.7	91.89 ± 1.7	**92.12 ± 1.0**	90.54
Micro2016*-test	86.14 ± 1.2	85.42 ± 5.5	**87.50 ± 5.5**	83.15
Tw-train	81.71 ± 2.5	**81.80 ± 0.3**	81.66 ± 2.0	80.12
Tw-val	81.11 ± 2.5	80.16 ± 3.0	**81.69 ± 1.1**	80.96
Tw-test	**79.65 ± 5.2**	79.29 ± 1.1	79.14 ± 5.1	77.24
Micro2014	70.33 ± 2.5	70.46 ± 1.0	**70.59 ± 2.2**	69.05
Mena	83.45 ± 2.9	83.59 ± 0.6	**84.07 ± 1.1**	83.32
Brian	**61.74 ± 0.2**	60.76 ± 3.3	61.04 ± 0.3	59.64
AIDA-A	79.53 ± 1.9	77.42 ± 6.6	**79.78 ± 7.7**	73.73
AIDA-B	78.63 ± 1.8	76.31 ± 7.2	**79.75 ± 11.0**	71.79
MSNBC	**92.22 ± 0.2**	91.46 ± 0.8	92.17 ± 1.2	89.67
AQUAINT	**89.23 ± 1.3**	87.37 ± 2.9	89.09 ± 0.9	84.48
ACE2004	**88.13 ± 1.0**	87.86 ± 1.5	87.99 ± 1.2	87.32
WNED-CWEB	**74.82 ± 0.7**	72.45 ± 3.2	74.73 ± 2.5	69.74
WNED-WIKI	**71.20 ± 1.5**	67.80 ± 4.7	70.58 ± 6.2	63.96

Table 4.5: Micro F1 scores of models trained on Microposts2016 corpus (short documents). The best scores for each dataset are depicted in bold.*

An important thing to notice is that the 95% confidence intervals for most models and datasets are wide in this setting. This suggests that the cleaned Microposts2016* corpus might not be sufficient for training such models. Thus, we can not draw many conclusions out of these results.

It is interesting that, even in this setting, the performance of the models on long documents is better than the $p(e|m)$ baseline, as opposed to the performance of the models trained on long documents on data with short documents.

*Microposts2016 data with only ∼54% of mentions.

*Microposts2016 data with only ∼54% of mentions.

4.3.4 State of The Art Comparison

Since the *sgl-norm* model is less complex than the multi-relational one and its performance is, at least, comparable (or better) in all of our experiments, we choose to proceed our research using it as our best model.

As we mentioned before, the field of ED for short documents lacks a stable benchmark, besides the research conducted on Microposts2016. While we show the results reported on this dataset in Table 4.6, our results can not be compared to them, since we only had 54% of the mentions in this dataset linkable to our KB (see Section 4.1.2).

Method	**Microposts 2016**
NEEL (Rizzo et al., 2016)	53.6 micro F1
PBOH (Ganea et al., 2016)	72.1 micro F1
#KEA (Waitelonis and Sack, 2016)	75.2 micro F1
S-MART (Yang and Chang, 2016)	**81.1 micro F1**

Table 4.6: Results on Microposts2016 corpus reported by previous research.

The paper that released the Microposts2016 dataset (Rizzo et al., 2016) reported various results of models participating in their challenge. However, the best was their baseline model NEEL and not one of the competing models. Other results we show here are: PBOH (Ganea et al., 2016) model evaluated on short texts by later research (Zhu and Iglesias, 2018), #KEA model (Waitelonis and Sack, 2016) and the best reported results on Microposts2016 using S-MART model (Yang and Chang, 2016).

Our results on the cleaned Microposts2016* data report 83.2 micro F1 using the $p(e|m)$ baseline method and **87.5 micro F1** using the *sgl-norm* model trained on Microposts2016*-train dataset. We will explore, in our future research, whether our model could be better than the current state of the art for the full Microposts2016 data.

Methods that reported results on Microposts2014 dataset usually followed the end-to-end entity linking approach. The best one was a model by Microsoft that won the Microposts2014 entity linking challenge (Cano et al., 2014). The results reported for ED on Microposts2014 was a model based on dominant entity candidates (DEC) (Feng et al., 2018).

Results in Table 4.7 show, that even our baseline method is comparable to the results reported on this dataset. Our best model for Microposts2014 is the *sgl-norm* model trained on Tw corpus.

The last results are reported by the TwitterNEED model (Habib and Van Keulen, 2016) on Brian and Mena datasets. However, this method solves EL, instead of ED.

*Microposts2016 data with only ∼54% of mentions

Method (EL)	**Microposts 2014**
Microsoft (Cano et al., 2014)	70.1 micro F1
Method (ED)	**Microposts 2014**
DEC (Feng et al., 2018)	53.9 micro F1
$p(e\|m)$ baseline	69.1 micro F1
sgl-norm trained on Tw	**74.6 micro F1**

Table 4.7: Comparison of our and other reported results on Microposts2014 corpus.

Method (EL)	**Brian**	**Mena**
TwitterNEED (Habib and Van Keulen, 2016)	55.5 micro F1	70.1 micro F1
Method (ED)	**Brian**	**Mena**
$p(e\|m)$ baseline	59.6 micro F1	83.3 micro F1
sgl-norm trained on Tw	**66.7 micro F1**	**84.4 micro F1**

Table 4.8: Comparison of the results on Brian and Mena datasets.

We can see that, in many cases, our $p(e|m)$ baseline method is comparable to the reported results. This is because of leveraging additional statistical data from Wikipedia, which previous methods did not use. Our best model *sgl-norm* further improves the results.

4.4 Discussion

In our results, we observed that the models trained on short documents performed quite well on long documents, however, when we trained the models on long documents, the results were worse than our baseline. Our insight behind this is that the models trained on long documents learn to make decisions based on features that are more robust in long documents, such as long context and other mentions in the document, which are not so reliable in short documents, and perform worse than our baseline. However, the models trained on short documents have to work with less robust features within the document and rely more on features that come from additional sources, like representation learning and the $p(e|m)$ entity-mention probability map. These features are also present for the models trained on long documents, hence, they can still make good decisions.

To find out how important the pretrained embeddings are for our model, we performed experiments and randomly initialized the word and the entity embeddings, instead of using the pretrained ones. The results are shown in Table 4.9. Interestingly, we found out that the micro F1 scores dropped significantly, especially when

Embeddings	Tw	AIDA
Pretrained	84.99	91.64
All Randomized	81.62	74.49
Randomized Word	84.68	85.99
Randomized Entity	81.47	74.45
BASELINE	77.24	71.79

Table 4.9: Micro F1 score of sgl-norm model trained using randomly initialized embeddings. The reported score of the model trained on Tw corpus is the evaluation score of Tw-test. The score of the model trained on AIDA corpus is the score on the AIDA-B dataset.

randomizing the entity embeddings. This tells us that models for solving ED definitely benefit from using representation learning. Randomizing only the word embeddings decreased the performance on AIDA corpus with long documents, however, it did not make such a difference for Tw corpus with short documents. This might be a sign that the decisions for short documents are not based on the local context.

Conclusion

Looking into two different types of problems in ED, we have leveraged the state-of-the-art approaches for solving ED in long documents for short documents. Our system does not rely on hand-crafted features or Knowledge Graphs with relation data. On the contrary, we have shown the benefits of using representation learning and bringing additional statistical data into the decision process. Especially, entity embeddings have been a very informative data source for ED in short documents. The candidate selection process that leveraged high volume data from external sources guaranteed high quality entity candidates. To our knowledge, we are the first to use such information for ED in short texts. The proof of its profit is that just our simple baseline method, which always selects the top entity candidate, achieves competitive results to the state of the art for short documents.

In our experiments, we found out that modeling multiple relations between entities is not beneficial for our task and the trade-off between performance and complexity for long documents is high. Therefore, we decided to use only a single relation parameter to model the relationship between entities in the same document. Our model has shown competitive results to the state of the art, outperforming the current solutions on all available datasets for ED in short documents. We also found out that using models trained on long documents for ED on short documents ends up with score lower than our baseline. However, when using models trained on short documents for long documents, the performance is reasonable and the scores significantly higher than our baseline.

Moreover, we have introduced a novel dataset Tw for ED for short texts, containing posts from Twitter, that we published in standard formats used in ED. The data contains up to 20 entity candidates for each mention based on our entity-mention conditional probability map $p(e|m)$.

In future work, we would like to evaluate our approach on the full Microposts2016 data. For this, we will have to update our entity-mention map $p(e|m)$ using a newer version of Wikipedia and also train a new set of entity embeddings for new entities.

Furthermore, we would like to experiment with using contextual word embeddings, such as ELMo and BERT, instead of classic Word2Vec and GloVe word embeddings that we used in our model. It would be very interesting to extend this idea for training

entity embeddings, as this branch of research often opens the door to a generalizable state-of-the-art performance.

Bibliography

Akbik, A., Bergmann, T., and Vollgraf, R. (2019). Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Cano, A. E., Rizzo, G., Varga, A., Rowe, M., Stankovic, M., and Dadzie, A.-S. (2014). Making sense of microposts:(# microposts2014) named entity extraction & linking challenge. In *CEUR Workshop Proceedings*, volume 1141, pages 54–60.

Cao, Y., Li, J., Guo, X., Bai, S., Ji, H., and Tang, J. (2015). Name list only? target entity disambiguation in short texts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 654–664.

Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*.

Cheng, X. and Roth, D. (2013). Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716.

Delpeuch, A. (2019). Opentapioca: Lightweight entity linking for wikidata. *arXiv preprint arXiv:1904.09131*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Fang, Z., Cao, Y., Li, Q., Zhang, D., Zhang, Z., and Liu, Y. (2019). Joint entity linking with deep reinforcement learning. In *The World Wide Web Conference*, pages 438–447. ACM.

Feng, Y., Zarrinkalam, F., Bagheri, E., Fani, H., and Al-Obeidat, F. (2018). Entity linking of tweets based on dominant entity candidates. *Social Network Analysis and Mining*, 8(1):46.

Ferragina, P. and Scaiella, U. (2010). Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International conference on Information and Knowledge Management*, pages 1625–1628. ACM.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.

Ganea, O.-E., Ganea, M., Lucchi, A., Eickhoff, C., and Hofmann, T. (2016). Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web*, pages 927–938. International World Wide Web Conferences Steering Committee.

Ganea, O.-E. and Hofmann, T. (2017). Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*.

Globerson, A., Lazic, N., Chakrabarti, S., Subramanya, A., Ringaard, M., and Pereira, F. (2016). Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Goldberg, Y. (2015). A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Guo, Z. and Barbosa, D. (2018). Robust named entity disambiguation with random walks. *Semantic Web*, 9(4):459–479.

Habib, M. B. and Van Keulen, M. (2012). Unsupervised improvement of named entity extraction in short informal context using disambiguation clues. In *SWAIE*, pages 1–10.

Habib, M. B. and Van Keulen, M. (2016). Twitterneed: A hybrid approach for named entity extraction and disambiguation for tweet. *Natural Language Engineering*, 22(3):423–456.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., and Wang, H. (2013). Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–34.

Heinzerling, B. and Strube, M. (2019). Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. *arXiv preprint arXiv:1906.01569*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Houlsby, N. and Ciaramita, M. (2014). A scalable gibbs sampler for probabilistic entity linking. In *European Conference on Information Retrieval*, pages 335–346. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kolitsas, N., Ganea, O.-E., and Hofmann, T. (2018). End-to-end neural entity linking. *arXiv preprint arXiv:1808.07699*.

Lazic, N., Subramanya, A., Ringgaard, M., and Pereira, F. (2015). Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

Le, P. and Titov, I. (2018). Improving entity linking by modeling latent relations between mentions. *arXiv preprint arXiv:1804.10637*.

Locke, B. W. (2009). Named entity recognition: Adapting to microblogging. *Master's Thesis, University of Colorado*.

Mihalcea, R. and Csomai, A. (2007). Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the 16th ACM conference on Conference on Information and Knowledge Management*, pages 233–242. ACM.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Milne, D. and Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 509–518. ACM.

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

Pershina, M., He, Y., and Grishman, R. (2015). Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Phan, M. C., Sun, A., Tay, Y., Han, J., and Li, C. (2018). Pair-linking for collective entity disambiguation: Two could be better than all. *IEEE Transactions on Knowledge and Data Engineering*, 31(7):1383–1396.

Pilz, A. and Paaß, G. (2011). From names to entities using thematic context distance. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 857–866. ACM.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.

Raiman, J. R. and Raiman, O. M. (2018). Deeptype: multilingual entity linking by neural type system evolution. In *32nd AAAI Conference on Artificial Intelligence*.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Rizzo, G., van Erp, M., Plu, J., and Troncy, R. (2016). Making Sense of Microposts (#Microposts2016) Named Entity rEcognition and Linking (NEEL) Challenge. In *6th Workshop on Making Sense of Microposts (#Microposts2016)*, pages 50–59.

Röder, M., Usbeck, R., and Ngonga Ngomo, A.-C. (2018). Gerbil–benchmarking named entity recognition and linking consistently. *Semantic Web*, 9(5):605–625.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1.

Sakor, A., Mulang, I. O., Singh, K., Shekarpour, S., Vidal, M. E., Lehmann, J., and Auer, S. (2019). Old is gold: linguistic driven approach for entity and relation linking of short text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2336–2346.

Spitkovsky, V. I. and Chang, A. X. (2012). A cross-lingual dictionary for english wikipedia concepts. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012), Istanbul, Turkey,May 2012.*

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

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledge base. *Communications of the ACM*, 57(10):78–85.

Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305.

Waitelonis, J. and Sack, H. (2016). Named entity linking in# tweets with kea. In *# Microposts*, pages 61–63.

Yamada, I., Shindo, H., Takeda, H., and Takefuji, Y. (2016). Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.

Yamada, I., Shindo, H., Takeda, H., and Takefuji, Y. (2017). Learning distributed representations of texts and entities from knowledge base. *Transactions of the Association for Computational Linguistics*, 5:397–411.

Yang, Y. and Chang, M.-W. (2016). S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. *arXiv preprint arXiv:1609.08075*.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhu, G. and Iglesias, C. A. (2018). Exploiting semantic similarity for named entity disambiguation in knowledge graphs. *Expert Systems with Applications*, 101:8–24.