Entity Linking Meets Deep Learning: Techniques and Solutions

Wei Shen, Yuhan Li, Yinan Liu, Jiawei Han, *Fellow, IEEE*, Jianyong Wang, *Fellow, IEEE*, Xiaojie Yuan

Abstract—Entity linking (EL) is the process of linking entity mentions appearing in web text with their corresponding entities in a knowledge base. EL plays an important role in the fields of knowledge engineering and data mining, underlying a variety of downstream applications such as knowledge base population, content analysis, relation extraction, and question answering. In recent years, deep learning (DL), which has achieved tremendous success in various domains, has also been leveraged in EL methods to surpass traditional machine learning based methods and yield the state-of-the-art performance. In this survey, we present a comprehensive review and analysis of existing DL based EL methods. First of all, we propose a new taxonomy, which organizes existing DL based EL methods using three axes: embedding, feature, and algorithm. Then we systematically survey the representative EL methods along the three axes of the taxonomy. Later, we introduce ten commonly used EL data sets and give a quantitative performance analysis of DL based EL methods over these data sets. Finally, we discuss the remaining limitations of existing methods and highlight some promising future directions.

Index Terms—Entity linking, deep learning, entity disambiguation, knowledge base

1 Introduction

THE rapid growth of web data contains an overwhelming amount of information and knowledge. Natural language is one of the most important forms of web data. However, natural language is ambiguous, especially for named entities which appear in it frequently. In addition, with the development of information extraction (IE) techniques, a growing number of high-quality, large-scale, and machine-readable knowledge bases (KBs) have been constructed recently, such as YAGO [1], DBpedia [2], Freebase [3], and Probase [4]. These KBs contain tens of millions of named entities and billions of relational facts between named entities. Bridging web text data and KBs is very helpful for understanding the ambiguous natural language on the web and enriching the existing KBs. To achieve this goal, entity linking (EL) is a fundamental task which needs to be solved.

What is entity linking? Entity linking is the task to link entity mentions appearing in web text with their corresponding entities in a KB [5]. Figure 1 shows an illustration for the entity linking task. It acts an important pre-processing step for many downstream applications, such as question answering [6], relation extraction [7], knowledge base population (KBP) [8], and content analysis [9]. For example, KBP is the task of enriching existing KBs with newly extracted facts from the text. Before the enrichment, an EL system is needed to map entity mentions associated with facts to their corresponding named entities in a KB.

Entity linking is challenging due to the name variation and entity ambiguity. On one hand, a named entity may have many different surface forms (e.g., full name, partial name, nickname, alias, and abbreviation). As the example shown in Figure 1, entity mentions "NYC" and "Big Apple" are the abbreviation and nickname of the named entity "New York City", respectively. On the other hand, an entity mention may refer to many different named entities. For the example in Figure 1, the entity mention "MJ" may refer to an American professional basketball player named "Michael Jordan", an American recording artist named "Michael Jackson", an American singer named "Mj Rodriguez", or many other named entities which could be referred to as "MJ".

Traditional machine learning based entity linking. Many traditional machine learning (ML) based EL methods have been comprehensively reviewed in a survey [5]. These ML based EL methods mainly leverage manually designed features of entity popularity, local context compatibility, and document-level global coherence of referring entities. Their entity ranking techniques can be broadly divided into two categories: unsupervised and supervised ranking methods. Unsupervised ranking methods include vector space model based methods [10] and information retrieval based methods [11]. Supervised ranking methods mainly include binary classification methods [12], learning to rank methods [13], probabilistic methods [14], and graph-based methods [15].

Most traditional ML based EL methods follow the popular two-step procedure, which firstly extracts some hand-crafted features and then feeds those features to an entity ranking method to make the final linking predictions. However, these methods have two limitations: (1) features that lead to good performance require a lot of careful and tedious feature engineering; (2) generalizing the trained entity linking model to other KBs or domains is difficult due to the strong dependence on the specific KB and domain knowledge in the process of designing features.

Deep learning based entity linking. With the success of deep learning in various domains, deep learning (DL) based EL methods have been proposed and attracted significant attention recently [16]. Compared with traditional machine learning, deep learning

- *W. Shen, Y. Li, Y. Liu and X. Yuan are with the TKLNDST, College of Computer Science, Nankai University, Tianjin 300350, China. E-mail: {shenwei, yuanxj}@nankai.edu.cn, {yuhanli, liuyn}@mail.nankai.edu.cn.*
- *J. Han is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: hanj@illinois.edu.*
- *J. Wang is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, and also with the Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou 221009, China. E-mail: jianyong@tsinghua.edu.cn.*

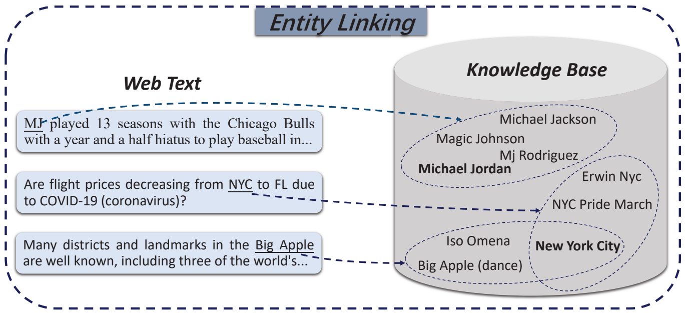

Entity Linking
Web Text
Knowledge Base
MJ played 13 seasons with the Chicago Bulls with a year and a half hiatus to play baseball in...
Are flight prices decreasing from NYC to FL due to COVID-19 (coronavirus)?
Many districts and landmarks in the Big Apple are well known, including three of the world's...
Michael Jackson
Magic Johnson
Mj Rodriguez
Michael Jordan
Erwin Nyc
NYC Pride March
Iso Omena
New York City
Big Apple (dance)

Fig. 1: An illustration for the task of entity linking. Entity mentions detected from web text are underlined; Candidate mapping entities in a knowledge base for each entity mention are shown via a dashed arrow line and circle; Their correct mapping entities (a.k.a. gold mapping entities) are in **boldface**.

can automatically learn important features and is more transferable from one domain to another. For example, DL based EL methods are able to obtain vector representations of the words via a pre-trained language model such as Word2Vec [17], [18] and GloVe [19]. This neural encoding can be directly used as semantic and syntactic features, which are then fed into neural network architectures, such as recurrent neural networks (RNNs) [20], [21], convolutional neural networks (CNNs) [22], attention [23], and Transformers [24] to capture more sophisticated long-distance feature representations.

Motivations. In the past few years, DL based EL methods with minimal feature engineering have been flourishing as shown in Figure 2. Deep learning has become the mainstream framework for the entity linking methods that achieve the state-of-the-art performance. In the previous comprehensive survey [5] of entity linking published in 2015, we reviewed and analyzed a variety of traditional ML based EL systems, which are mainly based on manually designed features without the use of deep learning. This prompts us to write this survey to systematically review and summarize the current status of deep learning techniques for entity linking. Our intended audience includes researchers, developers, and practitioners in related fields who are interested in entity linking. We hope that this survey paper will help them get a clear overall picture of current DL based EL methods and guide them to the right way to start with entity linking research and practice.

Contributions. Overall, the contributions of this survey can be summarized as follows.

- ***A comprehensive review.*** We conduct a comprehensive survey to present a thorough overview and analysis of DL based EL methods.
- ***A new taxonomy.*** We propose a new taxonomy, which organizes the techniques for ranking candidate entities in existing DL based EL methods using three axes: (1) embedding; (2) feature; (3) algorithm.
- ***A quantitative analysis.*** We present a quantitative analysis on the performance of the DL based EL methods on a variety of data sets.
- ***Some future directions.*** We discuss the remaining limitations of existing entity linking methods and point out possible future directions.

The remainder of this survey is organized as follows. We first give a formal formulation of the entity linking task in Section 2.

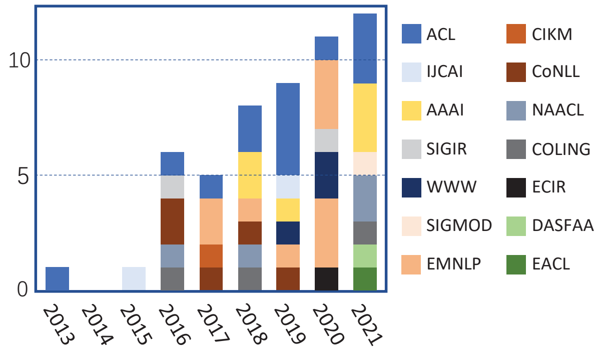


Fig. 2: Statistics of DL based EL publications in peer-reviewed venues.

Then we explain our new taxonomy of DL based EL methods in Section 3. We present a comprehensive survey along the three axes of the new taxonomy (i.e., embedding, feature, and algorithm) in Sections 4, 5, and 6 respectively. Section 7 introduces ten commonly used entity linking data sets as well as evaluation metrics, and presents a quantitative performance analysis of the DL based EL methods. Section 8 discusses the main limitations and future directions, and in Section 9 we finally give a conclusion.

2 Problem Overview

An entity mention is a token sequence in text which potentially refers to some named entity. A named entity is a word or a phrase that is explicitly defined by KBs with a unique identifier, and it could be a real-world object (e.g., organization and individual) or an abstract concept (e.g., song and event). Most of the existing EL works assume that entity mention boundaries are provided by users or mention detection systems, such as Named Entity Recognition (NER) tools, which can identify the boundaries of named entities in text automatically. Then we formally state the task of EL as follows.

Definition (Entity Linking). *Given a document D containing a set of recognized entity mentions $M = \{m_1, m_2, ..., m_{|M|}\}$ and a target KB containing a set of named entities $E = \{e_1, e_2, ..., e_{|E|}\}$, the goal is to map each entity mention m_i in M to its gold mapping entity e_i in E.*

It is possible that some entity mention's gold mapping entity does not exist in the target KB, which is called "unlinkable entity mention". A special label NIL is usually used to describe this kind of unlinkable entity mention.

As surveyed in [5], a typical EL system often consists of three stages: (1) *candidate entity generation* stage can generate a candidate mapping entity set for each entity mention using a name dictionary and web information; (2) *candidate entity ranking* stage can leverage different kinds of evidence to rank the candidate entities; (3) *unlinkable mention prediction* stage can validate whether the entity mention should be labeled as NIL. As the techniques used in the first and third stages have not changed much in recent years, in this survey we mainly concentrate on the *candidate entity ranking* stage and discuss how different neural architectures and features help to rank entities. For the technical details of approaches used in the first and third stages, you could refer to our previous survey [5].

3 A Taxonomy of DL Based EL Methods

In this section, we first introduce basic concepts of DL based EL methods and explain why EL methods can benefit from DL. Next, we give a brief introduction of the newly proposed taxonomy.

3.1 Why Choose Deep Learning?

DL is an artificial intelligence function that imitates the working mechanism of the human brain in processing information and creating patterns. It utilizes a cascade of multiple layers of non-linear processing units (i.e., neurons) for feature extraction and transformation by adjusting the connection weights between units, which can be regarded as resembling the learning behavior of a human brain. DL models have been applied widely and successfully to numerous tasks in the fields of computer vision (CV) and natural language process (NLP) since the DL based model AlexNet [25] won the ImageNet competition by a big margin in 2012.

The most basic model in DL corresponds to a fully connected layer [26]. It takes vectors $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ from the input layer as input, and outputs a value $y = f(\sum_{i=1}^{n} \mathcal{W}_i \mathbf{x}_i + b)$ by a non-linear activation function $f(\cdot)$ at the output layer, where $\mathcal{W}_i$ are weights, b is the bias, and the common choice of $f(\cdot)$ is *tanh* function, *sigmoid* function, or *ReLU* function [27]. Multi-layer neural network is simply a generalization of this basic model where neural network layers are stacked in sequence. A multi-layer neural network is also called a multi-layer perceptron (MLP).

Compared with traditional ML, DL has three main strengths to solve the EL task. First and foremost, DL can utilize the given documents and KBs to automatically discover multiple levels of distributed representations to help disambiguation without human intervention [28], while traditional ML based methods require considerable feature engineering and analysis. Second, DL is more transferable, which means that deep neural networks can learn more transferable representations that disentangle the exploratory factors of variations underlying the data samples and group features hierarchically in accordance with their relatedness to invariant factors [29]. Finally, DL is able to learn feature representations and perform classification or regression in an end-to-end style, which possibly motivates complex and advanced EL methods. Along with these advantages of DL, a lot of DL based EL methods have been proposed in recent years and achieved the state-of-the-art performance.

3.2 Basic Structure of Our Taxonomy

Previous survey [5] roughly divides the candidate entity ranking methods into two categories: supervised ranking methods and unsupervised ranking methods. Different from the previous survey, we propose a new taxonomy of the techniques in the *candidate entity ranking* stage of DL based EL methods, which contains three steps as follows:

- ***Embedding.*** Embedding can use low-dimensional and dense vectors to implicitly represent the semantic and syntactic properties of natural language, which is also called distributed representation. Generally, DL models need embeddings as input [30]. Thus, a key point in applying DL to EL is the embeddings of different inputs (e.g., mention surface form, entity context, and entity description). Lots of embedding techniques have been utilized by existing DL based EL methods, and we divide them into four categories: (1) word embedding; (2) mention embedding; (3) entity embedding; (4) alignment embedding.
- ***Feature.*** Features intend to measure the similarity between the entity mention and the candidate entity in different aspects in the form of a vector or a score. According to the previous survey [5], almost all traditional EL methods are based on quantitative features of mention-entity popularity, mention-entity similarity, and entity-entity topical coherence. These features are still widely utilized by researchers until now, and they can benefit from DL according to the following two aspects: (1) features can be directly learned by neural models without manual feature engineering. For example, Wu et al. [31] learned an entity-entity topical coherence feature vector for each candidate entity automatically via a dynamic Graph Convolutional Network architecture; (2) embeddings introduced above can be leveraged to generate diverse features. For example, unlike traditional methods preferring to represent the context as a bag-of-words, many DL methods could leverage the embedding of text to calculate the similarity between mention context and entity description as a context similarity feature. In this survey, we concentrate on the following five principal features for entity linking: (1) prior popularity; (2) surface form similarity; (3) type similarity; (4) context similarity; (5) topical coherence.
- ***Algorithm.*** Algorithm is the final step in the *candidate entity ranking* stage. It takes features as input and outputs the final entity linking result. The commonly used algorithms in DL based EL methods include MLP, PageRank, graph regularization, and reinforcement learning. The output of the algorithm is a score list of all candidate entities for each entity mention. We summarize such algorithms into three groups, namely, MLP, graph-based algorithms, and reinforcement learning (RL).

Although the implementation details of DL based EL methods may vary considerably, their general steps for ranking entities are very similar. Given an entity mention in a document and a set of candidate entities, various embeddings are generated at first. Then features can be calculated using embeddings. Finally, features are fed into algorithms to rank the candidate entities and get the final linking result. It is noted that these three steps are in a sequence manner, which means the output of the previous step is used as the input of the current step. In the following, we review the main techniques used in these three steps in detail. Table 1 summarizes the step choices of DL based EL methods according to the proposed taxonomy.

4 Embedding

The core idea of embedding is to represent the meaning of a piece of natural language by using a low-dimensional real-valued dense vector. Based on the given documents and KBs, diverse embedding techniques could be leveraged to capture the linguistic patterns and common sense knowledge in text, such as semantic roles, syntactic structures, and lexical meanings. In this section, we introduce four kinds of embeddings used in DL based EL methods: (1) word embedding; (2) mention embedding; (3) entity embedding; (4) alignment embedding. Considering the strong correlation between context embedding and context similarity feature, we will introduce the context embedding in Section 5.4.1.

4.1 Word Embedding

A word embedding can map words from a vocabulary to vectors of real numbers to represent their meaning. In this subsection, we classify word embeddings used by most EL works into two categories: (1) pre-trained; (2) learned.

TABLE 1: Summary of the step choices of DL based EL methods according to our proposed taxonomy. Due to the limited space, "PP" refers to the prior popularity feature, "SFS" refers to the surface form similarity feature, "TS" refers to the type similarity feature, "CS" refers to the context similarity feature, and "TC" refers to the topical coherence feature. "-" in the column of Algorithms means the corresponding model uses a relatively simple algorithm to select the mapping entity, such as a linear combination of features.

Model	**Embeddings**				**Features**					**Algorithms**
	Word	**Mention**	**Entity**	**Alignment**	**PP**	**SFS**	**TS**	**CS**	**TC**	
He et al. (ACL 2013) [32]			description					✓		-
Sun et al. (IJCAI 2015) [33]	learned	✓	surface form, type					✓		-
DSRM (arXiv 2015) [34]			description, context, type		✓	✓			✓	graph-based
Globerson et al. (ACL 2016) [35]					✓	✓			✓	-
Zwicklbauer et al. (SIGIR 2016) [36]	pre-trained		description, context		✓			✓	✓	graph-based
EDKate (CoNLL 2016) [37]	learned		context	✓	✓	✓		✓	✓	-
Yamada et al. (CoNLL 2016) [38]	learned		context	✓	✓	✓		✓	✓	-
Francis-Landau et al. (NAACL 2016) [39]	learned	✓	surface form, description		✓	✓		✓		-
Nguyen et al. (COLING 2016) [40]	learned	✓	surface form, description		✓	✓		✓	✓	-
Cao et al. (ACL 2017) [41]	learned	✓	context	✓	✓			✓	✓	-
Gupta et al. (EMNLP 2017) [42]	pre-trained		description, type		✓			✓		-
Deep-ED (EMNLP 2017) [43]	pre-trained		description, context		✓			✓	✓	MLP
NeuPL (CIKM 2017) [44]	learned		description, context	✓	✓	✓		✓	✓	-
Eshel et al. (CoNLL 2017) [45]	learned		context	✓	✓	✓		✓		MLP
MR-Deep-ED (ACL 2018) [46]	pre-trained		description, context		✓			✓	✓	MLP
Moon et al. (ACL 2018) [47]	pre-trained		context			✓		✓		-
Sil et al. (AAAI 2018) [48]	learned		description		✓			✓		MLP
DeepType (AAAI 2018) [49]					✓		✓			-
Mueller and Durrett (EMNLP 2018) [50]	learned		context	✓	✓	✓		✓		MLP
Kolitsas et al. (CoNLL 2018) [51]	pre-trained	✓	description, context		✓			✓	✓	MLP
SGTB-BiBSG (NAACL 2018) [52]	pre-trained		description, context		✓			✓	✓	-
NCEL (COLING 2018) [53]	learned	✓	context	✓	✓	✓		✓	✓	MLP
Le and Titov (ACL 2019) [54]	pre-trained		type					✓		MLP
Le and Titov (ACL 2019) [55]	pre-trained		description, context		✓			✓	✓	MLP
Logeswaran et al. (ACL 2019) [56]								✓		-
Sevgili et al. (ACL 2019) [57]			description, context					✓		MLP
RRWEL (IJCAI 2019) [58]	learned	✓	surface form, description		✓	✓		✓	✓	graph-based
RLEL (WWW 2019) [59]	pre-trained		description, context					✓	✓	RL
DCA (EMNLP 2019) [60]	pre-trained	✓	surface form, description, context		✓	✓	✓	✓	✓	MLP, RL
Gillick et al. (CoNLL 2019) [61]	pre-trained		description		✓			✓		MLP
E-ELMo (arXiv 2019) [62]	learned		context		✓	✓		✓		MLP
FGS2EE (ACL 2020) [63]	pre-trained		description, context, type		✓			✓	✓	MLP
ET4EL (AAAI 2020) [64]	learned	✓			✓		✓			-
Chen et al. (AAAI 2020) [65]	pre-trained		description, context, type		✓			✓	✓	MLP
REL (SIGIR 2020) [66]	learned		context	✓	✓			✓	✓	MLP
SeqGAT (WWW 2020) [67]			description		✓	✓		✓	✓	MLP
DGCN (WWW 2020) [31]		✓	description, context, type		✓			✓	✓	MLP
BLINK (EMNLP 2020) [68]			description					✓		-
ELQ (EMNLP 2020) [69]		✓	description					✓		-
GNED (KBS 2020) [70]	pre-trained		description, context		✓			✓	✓	MLP
JMEL (ECIR 2020) [71]	learned							✓		MLP
Yamada et al. (arXiv 2020) [72]		✓	context					✓	✓	-
M3 (AAAI 2021) [73]								✓	✓	-
Bi-MPR (AAAI 2021) [74]			description					✓		MLP
Chen et al. (AAAI 2021) [75]	learned	✓	surface form		✓	✓		✓	✓	MLP
CHOLAN (EACL 2021) [76]								✓		-
Zhang et al. (DASFAA 2021) [77]			description					✓		-

4.1.1 *Pre-trained*

Pre-trained language models can learn widely applicable embeddings of words based on their co-occurrences and neighborhoods in a large quantity of text corpora. They learn word embeddings in advance and store them in lookup tables. Lots of DL based EL methods directly use a pre-trained (fixed) lookup table to get word embeddings. Specifically, EL methods [36], [43], [51], [52], [55], [60], [63], [65], [70] utilized a Word2Vec [17] lookup table to get word embeddings. Additionally, several DL based EL methods [42], [46], [47], [54], [55], [59], [61] exploited a GloVe [19] lookup table as the word embedding source.

4.1.2 *Learned*

Pre-trained embedding may not be suitable for domain-specific data sets which contain string tokens with highly specialized semantics. In this case, many DL based EL methods learn a domain-specific word embedding using some embedding techniques. DL based EL methods [33], [37], [38], [39], [40], [41], [44], [45], [48], [50], [53], [58], [66], [78] learned word embeddings via Word2Vec based on the huge corpora such as Wikipedia. Word2Vec contains continuous bag-of-words (CBOW) model and skip-gram (SG) model [17]. In reality, EL methods [33], [38], [41], [44], [45], [50], [53], [66], [78] prefer to use SG model for training. Additionally, Shahbazi et al. [62] utilized ELMo [79] to learn word embeddings. ELMo is a large-scale context-sensitive language model that uses bi-directional long short-term memory (Bi-LSTM) [80] in forward and backward directions to encode word embeddings depending on its context. Adjali et al. [71] used Sent2Vec [81] which extends the CBOW model to learn representations of words.

Some word embedding aims to obtain numeric representations of words by leveraging their character-level information. The main idea is that words are made of morphemes, i.e., meaningful sequences of characters with different lengths. Injecting character-level information into the final word embedding could effectively handle the issue of out-of-vocabulary (OOV) words [82]. For example, OOV words often occur due to misspellings, and substrings of the word can be leveraged to approximate its embedding. Mueller and Durrett [50] and Onoe and Durrett [64] utilized CNNs [22] to generate character-level representations of words concatenated with the learned word embedding, which makes their models recognize character-level correspondences between entity mention and candidate entity better. Kolitsas et al. [51] and Chen et al. [75] utilized Bi-LSTM to capture significant character lexical information. Both of them generated the character-dependent embedding of a word by concatenating the forward hidden state of its last character, the backward hidden state of its first character, and its pre-trained word embedding.

Generally, word embedding cannot represent the order information of words. Intuitively, the position-dependent signal can reinforce semantics by incorporating the order information. Some EL systems add the positional embedding to the word embedding to explicitly encode the relative/absolute positions of words as vectors. Sun et al. [33] and Le and Titov [54] leveraged a positional embedding that was modeled by the distance between the word and the entity mention in a given piece of text. Xue et al. [58] followed Vaswani et al. [24] to define the positional embedding of each word in the entity mention. Specifically, they took the position index of each word as input and utilized a pre-defined sinusoidal function of the position index as its positional encoding.

4.2 Mention Embedding

Mention embedding is a learned representation for an entity mention. It is usually used to model the similarity between the entity mention and the candidate entity. For some EL methods [31], [33], it is simply obtained by averaging the embeddings of words which compose the entity mention.

Additionally, several DL based EL models utilize CNNs to generate vectors of entity mentions. Specifically, each word in the entity mention is encoded into a word embedding using a pre-trained Word2Vec lookup table, and thus a sequence of vectors $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n$ can be found, where n is the word length of the entity mention. Francis-Landau et al. [39] mapped the vector sequence into a fixed-size vector using a CNN parameterized with a filter bank $\mathcal{W} \in \mathbb{R}^{k \times dz}$, where k is the number of filter maps, d is the dimension of words, and z is the width of the convolution. Then they put the result through a *ReLU* function and combined the results with sum-pooling, based on the following formulation:

$$conv(w_{1:n}) = \sum_{i=1}^{n-z} \max\{0, \mathcal{W}\mathbf{w}_{i:i+z}\} \quad (1)$$

where $\mathbf{w}_{i:i+z}$ is a concatenation of word embeddings and the $\max$ is element-wise. Some other EL methods [40], [58], [60], [75] also utilized this similar process to obtain the mention embedding.

Moreover, due to the high ambiguity of entity mentions, Cao et al. [41], [53] proposed an embedding model, which can learn multiple sense embeddings for each entity mention to denote its different meanings. Specifically, each entity mention is first mapped to a set of shared mention senses according to a pre-defined dictionary. To train the mention sense embedding, they combined each mention sense and its context to predict the corresponding named entity by extending the CBOW model.

Kolitsas et al. [51] defined a "soft head" embedding to capture key components of the entity mention, which is built using an attention mechanism on top of the entity mention's word embeddings. They concatenated the embeddings of the first, last, and the "soft head" words of the entity mention, and then leveraged a shallow feedforward neural network (FFNN) to produce the representation $\mathbf{m}$ of the entity mention as follows:

$$\mathbf{m} = \text{FFNN}([\mathbf{m}_f; \mathbf{m}_l; \mathbf{m}_s]) \quad (2)$$

where $\mathbf{m}_f$, $\mathbf{m}_l$, $\mathbf{m}_s$ are the representations of the first, last, and "soft head" words respectively.

Wu et al. [31] took the entity mention with the text where it appears as input and leveraged ELMo [79] to learn a contextualized representation for the entity mention. What's more, Onoe and Durrett [64] fed the representation of the entity mention learned by ELMo into a Bi-LSTM encoder followed by a span attention layer to get the final mention embedding.

Li et al. [69] and Yamada et al. [72] leveraged the pre-trained language model BERT [83] to encode the entity mention. When training the BERT model, the masked language model and the next sentence prediction model are trained together, with the goal of minimizing the combined loss function of the two strategies. The masked language model randomly masks some tokens in text, and then independently recovers these masked tokens by conditioning on the encoding vectors obtained via a bi-directional Transformer. The next sentence prediction model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. Li et al. [69] took the entity mention with its context as a sequence of words as input to BERT and the outputs of tokens of the last layer were selected as the contextualized embeddings of words. They averaged the output embeddings of the entity mention words to generate the mention embedding. Yamada et al. [72] leveraged BERT to encode an input sequence consisting of words in the document and masked entity tokens corresponding to the entity mention. For each entity mention, taking the BERT output vector of its corresponding masked entity as input, the mention embedding is learned via a single-layer perceptron.

4.3 Entity Embedding

Entity embedding, which represents each entity in a KB in a continuous vector space, is important for DL based EL. It is helpful for capturing the entity-entity topical coherence feature (described in Section 5.5) by mapping similar entities close to each other in the same space. Entity embedding is also used as an important raw material to calculate the context similarity feature, which will be introduced in Section 5.4. KBs like Wikipedia provide lots of structured and textual data for learning entity embeddings, such as surface form, entity description, entity context, and type information. Some EL methods leverage one type of them, while others combine several types to learn the entity embedding.

4.3.1 Surface Form

The most direct way to get entity embeddings is using entity surface forms to learn lexical representations. Surface form consists of a word or a sequence of words like "Paris" or "New York City". It is usually used as the matching sequence to locate the corresponding candidate entity due to its uniqueness in the reference KB [84]. Sun et al. [33] obtained the entity surface form embedding by averaging the embeddings of words which compose the entity surface form based on learned word embeddings. Several EL methods [39], [40], [58], [60] used CNNs to convert each entity surface form into a vector, which is the same as the way of using CNNs to generate mention embeddings introduced in Section 4.2. What's more, Chen et al. [75] ran a Bi-LSTM on the entity surface form to get a representation for each entity.

4.3.2 Entity Description

Entity description is a piece of text that introduces useful background information about the entity. It usually provides a concise summary of salient properties for the entity. For instance, in the description page of the entity "Los Angeles Lakers" in Wikipedia, much useful information such as players, championships, and history can be found. A great quantity of EL works consider it as a good resource during the process of constructing entity embeddings, and take the representation of the entity description as the important auxiliary information to reinforce the semantic information of entities. It is also noted that zero-shot setting has been a surge in the field of EL [56], [68], [69], [74] where each named entity is defined only by its entity description, while other information such as type information and relations between entities are absent. Thus, it is essential to design a suitable method to learn the representation of the entity description to obtain the entity embedding.

He et al. [32] built a stacked denoising auto-encoders (DA) [85] to encode the entity description. Stacked DA is able to capture general concepts of the input and ignore noise. It takes a one-hot vector of the entity description as input and outputs a learned representation.

Huang et al. [34] first represented each entity description as a bag-of-words, which is then transformed by a word hashing layer into letter tri-gram vectors. They applied a deep neural network (DNN) on these tri-gram vectors to learn useful semantic entity embedding based on the entity description.

Sil et al. [48] computed a weighted average of embeddings of all words in the entity description by using the inverse document frequency of each word as the weight. They further applied a fully connected *tanh* activation layer to this obtained embedding to get the final entity description embedding.

Lots of studies [39], [40], [42], [58], [60] utilized CNNs to distill the entity description into a meaningful topic embedding. They combined this embedding with the entity surface form embedding mentioned in Section 4.3.1 to form the final entity embedding.

Doc2Vec [86] is a modification of Word2Vec and can learn fixed-size embeddings from variable-length pieces of text like entity descriptions. Both Zwicklbauer et al. [36] and Sevgili et al. [57] leveraged Doc2Vec to generate an entity embedding based on the entity description.

Phan et al. [44] and Fang et al. [59] both used a single-directional LSTM network to encode the entity description. Instead of taking the last hidden state as the representation for entity description, Phan et al. [44] applied max-pooling over all hidden state vectors to produce a fixed-size entity embedding.

Ganea and Hofmann [43] collected co-occurrence words from the entity description of each entity and regarded them as positive words of that entity for learning its embedding. They leveraged the word-entity co-occurrence counts to define a practical approximation of a word-entity conditional distribution. In addition, they sampled negative words unrelated to that entity from a generic word distribution. Finally, they used a max-margin loss to infer the optimal embedding of the entity with a goal that vectors of positive words are closer to the embedding of that entity compared with vectors of negative words. Many EL methods [31], [46], [51], [52], [55], [59], [60] follow this work and use the same method to generate the entity embedding.

Hu et al. [70] extracted words with the highest TF-IDF from the entity description as related words of the entity. To learn better representations of entities by aggregating the semantic information from their neighboring entities and words, they constructed a heterogeneous entity-word graph for each document, which consists of entity nodes corresponding to the candidate entities of mentions in the document, word nodes corresponding to the related words of candidate entities, and relationships added by computing the similarity score between nodes based on pre-trained word embeddings. They applied Graph Convolutional Network (GCN) [87] on this graph to encode the global semantic information among candidate entities and finally generated entity embeddings.

Sometimes, entity descriptions are a little too long to accurately express semantic information of entities. To alleviate the long-term dependency problem and extract worthy information, some DL based EL systems leverage BERT [83] to encode the entity description in order to learn the entity embedding. Specifically, several EL works [67], [68], [69], [77] took the entity description as a sequence of words as input to BERT. Most of these works [68], [69], [77] inserted a special start token [CLS] at the beginning of the input sequence and the output of the last layer at this start token produced by the Transformer encoder is regarded as the vector representation of the input sequence. Fang et al. [67] obtained the entity embedding via average-pooling over the hidden states of all description tokens in the last BERT layer.

However, BERT can only take a limited length (i.e., 512 tokens) of the entity description as input. To tackle this problem, Tang et al. [74] proposed a multi-paragraph reading model for discovering more textual information in the entity description which is composed of multiple paragraphs. Specifically, they first utilized BERT to encode the concatenation of the mention context and the entity description paragraph to obtain a representation for each paragraph. These representations are then used as the input of a multi-head attention module proposed in Transformer [24] to gather the semantic dependence among the paragraphs. Finally, a weighted-pooling layer is applied on hidden states output by the multi-head attention module to produce the entity embedding.

4.3.3 Entity Context

In addition to entity descriptions, KBs like Wikipedia can provide additional valuable context information for named entities. Some DL based EL methods utilize various kinds of entity context to learn entity embeddings, such as anchor texts, fact information, and neighbor entities.

Anchor text in Wikipedia is a hyperlink from an entity mention in an entity page linking to its corresponding entity page, and can help the learning of entity embeddings in two aspects. First, the anchor text is a link that jumps from one entity page to another, providing rich entity-entity co-occurrence information. What's more, anchor texts residing in documents supply abundant context words for their referring entities, and thus they can offer rich word-entity co-occurrence information.

Specifically, Yamada et al. [38] used anchor texts in Wikipedia to learn the entity embedding based on the entity-entity co-occurrence information they provide. Inspired by Wikipedia Link-based Measure (WLM) [88], which is a standard entity relatedness measure in traditional EL, they assumed that entities with similar hyperlinks are related. For example, the teams "Boston Celtics" and "Toronto Raptors" are highly related because they have many common hyperlinks to the entity pages about "NBA". Their model simply learned to place entities with similar hyperlinks near one another in the vector space via maximizing the conditional probability $P(e_o|e)$, where e_o is one of the hyperlinks of the entity e, which is similar to the idea of the SG model in Word2Vec [17]. Fang et al. [37] exploited the entity-entity co-occurrence information offered by anchor texts to learn the entity embedding by placing the representations of two entities connected by anchor texts close to each other in the vector space. Sevgili et al. [57]

used anchor texts in Wikipedia to create a graph whose nodes are entities and edges are the hyperlinks between entities. Then the vector representation of each entity is generated by running DeepWalk [89] over the constructed graph.

Starting from Ganea and Hofmann [43], many EL methods [31], [46], [51], [52], [55], [59], [60], [70] took word-entity co-occurrence counts provided by anchor texts as another source to learn entity embeddings. We have introduced their specific learning method in Section 4.3.2. Additionally, lots of EL approaches [31], [37], [38], [41], [44], [45], [50], [66], [78] utilized anchor texts to construct entity embeddings by mapping embeddings of words and entities into the same vector space. In these approaches, similar words and entities are placed close to each other in a vector space. More details will be introduced in Section 4.4.

Recently, Yamada et al. [72] utilized anchor texts residing in Wikipedia to learn the contextualized entity embedding. Specifically, they proposed a new masked entity prediction model to learn the entity embedding, inspired by the masked language model adopted in BERT [83]. Masked entity prediction aims to predict randomly masked entities based on words and non-masked entities. They trained this model based on the contexts of the referring entities provided by the anchor texts in Wikipedia.

A KB is usually composed of facts about entities. Facts are also important information for modeling entity embeddings. We denote a fact as $\langle h, r, o \rangle$, where h is a head entity, r is a relation, and o is a tail entity. Fang et al. [37] followed TransE [90] to define a score function $s(\cdot)$ for a fact $\langle h, r, o \rangle$ based on embeddings of h, r, and o as follows:

$$s(h, r, o) = b - \frac{1}{2} \| \mathbf{h} + \mathbf{r} - \mathbf{o} \|^2 \tag{3}$$

where $\mathbf{h}$, $\mathbf{r}$, $\mathbf{o}$ are the embeddings of h, r, o respectively, and b is a constant for numerical stability. After maximizing the above score function, the representations of h and o can be used as entity embeddings. Moon et al. [47] also leveraged facts to generate entity embeddings as follows:

$$s(h, r, o) = score(\mathbf{h}, \mathbf{r}, \mathbf{o}) \tag{4}$$

where $score()$ is a deep neural network that produces a likelihood of a valid fact $\langle h, r, o \rangle$. For a certain entity e, Huang et al. [34] defined a connected entity set $\mathcal{E}(e)$ containing its corresponding head entities and tail entities existing in the same facts, which is another form of entity-entity co-occurrence information. For each entity in $\mathcal{E}(e)$, they obtained its surface form and represented it as a bag-of-words. The vectors of all connected entities are concatenated to generate the final entity embedding. Cao et al. [41], [53] extended the SG model in Word2Vec by maximizing the log probability of observing its connected entity set given an entity as follows:

$$\mathcal{L} = \sum_{e} \log P(\mathcal{E}(e) | e) \tag{5}$$

Thus, entities sharing many common connected entities tend to have similar representations.

Zwicklbauer et al. [36] defined the neighbor entities of entity e as entities that appear surrounding e in text. They generated a corpus that exclusively comprises entities sequentially by replacing all available linked surface forms in documents with its corresponding target entities and removing all non-entity identifiers like words and punctuations. Similar to Equation 5, they leveraged the SG model in Word2Vec over this created corpus to learn entity embeddings by leveraging the neighbor entities in a window.

4.3.4 Type Information

A KB consists of a type hierarchy and entities that are instances of types. The entity type is a kind of important information to represent the semantics of entities. A large number of EL methods have leveraged entity type information to learn entity embeddings.

Sun et al. [33] averaged embeddings of entity type words based on the learned word embeddings to get the entity type embedding. They utilized a low-rank neural tensor network [91] to jointly encode the entity type embedding and the entity surface form embedding introduced in Section 4.3.1 to learn the final entity representation. Gupta et al. [42] calculated a relevant probability $P(t|e)$ of observing type t given entity e as $\mathrm{sigmoid}(\mathbf{t} \cdot \mathbf{e})$, where $\mathbf{t}$ and $\mathbf{e}$ are type embedding and entity embedding respectively. They maximized this probability to learn entity and type embeddings jointly and injected type embeddings into entity embeddings ultimately.

Huang et al. [34] extracted a set of attached entity types for each entity based on Freebase [3], represented each entity type as a one-hot vector, and used it as one of the inputs for DNN to learn entity embeddings. Wu et al. [31] extracted the notable type from Freebase for each entity and utilized the average embedding of words in the type string as a type embedding. They then obtained the final entity embedding by concatenating the type embedding with the learned entity embedding [43]. Le and Titov [54] only used type information to produce the entity embedding. They first leveraged Freebase to get a type set for each entity. Each type t of the entity e is assigned a vector $\mathbf{t}$ based on the pre-trained word embeddings, and then the representation $\mathbf{e}$ of the entity can be calculated as follows:

$$\mathbf{e} = ReLU(\mathcal{W} \frac{1}{|T_e|} \sum_{t \in T_e} \mathbf{t} + b) \tag{6}$$

where T_e is the type set of the entity e, $\mathcal{W}$ is a weight matrix and b is a bias vector.

Chen et al. [65] proposed to inject latent type information into the entity embedding via modeling the immediate context where the entity appears. They considered that context consistency is a strong proxy for type compatibility, so they leveraged the pre-trained BERT to encode entity contexts that are randomly sampled from Wikipedia for that entity. The representation of the entity is computed by aggregating all the context representations via average-pooling.

Hou et al. [63] exploited the embeddings of semantic types to generate entity embeddings. They first created a dictionary of fine-grained semantic types, and then extracted semantic types from its Wikipedia article for each entity. To construct semantic reinforced entity embedding, they averaged the semantic type embeddings obtained from Word2Vec and combined them with learned entity embeddings [43] via linear aggregation.

4.4 Alignment Embedding

As introduced in Sections 4.1 and 4.3, the word embedding and the entity embedding could be learned in different ways based on various data. However, in many cases, these two categories of embeddings are learned separately and are not in the same vector space. This makes it impossible to calculate the similarity between words and entities effectively, which is an essential operation for computing the context similarity feature in EL.

Therefore, aligning word and entity embeddings into the same vector space (called alignment embedding) is very necessary and important so that similar words and entities are placed close to each other in a common space. After alignment, we can measure the similarity between words and entities by simply computing the cosine similarity of their embeddings. We introduce some classic alignment embedding techniques as follows.

As mentioned in Section 4.3.3, anchor text is a key resource for aligning embeddings of words and entities. The window words surrounding the anchor text could be regarded as context words of its referring entity, and could provide ample word-entity co-occurrence examples, which could be leveraged to align embeddings. Yamada et al. [38] proposed an alignment model which leverages anchor texts and their context words by extending the SG model in Word2Vec. The objective function is defined to predict the surrounding context words of the referring entity of the anchor text:

$$\mathcal{L} = \sum_{(e,Q)\in A} \sum_{w\in Q} \log P(w|e) \tag{7}$$

where A denotes the set of anchor texts, e denotes the referring entity of the anchor text, Q is the set of its surrounding context words, and w is a word in Q.

Moreover, based on the word-entity co-occurrence counts collected from anchor texts and entity descriptions, Fang et al. [37] maximized the score function $s(\cdot)$ to shorten the distance between each co-occurrence pair:

$$s(w,e) = b - \frac{1}{2}\| \mathbf{w} - \mathbf{e} \|^2 \tag{8}$$

where $\mathbf{w}$ and $\mathbf{e}$ represent the corresponding embeddings for word w and entity e respectively, and b is a constant.

Additionally, Eshel et al. [45] used Word2Vecf embedding algorithm [92] to train word and entity embeddings jointly via leveraging the word-entity co-occurrence counts collected from the set of frequent words appearing in the Wikipedia article of that entity.

5 Feature

Features are designed to calculate the similarity between the entity mention and the candidate entity in various aspects. Benefiting from DL, more and more EL methods have utilized embeddings as features directly or used different embeddings as a source for features design. In addition, some traditional features which are manually designed, such as the prior popularity feature, are still applied in recent DL based EL methods. In this section, we review five categories of features found to be effective and broadly used to rank candidate entities in DL based EL methods, i.e., prior popularity, surface form similarity, type similarity, context similarity, and topical coherence.

5.1 Prior Popularity

The prior popularity is the probability of the appearance of a candidate entity given an entity mention without considering the context where the mention appears. It is a simple but strong feature, and almost all DL based EL methods leverage this feature. For example, the former U.S. president "Barack Obama" is more popular than his wife "Michelle Obama", both of which could be referred to by "Obama". In most cases when people mention "Obama", they mean the former president rather than his wife. Anchor text in Wikipedia is the most broadly used source to estimate the prior popularity feature. Given an entity mention m, the prior popularity feature $p(e|m)$ of a candidate entity e can be estimated as follows:

$$p(e|m) = \frac{count(m,e)}{count(m)} \tag{9}$$

where $count(m)$ denotes the number of anchor texts having the entity mention m as the surface form in Wikipedia; $count(m,e)$ represents the number of anchor texts with the surface form m pointing to the candidate entity e. In many common real-world entity linking data sets, the accuracy of using this prior popularity feature alone can reach $70\% - 85\%$ [43], which demonstrates its effectiveness in EL.

5.2 Surface Form Similarity

Intuitively, an entity mention and a candidate entity with the same or similar surface forms are likely to indicate a gold mapping. For example, the entity mention "Univ Manchester" is more likely to refer to the university "The University of Manchester" rather than the football team "Manchester City F.C." due to their greater surface form similarity. A great many DL based EL systems leverage some traditional surface form similarity measures, such as edit distance, Dice coefficient score, character Dice, skip bigram Dice, and left and right Hamming distance scores. We have introduced them in our previous survey [5] and would not introduce them in detail here.

In addition, several DL based EL methods [39], [40], [58], [60] considered the cosine similarity between the mention embedding introduced in Section 4.2 and the entity surface form embedding introduced in Section 4.3.1 as the surface form similarity feature. Chen et al. [75] leveraged CNNs over the mention embedding and the entity surface form embedding to extract n-gram features of the entity mention and the candidate entity respectively. A two-layer fully connected neural network is then applied to the concatenation of the output embeddings to obtain the surface form similarity feature. What's more, Moon et al. [47] trained a separate deep neural network to encode surface forms of the entity mention and the candidate entity, which produces a purely lexical embedding without semantic allusion. The surface form similarity feature between the entity mention and the candidate entity is computed based on their distance in the embedding space.

5.3 Type Similarity

In addition to being used as auxiliary information to construct entity embeddings introduced in Section 4.3.4, type information is utilized by several DL based EL methods to calculate the type similarity feature. For example, knowing that the correct type of the entity mention "Boston" in some context is *sports_team* could constrain its corresponding entity with relevant types, such as sport or team. Generally, KBs contain rich type information for a candidate entity, while the type information of an entity mention in some context is absent. Some typing systems are proposed to predict types of an entity mention with the help of its surrounding context.

Raiman and Raiman [49] proposed DeepType, which is a method to solve EL only using type constraints. They restricted the types in their typing system via selecting a set of parent-child relations over the ontology in Wikipedia. Then they leveraged a Bi-LSTM classifier to obtain the type conditional probability

$P(t|m,c)$ for type t given the entity mention m and its surrounding context c. For a candidate entity e belonging to types $t_1, ..., t_n$, the type similarity feature $f_{typ}(m,e)$ is calculated as follows:

$$f_{typ}(m,e) = 1 - \beta + \beta \cdot \prod_{i=1}^{n} (1 - \alpha_i + \alpha_i \cdot P(t_i|m,c)) \quad (10)$$

where β is a smoothing parameter over all types and α_i is a smoothing parameter per type.

Onoe and Durrett [64] proposed a fine-grained typing system, which contains tens of thousands of types derived from Wikipedia categories. They selected a single linear layer to decode the concatenation of the mention embedding introduced in Section 4.2 and the context embedding which will be introduced in Section 5.4.1. They then utilized a *sigmoid* function to normalize the output vector of the decoder and obtained the type conditional probability $P(t|m,c)$ for type t given the entity mention m and its surrounding context c. The type similarity feature $f_{typ}(m,e)$ between the entity mention m and the candidate entity e of types $t_1, ..., t_n$ is calculated as follows:

$$f_{typ}(m,e) = \sum_{i=1}^{n} P(t_i|m,c) \quad (11)$$

Yang et al. [60] trained a typing system proposed by Xu and Barbosa [93]. They concatenated the mention embedding and the context embedding to form a representation, which is then used as the input of a softmax classifier to predict the probability distribution over four types (i.e., PER, GPE, ORG, and UNK) of the entity mention. The type similarity feature is measured as the similarity between the predicted type distribution of the entity mention and the types of the candidate entity.

5.4 Context Similarity

The most straightforward feature for entity linking is to measure the similarity between the representation derived from the context around the entity mention and the representation associated with the candidate entity. To model this context similarity feature, the corresponding context of the entity mention would be encoded as a context embedding using various neural architectures, and then diverse computing methods would be utilized to calculate the similarity score between the generated context embedding and the entity embedding mentioned in Section 4.3. In the following, we first describe how to generate the context embedding, and subsequently show the computing methods for calculating this context similarity feature.

5.4.1 Context Embedding

With the advent of latent embeddings, traditional bag-of-words [94], [95], [96] and keyphrase [15], [97] models may seem to be superseded by neural models, which can implicitly capture semantic and syntactic information of the context. Formally, given an entity mention m in a sentence $(w_j^{\text{left}}, ..., w_1^{\text{left}}, m, w_1^{\text{right}}, ..., w_j^{\text{right}})$ within a pre-defined window size j, the context of m contains the left-side j context words of m as $(w_j^{\text{left}}, ..., w_1^{\text{left}})$ and the right-side j context words of m as $(w_1^{\text{right}}, ..., w_j^{\text{right}})$. Context embedding aims to represent the surrounding context of m as a dense and low-dimensional vector. As the context is composed of words, a large number of DL based EL methods leverage word embeddings introduced in Section 4.1 as input to learn the context embedding based on various neural models. For the simplest case, some EL methods [38], [41], [52], [62], [71] averaged the embeddings of context words to derive the context embedding. For the other cases, we group models of generating context embeddings into several categories based on their different neural architectures, such as RNNs, CNNs, attention, Transformers, and others. In the remainder, we will describe these categories of context embedding models in detail.

RNNs-based. RNN is designed to process data that is sequential in nature, especially when the input sequence has variable-length. Accordingly, the RNNs-based neural models are appropriate for context learning. RNN intends to capture word dependencies and context structures using the recurrent unit, which is an NN that is shared between all time steps [26]. At time step l, the recurrent unit takes the l-th input $\mathbf{x}_l$ and the hidden state vector of the previous time step $\mathbf{h}_{l-1}$ to produce the hidden state vector of the current time step $\mathbf{h}_l$. Hence, $\mathbf{h}_l$ contains the previous and current input information $\mathbf{x}_1, ..., \mathbf{x}_l$.

LSTM [20] is the most popular RNN architecture, which is designed to better maintain the long term memory. It introduces a memory cell to remember values over arbitrary time intervals, and uses three kinds of gates (input gate, output gate, and forget gate) to regulate the flow of information into and out of the cell. Fang et al. [59] utilized a single-directional LSTM to encode the context as an embedding, while some works [31], [54], [64], [75] used a Bi-LSTM [80] to encode context. Additionally, a few DL based EL methods [42], [44], [48] used a forward LSTM and a backward LSTM to encode the left-side and right-side context of the entity mention respectively. Specifically, Phan et al. [44] leveraged the hidden state vectors to generate the context embedding, while the others [42], [48] used the output vectors. It is noted that these two LSTMs are different from Bi-LSTM since each LSTM of them only embeds half of the context while each LSTM of Bi-LSTM encodes the full context.

A slight variation of LSTM is the Gated Recurrent Unit (GRU) [21]. Compared with LSTM, it is simpler by combining the input and forget gate into a single update gate, and has been adopted by some DL based EL methods such as [45], [50]. Both of them utilized a forward GRU and a backward GRU to encode the left-side and right-side context respectively. The outputs of the forward GRU and the backward GRU are leveraged to generate the context embedding.

CNNs-based. RNN is trained to recognize patterns across time, while CNN learns to recognize patterns across space [22]. RNN works well when the long-term semantics is required, while CNN works well when detecting local and position-invariant patterns is important, which might be a keyphrase that expresses a particular sentiment [98]. Thus, contexts in CNNs-based models are not supposed to be too long. CNN consists of multiple convolutional layers, each of which extracts local features around each context word, and the size of the output of the convolutional layers depends on the number of words in context. The context embedding is constructed via combining local feature vectors extracted by convolutional layers.

Several DL based EL methods [33], [39], [40], [58] applied CNNs over the context words to generate the hidden vector sequences, which were then transformed by a non-linear function and pooled by sum-pooling [39], [40], [58] or average-pooling [33] to generate the context embedding.

Attention-based. Starting from Bahdanau et al. [23], attention mechanism has become an increasingly popular concept and useful tool in many NLP tasks. The idea of attention is to highlight the

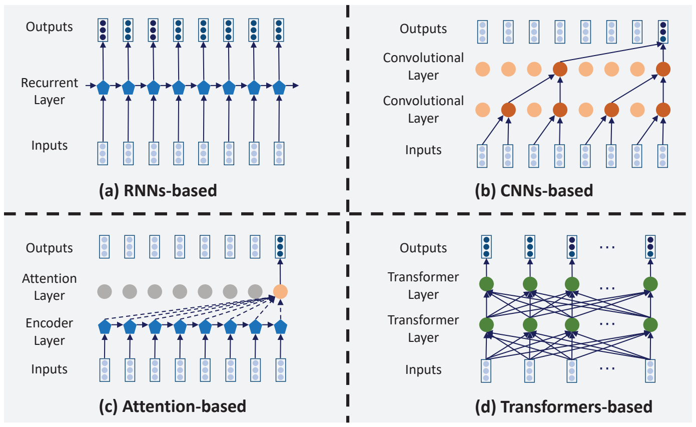


Fig. 3: Architectures of different neural networks for learning the context embedding.

informative parts of the input based on an attention vector. In order to give different attention weights to the context words in learning the context embedding, attention-based EL models estimate how strongly they are correlated with the attention vector. Then these models take the weighted sum of the values of the context words as the context embedding. To illustrate more clearly, we propose a general formula to introduce various attention-based models for learning the context embedding, which is generally defined as a sum of the context word values weighted by the correlation between the value and the attention vector:

$$\mathbf{c} = F(U(\mathcal{V}, \mathcal{A}))\mathcal{V} \tag{12}$$

where $\mathcal{V}$ and $\mathcal{A}$ correspond to the matrices of values and attention vectors respectively, $U(\cdot)$ denotes a function used to calculate the attention weight of each value, $F(\cdot)$ usually denotes a *softmax* function, and $\mathbf{c}$ is the context embedding. We summary some prominent attention-based EL models by introducing different choices of values, attention vectors, and $U(\cdot)$ as follows.

In learning the context embedding, values are usually components of the context. Lots of attention-based EL models [43], [46], [51], [53], [55], [60], [63], [70] used embeddings of context words as values. Phan et al. [44] took hidden state vectors of the forward LSTM and the backward LSTM as values, while Wu et al. [31] and Chen et al. [75] took outputs of the Bi-LSTM as values. Eshel et al. [45] and Mueller and Durrett [50] regarded outputs of the forward GRU and the backward GRU as values.

Attention vectors are used to provide background information and discover the important and relevant parts of values. Some EL works [31], [44], [45], [50], [53] chose the same attention vector for all values, which is the embedding of the candidate entity. Several DL based EL models [43], [46], [51], [55], [60], [63], [70] assumed that a context word is important if it is strongly related to at least one candidate entity, so they selected the embedding of the most related candidate entity for each context word as the attention vector. Additionally, Zhang et al. [77] regarded the representations of connected entities of the candidate entity introduced in Section 4.3.3 as the attention vectors.

$U(\cdot)$ is a function used to determine the attention weight of each value by measuring the similarity or correlation between the value and the attention vector. Cao et al. [53] used the cosine similarity, Mueller and Durrett [50] and Zhang et al. [77] leveraged the dot product, and a great many attention-based models [31], [43], [46], [51], [55], [60], [63], [70] utilized the bilinear similarity. We will introduce these three metrics in detail in Section 5.4.2 as they are also utilized to calculate the context similarity feature. Additionally, Phan et al. [44] and Eshel et al. [45] obtained the similarity by leveraging a single-layer perceptron to encode the value and the attention vector.

Transformers-based. RNNs-based models suffer from the sequential processing of the context, which is one of their computation bottlenecks. Despite that CNNs-based models are less sequential, the computational cost to capture relations between words in the context also grows with the increasing length of the context. Transformers [24] overcome this limitation via applying self-attention mechanism to calculate an attention weight for each word in the context, in order to model the influence each word has on another [98]. Recently, Transformers-based pre-trained language models use much deeper neural architectures compared with models based on RNNs and CNNs, and are pre-trained on much larger text corpora to learn contextualized text representations by predicting words conditioned on their surrounding contexts. These pre-trained language models could be fine-tuned using task-specific labels of downstream tasks like EL.

BERT [83] is a widely used Transformers-based pre-trained language model in context embedding learning for EL task. Several DL based EL methods [67], [68], [77] took the word-pieces of the entity mention and its context as the input of BERT. Wu et al. [68] and Zhang et al. [77] inserted a start token [`CLS`] and regarded the output of the last layer at this start token as the context embedding, while Fang et al. [67] obtained the context embedding via average-pooling over the hidden states of all context tokens in the last layer. Tang et al. [74] leveraged the multi-paragraph reading model based on BERT introduced in Section 4.3.2 to encode the context as a representation. Compared with the model introduced in Section 4.3.2, they added a new multi-head attention module which leverages the learned entity embedding as the query to emphasize the importance of the context which is encoded in the entity embedding.

Architectures of RNNs-based, CNNs-based, attention-based, and Transformers-based models for learning the context embedding are shown in Figure 3.

Other Methods. There are also some other methods to learn context embeddings, which we briefly introduce as follows. He et al. [32] used stacked DA [85] to encode the context as a representation, while Zwicklbauer et al. [36] and Sevgili et al. [57] utilized Doc2Vec [86] to generate an embedding for the context, which is the same way for them to learn the entity embedding based on the entity description introduced in Section 4.3.2.

5.4.2 *Context Similarity Feature Computation*

After the generation of the context embedding described above, we could compute the context similarity feature based on some similarity measures between the context embedding of the entity mention and the entity embedding of the candidate entity. Here, we overview the context similarity measures used by DL based EL models in the following.

Cosine similarity is a commonly used similarity measure for real-valued vectors. Based on it, the context similarity feature $f_{cnt}(m, e)$ between the entity mention m and the candidate entity e can be defined as follows:

$$f_{cnt}(m, e) = \frac{\mathbf{c} \cdot \mathbf{e}}{\parallel \mathbf{c} \parallel \parallel \mathbf{e} \parallel} \tag{13}$$

where $\mathbf{c}$ is the context embedding for the entity mention m, $\mathbf{e}$ is the entity embedding of the candidate entity e, and $\parallel \mathbf{c} \parallel$ and

$\parallel \mathbf{e} \parallel$ are lengths of two embeddings respectively. Cosine is a trigonometric function that helps to describe the orientation of two vectors. The highest similarity value 1 is reserved for the two vectors that are the most close together, while the lowest similarity value 0 is reserved for the two vectors that are the least close together. Lots of DL based EL methods [31], [33], [36], [38], [39], [40], [41], [48], [52], [53], [58], [61], [65], [66], [75], [78] applied this metric to calculate the context similarity feature.

Additionally, the numerator of Equation 13, which is the dot product of the context embedding $\mathbf{c}$ and the entity embedding $\mathbf{e}$, has been directly used to measure the context similarity feature by several EL methods [32], [47], [51], [68], [69], [72]. Compared with cosine similarity that cares about angle difference between two vectors, dot product focuses on both angle and magnitude.

Starting from Ganea and Hofmann [43], bilinear similarity has been utilized by several DL based EL systems [31], [46], [55], [60], [63], [66], [70] to compute the context similarity feature, which can be denoted as follows:

$$f_{cnt}(m, e) = \mathbf{c}\mathcal{W}\mathbf{e}^{\mathrm{T}} \tag{14}$$

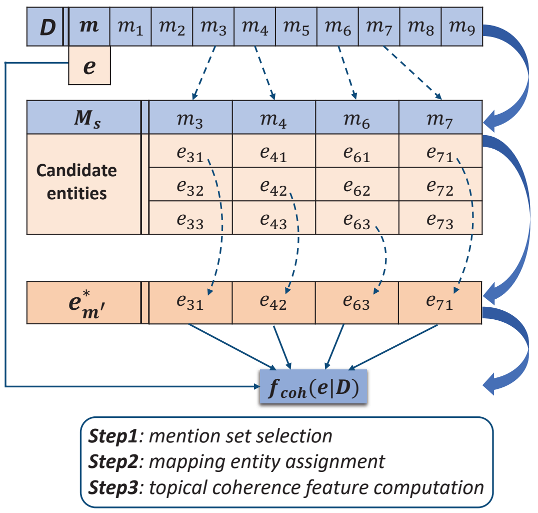
where $\mathbf{c}$ is the context embedding for the entity mention m, $\mathbf{e}$ is the entity embedding of the candidate entity e, and $\mathcal{W}$ is a trainable diagonal matrix, which indicates the relationships between two embeddings.

Moreover, several EL works utilized neural architectures to calculate the context similarity feature. Specifically, some DL based EL methods [50], [54], [67] passed the concatenation of the context embedding and the candidate entity embedding to a single-layer perceptron to learn a context similarity score for each candidate entity, while Sevgili et al. [57] and Fang et al. [59] used an MLP instead. Some DL based EL methods [56], [73], [76] concatenated the mention context and the entity description as a sequence pair together with a special start token [CLS] and separator tokens [SEP] as the input of BERT, while Wu et al. [68] concatenated the context embedding described in Section 5.4.1 and the entity embedding described in Section 4.3.2 as the input of BERT. All these works regarded the output of the last hidden layer at the start token as the representation of the input pair. Some of these works [56], [68], [76] applied a linear layer to this representation to produce the context similarity feature, while Gu et al. [73] utilized an MLP with softmax function over this representation.

5.5 Topical Coherence

The aforementioned features mainly concern the local similarity between the entity mention and one of its candidate entities, and each entity mention in the document could be linked independently based on these features, which are often called local features. Starting from Cucerzan et al. [10], more and more EL methods take into consideration the global topical coherence among the referring entities within the same document, where entity mentions are linked collectively. It is based on an assumption that co-occurring entity mentions in the same document often refer to topically coherent entities. Given a document D, the topical coherence feature $f_{coh}(e|D)$ for a candidate entity e of an entity mention m could be calculated via averaging the topical coherence between the candidate entity e and assigned mapping entities of an entity mention set in the same document:

$$f_{coh}(e|D) = \frac{1}{|M_s|} \sum_{m' \in M_s} \psi(e, e^*_{m'}) \tag{15}$$



D
m
m_1
m_2
m_3
m_4
m_5
m_6
m_7
m_8
m_9
e
Step1
M_s
m_3
m_4
m_6
m_7
Candidate entities
e_{31} e_{41} e_{61} e_{71}
e_{32} e_{42} e_{62} e_{72}
e_{33} e_{43} e_{63} e_{73}
Step2
$e^*_{m'}$
e_{31} e_{42} e_{63} e_{71}
Step3
$f_{coh}(e|D)$
Step1: mention set selection
Step2: mapping entity assignment
Step3: topical coherence feature computation

Fig. 4: The process of calculating the topical coherence feature $f_{coh}(e|D)$ for a candidate entity e of an entity mention m in a document D.

where M_s is a mention set in the document D, $e^*_{m'}$ is the assigned mapping entity of the entity mention m' in the mention set M_s, and $\psi(\cdot)$ is a function to calculate a pairwise similarity between entities which may indicate the degree of their topical coherence. In order to introduce the various methods of calculating the topical coherence feature clearly, we summarize the process of this feature generation in the following three steps: (1) mention set selection; (2) mapping entity assignment; and (3) topical coherence feature computation. Firstly, a mention set M_s whose mapping entities are the targets that the candidate entity e needs to be coherent with is selected from all the entity mentions in the document D. Next, for each entity mention m' in the mention set M_s, we *temporarily* assign a mapping entity $e^*_{m'}$ to it from its candidate entities when computing this feature, as its real mapping entity is unknown to us and needs to be figured out in this entity linking task. Finally, the topical coherence feature is obtained by computing the average similarity between the candidate entity e and assigned mapping entities $e^*_{m'}$. An illustration of this process is shown in Figure 4.

The key component in the third step of feature computation is the similarity function $\psi(\cdot)$ for a pair of entities. Entity embeddings introduced in Section 4.3 are usually utilized in this step to calculate the entity similarity. Specifically, a large number of DL based EL methods [34], [36], [36], [40], [41], [44], [51], [52], [53], [58], [78] leveraged cosine similarity as the similarity function and several EL methods [31], [38], [43], [46], [55], [60], [63], [65], [66], [70] exploited bilinear similarity to calculate the entity similarity based on entity embeddings. Another well-known method of entity similarity is WLM [88], which is a powerful but simple method based on anchor texts in Wikipedia. More details of WLM can be found in our previous survey [5]. Some DL based EL methods [52], [58], [78] used WLM as the similarity function $\psi(\cdot)$. Moreover, starting from Le and Titov [46], some EL works [55], [63], [66] incorporated relations between entities as latent variables to compute pairwise entity similarity. Fang et al. [37] utilized a distance based metric which is similar to Equation 8. Some hand-crafted similarity measurements such as the number of KB relations existing between entities [35] and entity-entity co-occurrence counts obtained from Wikipedia [52] were also leveraged to calculate the entity similarity.

However, even if we get the pairwise similarity function between entities, it is still hard to compute the topical coherence feature for the candidate entity. According to some works [38], [43], [55], [60], the optimization of this feature is an NP-hard problem since the second step (i.e., mapping entity assignment) that needs to assign a mapping entity for each entity mention in M_s, is impossible to be implemented before the entity linking task is finished as its real mapping entity is unknown to us. To solve this problem, several DL based EL methods obtain the assigned mapping entity for an entity mention temporarily by selecting an entity from its candidate entities, which are called simplified methods. In addition, some DL based EL methods do not obtain the assigned mapping entities explicitly but leverage the whole set of candidate entities to compute the topical coherence feature, which are called optimized methods. In the remainder of this subsection, we introduce simplified methods and optimized methods respectively.

5.5.1 Simplified Methods

Simplified methods try to transform the NP-hard problem into a computable problem by determining a *temporary* mapping entity for each entity mention in the mention set in the calculation of the topical coherence feature. Here, we mainly focus on how to determine a temporary mapping entity for an entity mention. According to the range of entity mentions that are selected to form the mention set in the first step (i.e., mention set selection), we classify the simplified methods into the following two groups: (1) whole coherence; (2) partial coherence.

Whole coherence. The whole coherence means all the other entity mentions in the document D are selected to form the mention set M_s except the entity mention m of the candidate entity e for which we calculate the topical coherence feature via Equation 15.

Some EL method [99] chose the candidate entity with the highest prior popularity feature as the assigned mapping entity $e^*_{m'}$ for the entity mention m' in M_s, which is defined as follows:

$$e^*_{m'} = \underset{e' \in C(m')}{\arg\max}\, P(e'|m') \tag{16}$$

where $C(m')$ is a set of candidate entities for entity mention m' and $P(e'|m')$ is the prior popularity feature introduced in Section 5.1. Additionally, several DL based EL methods [31], [35], [37], [55] regarded the most similar candidate entity to the candidate entity e in Equation 15 as the assigned mapping entity $e^*_{m'}$ for the entity mention m' in M_s, which is defined as follows:

$$e^*_{m'} = \underset{e' \in C(m')}{\arg\max}\, \psi(e, e') \tag{17}$$

where $\psi(\cdot)$ is an entity similarity function as we have introduced earlier.

Partial coherence. Different from the whole coherence, the partial coherence means that the candidate entity e in Equation 15 only needs to be coherent with the assigned mapping entities of parts of entity mentions in the same document, which significantly reduces computational complexity and time consumption.

Yamada et al. [38] regarded a set of unambiguous entity mentions in the document D as the mention set M_s and considered an entity mention unambiguous when the prior popularity feature of one of its candidate entities is greater than 0.95. They used Equation 16 to obtain the assigned mapping entity $e^*_{m'}$ for the entity mention m' in M_s. The topical coherence feature for the candidate entity is computed as the similarity between the candidate entity embedding and the averaged embedding of assigned mapping entities of unambiguous entity mentions.

Kolitsas et al. [51] leveraged the candidate entities having high local scores derived from local features as assigned mapping entities $e^*_{m'}$ to participate in the calculation of the topical coherence feature. The entity mentions of these selected candidate entities form the mention set M_s. To obtain the topical coherence feature for each candidate entity, they summed up the representations of assigned mapping entities and calculated the similarity between the candidate entity embedding and this generated representation.

It is noted that in the above two simplified methods, candidate entities of different entity mentions have a fixed mention set throughout the EL process, while in the following simplified methods, the mention set changes with candidate entities of different entity mentions.

Fang et al. [37] and Chen et al. [75] selected entity mentions around the entity mention m of the candidate entity e in Equation 15 in a pre-defined window as the mention set M_s. They obtained the topical coherence feature for the candidate entity by averaging the similarities between the candidate entity embedding and embeddings of assigned mapping entities defined via Equation 17.

A great number of DL based EL methods link entity mentions in a document in a sequential manner and utilize already disambiguated entities to help generate the topical coherence feature for the subsequent candidate entities. When we calculate this feature for some candidate entity e via Equation 15, the mention set M_s is composed of entity mentions which have already been linked before the entity mention m of the candidate entity e and those already disambiguated entities are considered as assigned mapping entities. Specifically, Zwicklbauer et al. [36] created a topic vector, which is set by summing up the representations of already disambiguated entities. Once the linking of some mention is completed, the topic vector changes accordingly. The topical coherence feature for the candidate entity is defined as the similarity between the candidate entity embedding and the topic vector. Nguyen et al. [40] used GRUs to encode already disambiguated entities with an assumption that the hidden state vector of GRUs could summarize the information about the already disambiguated entities based on the characteristics of RNN. The topical coherence feature for the candidate entity is calculated as the similarity between the candidate entity embedding and the current hidden state vector of GRUs. Similarly, Fang et al. [59] utilized an LSTM as the encoder instead. What's more, to generate a topical coherence feature vector for each candidate entity, Yang et al. [52] first calculated the similarities between the candidate entity embedding and the embeddings of already disambiguated entities, and then concatenated the maximal and average similarity score to form the topical coherence feature vector. Some EL methods [41], [59], [72], [73] used a simple to complex strategy, i.e., began to link entity mentions that are easier to be disambiguated and then utilized information provided by already disambiguated entities to generate topical coherence features for subsequent entity mentions that are relatively more difficult to be disambiguated. Yang et al. [60] proposed the dynamic context augmentation (DCA), whose basic idea is to accumulate knowledge from already disambiguated entities as a dynamic context to enhance later decisions. They applied an attention mechanism on DCA to adjust the weights of already disambiguated entities. The topical coherence feature for the candidate entity is obtained by summing up the similarity scores between the candidate entity embedding and the embeddings of already disambiguated entities. Chen et al. [65] also applied DCA in their global model. Gu et al. [73] utilized already disambiguated entities to enhance the later decisions through a dynamic multi-turn way. Specifically, once an entity mention m_{i-1} is linked, the context for identifying the entity mention m_i will be updated by replacing m_{i-1} with its assigned mapping entity e^*_{m-1}. Then the updated context is leveraged to generate the topical coherence feature for m_i using the BERT [83] encoder in each turn. To alleviate error propagation caused by falsely linked entities, a gate mechanism is introduced on the current and historical representations to control which part of history cues should be inherited.

In addition, Phan et al. [44], [78] proposed pair-linking, which is based on an assumption that each candidate entity only needs to be coherent with another candidate entity. They selected the most compatible candidate entity for the candidate entity e in Equation 15 from candidate entities of all entity mentions in the document D as the assigned mapping entity. The topical coherence feature of the candidate entity e could be computed based on local features of the pair of the candidate entity and the assigned mapping entity as well as the similarity between them.

5.5.2 Optimized Methods

Different from simplified methods, optimized methods do not select a temporary mapping entity but keep all candidate entities for the entity mention in the mention set M_s when computing the topical coherence feature. Thus, optimizing the topical coherence objective is intractable due to the large number of candidate entities. Optimized methods usually leverage graph-based approaches to perform optimization as they can dynamically capture the global interdependence between different candidate entities in the document and learn the topical coherence feature for each candidate entity. We will introduce some optimized methods in detail as follows.

We first introduce three EL methods based on Graph Neural Network (GNN) to automatically decide the relevant candidate entities of entity mentions in the mention set and then generate a topical coherence feature for each candidate entity. Cao et al. [53] applied GCN [87] whose basic idea is to enhance the feature of a node according to its neighbor nodes to integrate topical coherence information. The entity mentions around the entity mention m in a pre-defined window are regarded as the mention set M_s. To obtain the topical coherence feature for each candidate entity, they constructed a graph for the candidate entity by taking this candidate entity and all candidate entities of entity mentions in the mention set M_s as nodes and relations between them extracted from a KB as edges. Then they applied a GCN on this graph to aggregate information from linked nodes to the candidate entity node. Given the hidden state $\mathbf{h}_{l-1}$ of the $(l-1)$-th layer, graph convolution is operated to generate the hidden state $\mathbf{h}_l$ as follows:

$$\mathbf{h}_l = f(\mathcal{W}_l(\tilde{A}\mathbf{h}_{l-1}) + b_l) \tag{18}$$

where $\tilde{A}$ is a normalized adjacent matrix of the input graph with self-connection, $\mathcal{W}_l$ and b_l are the weights and bias in the l-th layer respectively, and $f(\cdot)$ represents a non-linear activation function. After $\mathcal{T}$ times of the graph convolution, the hidden state $\mathbf{h}_{\mathcal{T}}$ integrates information from both the candidate entity and its neighbor nodes, and it is treated as a topical coherence feature of the candidate entity.

Similarly, Wu et al. [31] leveraged a dynamic GCN learning paradigm to calculate the topical coherence feature. The graph structure of $\tilde{A}$ is dynamically computed and modified by a graph weight determinator during training in each layer, and it can capture topical coherence better than GCN with a fixed graph structure. Fang et al. [67] utilized a Graph Attention Network (GAT) model to encode the global topical coherence information of the candidate entity. It is noted that GAT is a typical dynamic GCN, which can dynamically change nodes and edges according to the current state. Moreover, with the help of the masked self-attention layers, GAT can implicitly assign different importance to different neighbor nodes and capture the relevance between candidate entities well.

In addition, Ganea and Hofmann [43] formulated this problem based on a binary Conditional Random Field (CRF) model by taking candidate entities of all entity mentions in the document as nodes. Considering the exact maximum a posteriori inference on this CRF is NP-hard, they adopted loopy belief propagation (LBP) as an approximate inference method based on $\mathcal{T}$ times message-passing iterations to produce a marginal probability for each candidate entity as its topical coherence feature. This optimized method is followed by several DL based EL methods [46], [55], [63], [65], [66], [70].

6 Algorithm

Algorithm takes features introduced in the above section as input and outputs the final entity linking result. Specifically, algorithm aims to select the target mapping entity for each entity mention based on the local and global features of its candidate entities. Traditional EL methods pursued some algorithms to select the mapping entity, such as linear model [13], [99], [100], support vector machines [12], [101], logistic classifier [37], [102] and Naïve Bayes classifier [11]. Recently, DL based EL methods that we focus on in this survey used algorithms, such as MLP, PageRank, graph regularization, and reinforcement learning. In this section, we introduce such algorithms by summarizing them into three groups, namely, MLP, graph-based algorithms, and RL.

6.1 MLP

For each candidate entity e of the entity mention m, MLP could transform its local and global features to a ranking score $\Phi(m, e)$ via a learned non-linear transformation. We have introduced its basic structure in Section 3.1. Some DL based EL methods [31], [45], [48], [53], [54], [55] leveraged a single-layer perceptron to encode features to produce a ranking score for the candidate entity. For each entity mention, the candidate entity with the highest ranking score is chosen as the output mapping entity. Additionally, several EL works [43], [46], [50], [51], [57], [60], [61], [62], [63], [65], [66], [67], [70], [71], [74], [75] leveraged an MLP instead.

MLP could be trained using the stochastic gradient descent algorithm and require a loss function to guide the parameter learning. Loss functions are used to evaluate how well the specific algorithm models the given data. If the predicted mapping results output by the algorithm deviate too much from the gold mapping results, loss functions would generate large values. Thus, loss functions need to be minimized during the training of MLP. There are two widely used loss functions which we introduce briefly.

A large quantity of MLP methods [31], [43], [46], [51], [54], [55], [60], [63], [65], [66], [67], [70], [75] leveraged the max-margin loss, which tries to make the ranking score of the gold mapping entity higher than the ranking scores of the other candidate entities by a safety margin. The max-margin loss of a training instance is defined as follows:

$$\mathcal{L} = \max(0, \gamma - \Phi(m, e^+) + \Phi(m, e^-)) \tag{19}$$

where $\Phi(m, e)$ denotes the ranking score of the candidate entity e given the entity mention m, and γ is the safety margin. Each training instance is constructed by a positive gold mapping entity e^+ with a negative entity e^-, with the purpose to make the ranking score of e^+ be at least a margin γ larger than that of e^-.

Cross-entropy loss is another loss function utilized by some MLP methods [45], [53], [61], [71], [74]. The binary cross-entropy loss which increases when the predicted label diverges from the actual label is defined as follows:

$$\mathcal{L} = -(y \log \Phi(m, e) + (1 - y) \log(1 - \Phi(m, e))) \tag{20}$$

where $y \in \{0, 1\}$ denotes the actual label of the candidate entity. If the candidate entity e is the gold mapping entity for the entity mention m, the value of y is 1; otherwise 0. $\Phi(m, e) \in (0, 1)$ indicates the ranking score of the candidate entity e given the entity mention m.

6.2 Graph-Based Algorithms

In general, graph-based algorithms are usually leveraged by collective EL methods, which make decisions on mapping entities jointly for all entity mentions in the same document. Specifically, for each document containing a set of entity mentions, the graph-based algorithm first needs to construct a graph by taking the candidate entities of all entity mentions in the document as nodes and similarity scores between candidate entities as edge weights. Next, a graph-based ranking algorithm is performed on this graph to assign a ranking score to each candidate entity, which represents its degree of importance in the overall graph structure. Finally, for each entity mention, the candidate entity with the highest ranking score is chosen as the output mapping entity. In the following, we introduce graph-based algorithms leveraged by DL based EL methods in detail.

Zwicklbauer et al. [36] applied PageRank on a graph which is composed of the candidate entities of all entity mentions in the same document and a topic vector node introduced in Section 5.5.1. The transition matrix of the graph describes the likelihood of walking from a node to the adjacent node and is calculated as the harmonic mean between two nodes' local scores. They employed the prior popularity feature introduced in Section 5.1 as a jump probability for each candidate entity node. Ultimately, they applied the PageRank algorithm over the constructed graph to compute a ranking score for each candidate entity.

Xue et al. [58] introduced recurrent random-walk layers for collective EL, which reinforce the evidence for related EL decisions into high probability decisions with the help of external KB. The graph constructed for each document contains the candidate entities of all entity mentions in the document. To define the transition matrix between candidate entities, they calculated a relevance score for each pair of candidate entities by summing up their semantic similarity score and the WLM [88] score based on Wikipedia. By introducing $\mathcal{T}$ random-walk layers, they can easily propagate evidence for $\mathcal{T}$ times and produce a ranking score for each candidate entity.

What's more, Huang et al. [34] leveraged graph regularization to perform collective inference. They constructed a graph for each document and each node in the graph represents a pair of entity mention and one of its candidate entities. A weighted edge is added between two nodes if satisfying some constraints and the weight is computed as the semantic similarity between two candidate entities. They first initialized a ranking score for each candidate entity based on the linear combination of local features. Some nodes with high prior popularity feature introduced in Section 5.1 are regarded as labeled seed nodes, which remain unchanged during the graph regularization. Then graph regularization is applied on this graph to refine the ranking scores of unlabeled nodes by means of the labeled seed nodes, with an assumption that two strongly connected nodes should have similar ranking scores.

6.3 Reinforcement Learning

Reinforcement learning (RL) is an area of ML concerned with how software agents ought to perform discrete actions in an environment according to a policy, which is trained to maximize some cumulative rewards [105]. Using RL as a ranking algorithm for entity linking, a set of candidate entities are selected by agents as the entity mapping results which can maximize the sum of expected rewards.

Fang et al. [59] and Yang et al. [60] regarded the EL task as a sequence decision problem and leveraged RL algorithm to obtain the entity mapping results. In their models, the agent is designed as a policy network which can learn a stochastic policy and prevent the agent from getting stuck at an intermediate state. Under the guidance of the policy network, the agent decides which action (i.e., choosing the target mapping entity from candidate entities) should be taken at each state (i.e., current local and global encoding). After performing all decisions in the episode, each action will get an expected reward and the goal is to maximize the total expected rewards $\mathcal{J}$, which is defined as follows:

$$\mathcal{J} = \sum_{l} \sum_{a} \pi(a|st) R(a_l)$$
$$R(a_l) = p(a_l) \sum_{i=l}^{L} p(a_i) + (1 - p(a_l))(\sum_{i=l}^{L} p(a_i) + l - L) \tag{21}$$

where $\pi(a|st)$ is the policy network indicating the probability of taking the action a under the state st, and $R(a_l)$ is the expected reward of the action a at l-th time step. To compute the expected reward $R(a_l)$, $p(a_l) \in \{1, 0\}$ indicates whether the current action is correct or not, and $\sum_{i=l}^{L} p(a_i)$ and $-(\sum_{i=l}^{L} p(a_i) + l - L)$ represent the number of correct actions and that of wrong ones from time l to the end of episode respectively. Accordingly, RL could explore the long-term influence of current selection on subsequent decisions.

7 Data Sets and Evaluation

In this section, we first introduce several widely used real-world EL data sets, tools, and evaluation metrics. Then we give a quantitative performance analysis of representative DL based EL methods.

7.1 Data Sets and Tools

Lots of data sets with different properties (e.g., genre, year, KB, and the number of entity mentions per document) have been used to evaluate EL systems. Table 2 shows an overview of ten well-known public EL data sets used by DL based EL methods. We introduce these data sets in detail as follows:

- **MSNBC** [10] is annotated from MSNBC news articles and contains documents from 10 different domains (i.e., two documents per domain).

TABLE 2: List of ten widely used data sets for entity linking. "# **M.**" refers to the number of entity mentions, "# **D.**" refers to the number of documents, and "# **M./D.**" refers to the number of entity mentions per document in the data set.

Data set (Abbreviation)	Genre	Year	KB	# M.	# D.	# M./D.	URL
MSNBC [10]	news	2007	Wikipedia	656	20	32.80	https://cogcomp.seas.upenn.edu/page /resource_view/4
AQUAINT (AQ) [88]	news	2008	Wikipedia	449	50	8.98	http://community.nzdl.org/wikification /docs.html
TAC-KBP2010 (KBP) [8]	news, blogs	2010	Wikipedia	3750	3684	1.02	https://tac.nist.gov/2010/KBP/data.html
AIDA-CoNLL (AIDA) [15]	news	2011	YAGO/Freebase/ Wikipedia	34587	1393	24.83	http://resources.mpi-inf.mpg.de/yago-naga /aida/download/
ACE2004 (ACE) [13]	news	2011	Wikipedia	257	35	7.34	https://cogcomp.seas.upenn.edu/page /resource_view/4
KORE50 (KORE) [97]	tweets	2012	YAGO/DBpedia	148	50	2.96	http://resources.mpi-inf.mpg.de/yago-naga /aida/download/
N3-RSS500 (RSS) [103]	RSS-feeds	2014	DBpedia	1000	500	2.00	http://aksw.org/Projects/N3NERNEDNIF.html
N3-Reuters128 (Reuters) [103]	news	2014	DBpedia	881	128	6.88	http://aksw.org/Projects/N3NERNEDNIF.html
WNED-CWEB (CW) [104]	news	2016	Wikipedia	11154	320	34.86	https://doi.org/10.7939/DVN/10968
WNED-WIKI (WI) [104]	news	2016	Wikipedia	6821	320	21.32	https://doi.org/10.7939/DVN/10968

- **AQUAINT** [88] contains documents collected from the Xinhua News Service, New York Times, and the Associated Press. Each document contains about 250 to 300 words, where the first entity mention of an entity is manually annotated to Wikipedia.
- **TAC-KBP2010** [8] contains news and blogs from various agencies. It is constructed for the TAC 2010 conference and only contains approximately one entity mention per document which is unsuitable to model the topical coherence feature.
- **AIDA-CoNLL** [15] is an annotated corpus of Reuters news documents. It contains much more documents than other existing EL data sets and is manually linked to KBs by authors. To train and test EL methods, the data set is often divided into three parts: AIDA-Train for training, AIDA-A for validation, and AIDA-B for testing.
- **ACE2004** [13] is a subset of ACE2004 [106] coreference documents annotated using Amazon Mechanical Turk[1].
- **KORE50** [97] is extracted from some microblogging platform (i.e., Twitter). It contains short documents (i.e., tweets) on various domains (e.g., music, business, sports, and celebrities). Each tweet consists of a few sentences with some ambiguous entity mentions, and most entity mentions are first names referring to persons with high level of ambiguity.
- **N3-RSS500** [103] is created using a data set of RSS-feeds (i.e., short formal documents), which are from major international newspapers. The data set covers a wide range of domains, such as world, business, and science.
- **N3-Reuters128** [103] contains economic news documents extracted from the Reuters-21587 corpus[2]. Both N3-RSS500 and N3-Reuters128 are manually annotated by Röder et al. [103].
- **WNED-CWEB** [104] is randomly picked from the FACC1 [107] data set, which provides annotations of mention-entity pairs for ClueWeb 2012 data[3].
- **WNED-WIKI** [104] is crawled from Wikipedia pages with its original anchor text annotations. Both WNED-CWEB and WNED-WIKI are automatically extracted by Guo and Barbosa [104] and are relatively large with a large number of documents.

1. https://www.mturk.com/
2. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
3. http://lemurproject.org/clueweb12

TABLE 3: Off-the-shelf EL tools.

EL tool	URL
StanfordCoreNLP	https://stanfordnlp.github.io/CoreNLP/entitylink. html
spaCy	https://spacy.io/api/entitylinker
TAGME	https://services.d4science.org/web/tagme/ tagme-help
Wikipedia Miner	http://community.nzdl.org/wikification/
DBpedia Spotlight	https://www.dbpedia-spotlight.org/
Babelfy	http://babelfy.org/
AGDISTIS	https://github.com/dice-group/AGDISTIS
WAT	https://services.d4science.org/web/tagme/wat-api
FEL	https://github.com/yahoo/FEL
REL	https://github.com/informagi/REL

In Table 2, it can be seen that each data set has its own characteristics. For example, KORE50 [97] and N3-RSS500 [103] emphasize entity linking over short documents, which are composed of tweets and RSS-feeds respectively, while other data sets are constructed using relatively long news documents. What's more, besides being used for testing, some data sets, particularly larger ones like AIDA-CoNLL [15] and TAC-KBP2010 [8] could be also used for training, while some data sets such as MSNBC [10] and ACE2004 [13] containing a few documents are generally not used for training. Thus, researchers could select appropriate data sets based on the different characteristics of their own EL systems when training or testing.

Lots of off-the-shelf entity linking tools are publicly available online. Table 3 summarizes popular EL tools and their URLs.

7.2 Evaluation Metrics

Evaluation metrics are used to evaluate the performance of EL systems on data sets. GERBIL [108] is a benchmark entity annotation platform that provides a unified comparison among different EL systems across various data sets and metrics. Currently, GERBIL offers six metrics and subdivides them into two groups, namely the macro- and the micro- groups of precision, recall, and F1-measure. The macro- metric is the average of the corresponding metric over each document in the data set $\mathcal{S}$, while the micro- metric takes into account all annotations together thus giving more importance to documents with more entity mentions [109]. The majority of DL based EL methods select micro- metrics for evaluation, whereas some EL works [32], [34], [38], [41], [53], [64], [65] utilize macro- metrics as well. Let $E_\mathcal{S}$, E_D be the gold mapping entity

TABLE 4: The performance of representative DL based EL methods on various data sets taken from both their original papers and the GERBIL platform. The best results are in **boldface** and the second-best results are underlined. Due to the limited space, in the table header we show the abbreviations of the data sets which are defined in the first column of Table 2.

Model	MSNBC	AQ	KBP	AIDA	ACE	KORE	RSS	Reuters	CW	WI
	$F1_{mic}$	$F1_{mic}$	A_{mic}	$F1_{mic}$	$F1_{mic}$	$F1_{mic}$	$F1_{mic}$	$F1_{mic}$	$F1_{mic}$	$F1_{mic}$
He et al. (ACL 2013) [32]	-	-	81.0	85.6	-	-	-	-	-	-
Sun et al. (IJCAI 2015) [33]	-	-	83.9	-	-	-	-	-	-	-
DSRM (arXiv 2015) [34]	-	-	-	86.6	-	-	-	-	-	-
EDKate (CoNLL 2016) [37]	75.5	85.2	88.9	-	80.8	-	-	-	-	-
Zwicklbauer et al. (SIGIR 2016) [36]	91.1	84.2	-	78.4	90.7	-	-	-	-	-
Francis-Landau et al. (NAACL 2016) [39]	-	89.9	-	85.5	-	-	-	-	-	-
Nguyen et al. (COLING 2016) [40]	-	-	-	87.2	89.7	-	-	-	-	-
Globerson et al. (ACL 2016) [35]	-	-	87.2	92.7	-	-	-	-	-	-
Yamada et al. (CoNLL 2016) [38]	-	-	85.5	93.1	-	-	-	-	-	-
Gupta et al. (EMNLP 2017) [42]	-	-	-	82.9	90.7	-	-	-	-	-
Eshel et al. (CoNLL 2017) [45]	-	-	-	87.3	-	-	-	-	-	-
Deep-ED (EMNLP 2017) [43]	93.7	88.5	-	92.2	88.5	-	-	-	77.9	77.5
NeuPL (CIKM 2017) [44]	91.8	-	-	-	**92.9**	**79.4**	**80.0**	**91.6**	-	-
NCEL (COLING 2018) [53]	-	87.0	91.0	80.0	88.0	-	-	-	-	86.0
Kolitsas et al. (CoNLL 2018) [51]	86.4	-	-	83.1	-	60.8	68.6	67.3	-	-
SGTB-BiBSG (NAACL 2018) [52]	92.6	89.9	-	93.0	88.5	-	-	-	**81.8**	79.2
MR-Deep-ED (ACL 2018) [46]	93.9	88.3	-	93.1	89.9	-	-	-	77.5	78.0
Sil et al. (AAAI 2018) [48]	-	-	87.4	94.0	-	-	-	-	-	-
DeepType (AAAI 2018) [49]	-	-	90.9	94.9	-	-	-	-	-	-
Le and Titov (ACL 2019) [54]	-	-	-	81.5	-	-	-	-	-	-
Gillick et al. (CoNLL 2019) [61]	-	-	87.0	-	-	-	-	-	-	-
Le and Titov (ACL 2019) [55]	92.2	90.7	-	89.7	88.1	-	-	-	78.2	81.7
RRWEL (IJCAI 2019) [58]	94.4	91.9	-	92.4	90.6	-	-	-	79.7	85.5
E-ELMo (arXiv 2019) [62]	92.3	90.1	88.3	93.5	88.7	-	-	-	78.4	79.8
RLEL (WWW 2019) [59]	92.8	87.5	-	94.3	91.2	-	-	-	78.5	82.8
DCA-RL (EMNLP 2019) [60]	93.8	88.3	-	93.7	90.1	-	-	-	75.6	78.8
DCA-SL (EMNLP 2019) [60]	94.6	87.4	-	94.6	89.4	-	-	-	73.5	78.2
SeqGAT (WWW 2020) [67]	80.0	88.0	-	83.0	89.0	68.0	68.0	71.0	-	-
REL (SIGIR 2020) [66]	85.8	-	-	84.0	-	54.0	64.1	64.9	-	-
ET4EL (AAAI 2020) [64]	-	-	-	85.9	-	-	-	-	-	-
FGS2EE (ACL 2020) [63]	94.2	88.5	-	92.6	90.7	-	-	-	77.4	77.8
DGCN (WWW 2020) [31]	92.5	89.4	-	93.1	90.6	-	-	-	81.2	77.6
Chen et al. (AAAI 2020) [65]	-	89.8	-	93.4	-	-	-	-	77.9	80.1
GNED (KBS 2020) [70]	95.5	91.6	-	92.4	90.1	-	-	-	77.5	78.5
BLINK (EMNLP 2020) [68]	-	-	**94.5**	-	-	-	-	-	-	-
Yamada et al. (arXiv 2020) [72]	**96.3**	**93.5**	-	**95.0**	91.9	-	-	-	78.9	**89.1**
M3 (AAAI 2021) [73]	-	-	-	-	-	74.3	-	-	-	-
CHOLAN (EACL 2021) [76]	83.4	76.8	-	85.7	86.8	-	-	-	-	-

annotations associated with all the entity mentions $M_{\mathcal{S}}$ in the data set $\mathcal{S}$, and a set of entity mentions M_D in the document D respectively. Let $G_{\mathcal{S}}$, G_D be the output entity annotations of some EL system associated with all the entity mentions $M_{\mathcal{S}}$ in the data set $\mathcal{S}$, and a set of entity mentions M_D in the document D respectively. We give the definitions of the evaluation metrics, i.e., precision, recall, and F1-measure of both macro- and micro-groups as follows:

$$\begin{aligned} P_{mac} &= (\textstyle\sum_{D\in\mathcal{S}} \frac{|E_D \cap G_D|}{|G_D|})/|\mathcal{S}| \\ R_{mac} &= (\textstyle\sum_{D\in\mathcal{S}} \frac{|E_D \cap G_D|}{|E_D|})/|\mathcal{S}| \\ F1_{mac} &= 2 \cdot P_{mac} \cdot R_{mac}/(P_{mac} + R_{mac}) \\ P_{mic} &= |E_{\mathcal{S}} \cap G_{\mathcal{S}}|/|G_{\mathcal{S}}| \\ R_{mic} &= |E_{\mathcal{S}} \cap G_{\mathcal{S}}|/|E_{\mathcal{S}}| \\ F1_{mic} &= 2 \cdot P_{mic} \cdot R_{mic}/(P_{mic} + R_{mic}) \end{aligned} \tag{22}$$

The precision is computed as the fraction of correctly linked entity mentions that are generated by the EL system, and the recall is computed as the fraction of correctly linked entity mentions that should be correctly linked. To take into consideration both of them, the F1-measure puts them together, and it is defined as the harmonic mean of the precision and recall. In addition, the accuracy refers to the ratio of the number of entity mentions that are correctly linked to the total number of entity mentions in the data set, and it is defined as follows:

$$\begin{aligned} A_{mac} &= (\textstyle\sum_{D\in\mathcal{S}} \frac{|E_D \cap G_D|}{|M_D|})/|\mathcal{S}| \\ A_{mic} &= |E_{\mathcal{S}} \cap G_{\mathcal{S}}|/|M_{\mathcal{S}}| \end{aligned} \tag{23}$$

7.3 Performance Analysis

Table 4 presents the performance of representative DL based EL methods on data sets introduced in Section 7.1. Most recent EL works used the GERBIL [108] platform to report their performance. To ensure correctness, we collect the experimental results from both their original papers and the GERBIL platform. The micro-F1 metric is commonly used by DL based EL methods to report the performance. Therefore, we show micro-F1 scores for all data sets except TAC-KBP2010 [8] since the micro-accuracy is regarded as the official evaluation metric in the TAC-KBP track.

In Table 4, we can see that DL based EL methods have achieved the state-of-the-art performance on all data sets, which demonstrates the effectiveness of DL. Specifically, Yamada et al. [72] achieves the best results on four data sets (i.e., MSNBC [10], AQUAINT [88], AIDA-CoNLL [15], and WNED-WIKI [104]), Wu et al. [68] performs best on TAC-KBP2010 [8], NeuPL [44] owns the leadership on four data sets (i.e., ACE2004 [13], KORE50 [97], N3-RSS500 [103], and N3-Reuters128 [103]), and SGTB-BiBSG [52] obtains the best result on WNED-CWEB [104].

It can be found from Table 4 that Transformers-based EL systems (e.g., Yamada et al. [72], Wu et al. [68], and Chen et al. [65]) achieve advanced performance on many data sets as they are pre-trained on huge corpora and can generate more sophisticated long-distance feature representations for entity linking. The good performance of models leveraging type information (e.g., DeepType [49], FGS2EE [63], and Chen et al. [65]) demonstrates the effectiveness of type information and points out a promising direction for entity linking. Specifically, DeepType [49] that only leverages the prior popularity feature and the type similarity feature for linking achieves the second-best result on AIDA-CoNLL, which is amazing and enlightening.

It is also noted that no perfect EL system can achieve the best results on all data sets due to the different characteristics of various data sets, such as the document genre, the document length, and the number of entity mentions per document. That is, the best EL method on one data set may perform poorly on other data sets. For example, Transformers-based EL method Yamada et al. [72] obtains four best results and one second-best result over five data sets, but performs not well on WNED-CWEB since documents in this data set are significantly longer than documents in other data sets. There are approximately 1700 words per document in the WNED-CWEB data set, which is much longer than the maximum word length that BERT can deal with (i.e., 512 words). Nevertheless, SGTB-BiBSG [52] performs excellent on WNED-CWEB because this model designs various hand-crafted features capturing document-level contextual information. What's more, NeuPL [44] performs best on data sets containing short documents such as KORE50 and N3-RSS500 since the pair-linking algorithm proposed in the NeuPL model iteratively identifies and resolves pairs of entity mentions without the requirement of much global information. Some EL methods (e.g., DCA-SL [60], Deep-ED [43], and DGCN [31]) mainly based on the topical coherence feature perform well on data sets such as MSNBC and AIDA-CoNLL which have tens of entity mentions per document, because they can capture the global interdependence between candidate entities of entity mentions in a document well.

Overall, the entity linking task is highly data and domain dependent and it is unlikely that a technique dominates all others across all types of data sets. For a given data set for entity linking, we should leverage suitable embeddings, features, and algorithms to obtain advanced results based on the characteristics of the data.

8 Future Directions

Based on the above review and analysis, we believe that there is still much space for further enhancement in this field. In this section, we discuss the remaining limitations of existing EL methods and list some directions for further exploration in EL research.

Multi-source heterogeneous text data. In the era of big data, text data have multi-source heterogeneous characteristics. Text may come from diverse data sources in various structures. For example, news documents from news websites are relatively long and formal. Web tables shown in web pages are structured. Queries from search engine logs are often short and noncontiguous. Reviews from e-commerce websites are usually colloquial and noisy. Entities may appear in those multi-source heterogeneous text data which contain abundant knowledge about them. Bridging the multi-source heterogeneous text data with KBs is beneficial for the understanding of the text data and the enrichment of KBs. Present EL studies mainly focus on linking entity mentions in common text data (e.g., news documents [38], [43], [46], [72], tweets [94], [110], [111], and web tables [112], [113], [114]). As different types of text data have various characteristics, existing EL methods may not be applicable or be difficult to achieve satisfactory linking performance when dealing with other types of text data (e.g., search queries, reviews, community question answering (CQA) text, and open information extraction (OIE) triples). Therefore, it is very meaningful and essential to develop EL techniques to link entities in these diverse types. Although some works have preliminarily addressed the entity linking task for search queries [115], [116], [117], CQA text [118], and OIE triples [119], [120] respectively, we believe there are still many opportunities for substantial improvement.

Joint NER and EL. NER is the task to identify text spans that mention named entities, and to classify them into pre-defined categories, such as person, location, and organization [121]. It serves as a preceded task for entity linking, which provides the boundaries of named entities in text. However, detecting the correct entity mentions is challenging, especially for informal short noisy text that often contains phrases with ambiguous meanings. Therefore, NER is often the performance bottleneck of EL since the performance of NER leads to an upper limit to the performance of EL. Some EL methods are designed to jointly perform NER and EL, which allows each subtask to benefit from another and alleviates error propagations that are unavoidable in pipeline settings. So far, Guo et al. [94] utilized a structured support vector machines algorithm for tweet entity linking that jointly optimizes NER and EL as an end-to-end task. Kolitsas et al. [51] and Broscheit [122] proposed neural end-to-end EL systems that jointly discover and link entities in the news documents. Li et al. [69] designed an end-to-end EL system used for downstream question answering systems. In summary, we consider it is worth exploring effective approaches for jointly performing NER and EL for real applications in the future.

More advanced language models. Neural language models have revolutionized the field of NLP due to their superior expressive power. As introduced earlier, many effective language models have been applied in the field of EL and achieved great success. Recently, there are many more advanced language models being developed and available. For instance, BERT [83], widely leveraged by existing EL works, has been exceeded by several variants and other transformer-based models, which made major changes to loss functions, model architecture, and pre-training objectives. Specifically, RoBERTa [123] is more robust than BERT which is trained using much more training data and leveraging dynamic masking rather than static masking. SpanBERT [124] extends BERT to better represent and predict text spans. XLNet [125] is a generalized order-aware autoregressive language model which makes use of a permutation operation to perform better than BERT on lots of NLP tasks. Moreover, GPT-3 [126] is the largest pre-trained language model by far with hundreds of billions of parameters, which has shown great abilities in zero-shot, one-shot, and few-shot settings. In summary, we consider there is still much space for further enhancement of EL by leveraging these more advanced language models in modeling text semantics.

EL model robustness. The robustness of deep learning has received great attention recently. A DL model is considered to be robust if its output label is consistently accurate even if one or more of the input features or assumptions are drastically changed due to unforeseen circumstances. For entity linking, robustness refers to the ability to achieve consistent performance over a wide range of data sets with different properties, such as different domains, text structures, and knowledge bases [36]. However, most existing DL based EL systems were designed and optimized for a specific domain (e.g., general domain or biomedical domain), for a specific text structure (e.g., news document or tweet), and for a specific knowledge base (e.g., Wikipedia or Freebase). This leads to very specialized models that lack robustness and are applicable for very specific tasks. Therefore, we strongly believe that robust DL based EL models that generalize well deserve much deeper exploration by the community.

9 Conclusion

Applying deep learning to entity linking has become a popular research topic today. In this survey, we give a comprehensive and detailed review of the existing DL based EL methods. We first propose a new taxonomy which categories DL techniques for ranking candidate entities based on three axes (i.e., embedding, feature, and algorithm). Second, we systematically review the representative DL based EL methods according to the taxonomy. Third, we present ten widely used real-world entity linking data sets and a quantitative performance analysis of DL based EL methods in tabular form. Finally, we discuss some limitations and highlight several future research directions. Through this paper, we hope to demonstrate the progress and problem in existing entity linking research and encourage more improvements in this area.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grant No. U1936206 and 61772289, Natural Science Foundation of Tianjin under Grant No. 19JC-QNJC00100, YESS by CAST under Grant No. 2019QNRC001, and CAAI-Huawei MindSpore Open Fund. Jianyong Wang was supported in part by National Key Research and Development Program of China under Grant No. 2020YFA0804503, National Natural Science Foundation of China under Grant No. 61532010 and 61521002, and Beijing Academy of Artificial Intelligence (BAAI).

References

[1] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW*, 2007, pp. 697–706.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives, "Dbpedia: A nucleus for a web of open data," in *ISWC*, 2007, pp. 722–735.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD*, 2008, pp. 1247–1250.

[4] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *SIGMOD*, 2012, pp. 481–492.

[5] W. Shen, J. Wang, and J. Han, "Entity linking with a knowledge base: Issues, techniques, and solutions," *TKDE*, vol. 27, no. 2, pp. 443–460, 2014.

[6] Y. Zhang, S. He, K. Liu, and J. Zhao, "A joint model for question answering over multiple knowledge bases," in *AAAI*, 2016, pp. 3094–3100.

[7] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," in *ACL*, 2016, pp. 2124–2133.

[8] H. Ji and R. Grishman, "Knowledge base population: Successful approaches and challenges," in *ACL*, 2011, pp. 1148–1158.

[9] M. Michelson and S. A. Macskassy, "Discovering users' topics of interest on twitter: a first look," in *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, 2010, pp. 73–80.

[10] S. Cucerzan, "Large-scale named entity disambiguation based on wikipedia data," in *EMNLP-CoNLL*, 2007, pp. 708–716.

[11] V. Varma, P. Pingali, R. Katragadda, S. Krishna, S. Ganesh, K. Sarvabhotla, H. Garapati, H. Gopisetty, V. B. Reddy, K. Reddy *et al.*, "Iiit hyderabad at tac 2009." in *Text Analysis Conference 2009 Workshop*, 2009.

[12] Z. Chen and H. Ji, "Collaborative ranking: A case study on entity linking," in *EMNLP*, 2011, pp. 771–781.

[13] L. Ratinov, D. Roth, D. Downey, and M. Anderson, "Local and global algorithms for disambiguation to wikipedia," in *ACL*, 2011, pp. 1375–1384.

[14] W. Shen, J. Han, J. Wang, X. Yuan, and Z. Yang, "Shine+: A general framework for domain-specific entity linking with heterogeneous information networks," *TKDE*, vol. 30, no. 2, pp. 353–366, 2017.

[15] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, "Robust disambiguation of named entities in text," in *EMNLP*, 2011, pp. 782–792.

[16] O. Sevgili, A. Shelmanov, M. Arkhipov, A. Panchenko, and C. Biemann, "Neural entity linking: A survey of models based on deep learning," *arXiv preprint arXiv:2006.00575*, 2020.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[19] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[26] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *SIGMOD*, 2018, pp. 19–34.

[27] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011, pp. 315–323.

[28] Y. Bengio, "Deep learning of representations: Looking forward," in *SLPS*, 2013, pp. 1–37.

[29] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[30] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *JMLR*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.

[31] J. Wu, R. Zhang, Y. Mao, H. Guo, M. Soflaei, and J. Huai, "Dynamic graph convolutional networks for entity linking," in *WWW*, 2020, pp. 1149–1159.

[32] Z. He, S. Liu, M. Li, M. Zhou, L. Zhang, and H. Wang, "Learning entity representation for entity disambiguation," in *ACL*, 2013, pp. 30–34.

[33] Y. Sun, L. Lin, D. Tang, N. Yang, Z. Ji, and X. Wang, "Modeling mention, context and entity with neural networks for entity disambiguation." in *IJCAI*, 2015, pp. 1333–1339.

[34] H. Huang, L. Heck, and H. Ji, "Leveraging deep neural networks and knowledge graphs for entity disambiguation," *arXiv preprint arXiv:1504.07678*, 2015.

[35] A. Globerson, N. Lazic, S. Chakrabarti, A. Subramanya, M. Ringgaard, and F. Pereira, "Collective entity resolution with multi-focal attention," in *ACL*, 2016, pp. 621–631.

[36] S. Zwicklbauer, C. Seifert, and M. Granitzer, "Robust and collective entity disambiguation through semantic embeddings," in *SIGIR*, 2016, pp. 425–434.

[37] W. Fang, J. Zhang, D. Wang, Z. Chen, and M. Li, "Entity disambiguation by knowledge and text jointly embedding," in *CoNLL*, 2016, pp. 260–269.

[38] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Joint learning of the embedding of words and entities for named entity disambiguation," in *CoNLL*, 2016, pp. 250–259.

[39] M. Francis-Landau, G. Durrett, and D. Klein, "Capturing semantic similarity for entity linking with convolutional neural networks," in *NAACL*, 2016, pp. 1256–1261.

[40] T. H. Nguyen, N. R. Fauceglia, M. R. Muro, O. Hassanzadeh, A. Gliozzo, and M. Sadoghi, "Joint learning of local and global features for entity linking via neural networks," in *COLING*, 2016, pp. 2310–2320.

[41] Y. Cao, L. Huang, H. Ji, X. Chen, and J. Li, "Bridge text and knowledge by learning multi-prototype entity mention embedding," in *ACL*, 2017, pp. 1623–1633.

[42] N. Gupta, S. Singh, and D. Roth, "Entity linking via joint encoding of types, descriptions, and context," in *EMNLP*, 2017, pp. 2681–2690.

[43] O.-E. Ganea and T. Hofmann, "Deep joint entity disambiguation with local neural attention," in *EMNLP*, 2017, pp. 2619–2629.

[44] M. C. Phan, A. Sun, Y. Tay, J. Han, and C. Li, "Neupl: Attention-based semantic matching and pair-linking for entity disambiguation," in *CIKM*, 2017, pp. 1667–1676.

[45] Y. Eshel, N. Cohen, K. Radinsky, S. Markovitch, I. Yamada, and O. Levy, "Named entity disambiguation for noisy text," in *CoNLL*, 2017, pp. 58–68.

[46] P. Le and I. Titov, "Improving entity linking by modeling latent relations between mentions," in *ACL*, 2018, pp. 1595–1604.

[47] S. Moon, L. Neves, and V. Carvalho, "Multimodal named entity disambiguation for noisy social media posts," in *ACL*, 2018, pp. 2000–2008.

[48] A. Sil, G. Kundu, R. Florian, and W. Hamza, "Neural cross-lingual entity linking," in *AAAI*, 2017, pp. 5464–5472.

[49] J. Raiman and O. Raiman, "Deeptype: Multilingual entity linking by neural type system evolution," in *AAAI*, 2018, pp. 5406–5413.

[50] D. Mueller and G. Durrett, "Effective use of context in noisy entity linking," in *EMNLP*, 2018, pp. 1024–1029.

[51] N. Kolitsas, O.-E. Ganea, and T. Hofmann, "End-to-end neural entity linking," in *CoNLL*, 2018, pp. 519–529.

[52] Y. Yang, O. İrsoy, and K. S. Rahman, "Collective entity disambiguation with structured gradient tree boosting," in *NAACL*, 2018, pp. 777–786.

[53] Y. Cao, L. Hou, J. Li, and Z. Liu, "Neural collective entity linking," in *COLING*, 2018, pp. 675–686.

[54] P. Le and I. Titov, "Distant learning for entity linking with automatic noise detection," in *ACL*, 2019, pp. 4081–4090.

[55] P. Le and I. Titov, "Boosting entity linking performance by leveraging unlabeled documents," in *ACL*, 2019, pp. 1935–1945.

[56] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee, "Zero-shot entity linking by reading entity descriptions," in *ACL*, 2019, pp. 3449–3460.

[57] Ö. Sevgili, A. Panchenko, and C. Biemann, "Improving neural entity disambiguation with graph embeddings," in *ACL*, 2019, pp. 315–322.

[58] M. Xue, W. Cai, J. Su, L. Song, Y. Ge, Y. Liu, and B. Wang, "Neural collective entity linking based on recurrent random walk network learning," in *IJCAI*, 2019, pp. 5327–5333.

[59] Z. Fang, Y. Cao, Q. Li, D. Zhang, Z. Zhang, and Y. Liu, "Joint entity linking with deep reinforcement learning," in *WWW*, 2019, pp. 438–447.

[60] X. Yang, X. Gu, S. Lin, S. Tang, Y. Zhuang, F. Wu, Z. Chen, G. Hu, and X. Ren, "Learning dynamic context augmentation for global entity linking," in *EMNLP-IJCNLP*, 2019, pp. 271–281.

[61] D. Gillick, S. Kulkarni, L. Lansing, A. Presta, J. Baldridge, E. Ie, and D. Garcia-Olano, "Learning dense representations for entity retrieval," in *CoNLL*, 2019, pp. 528–537.

[62] H. Shahbazi, X. Z. Fern, R. Ghaeini, R. Obeidat, and P. Tadepalli, "Entity-aware elmo: Learning contextual entity representation for entity disambiguation," *arXiv preprint arXiv:1908.05762*, 2019.

[63] F. Hou, R. Wang, J. He, and Y. Zhou, "Improving entity linking through semantic reinforced entity embeddings," in *ACL*, 2020, pp. 6843–6848.

[64] Y. Onoe and G. Durrett, "Fine-grained entity typing for domain independent entity linking." in *AAAI*, 2020, pp. 8576–8583.

[65] S. Chen, J. Wang, F. Jiang, and C.-Y. Lin, "Improving entity linking by modeling latent entity type information," in *AAAI*, 2020, pp. 7529–7537.

[66] J. M. van Hulst, F. Hasibi, K. Dercksen, K. Balog, and A. P. de Vries, "Rel: An entity linker standing on the shoulders of giants," in *SIGIR*, 2020, pp. 2197–2200.

[67] Z. Fang, Y. Cao, R. Li, Z. Zhang, Y. Liu, and S. Wang, "High quality candidate generation and sequential graph attention network for entity linking," in *WWW*, 2020, pp. 640–650.

[68] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, "Scalable zero-shot entity linking with dense entity retrieval," in *EMNLP*, 2020, pp. 6397–6407.

[69] B. Z. Li, S. Min, S. Iyer, Y. Mehdad, and W.-t. Yih, "Efficient one-pass end-to-end entity linking for questions," in *EMNLP*, 2020, pp. 6433–6441.

[70] L. Hu, J. Ding, C. Shi, C. Shao, and S. Li, "Graph neural entity disambiguation," *KBS*, vol. 195, p. 105620, 2020.

[71] O. Adjali, r. Besancon, o. Ferret, H. Le Borgne, and B. Grau, "Multimodal entity linking for tweets," in *ECIR*, 2020.

[72] I. Yamada, K. Washio, H. Shindo, and Y. Matsumoto, "Global entity disambiguation with pretrained contextualized embeddings of words and entities," *arXiv preprint arXiv:1909.00426*, 2020.

[73] Y. Gu, X. Qu, Z. Wang, B. Huai, N. J. Yuan, and X. Gui, "Read, retrospect, select: An mrc framework to short text entity linking," in *AAAI*, 2021, pp. 12 920–12 928.

[74] H. Tang, X. Sun, B. Jin, and F. Zhang, "A bidirectional multi-paragraph reading model for zero-shot entity linking," in *AAAI*, 2021, pp. 13 889–13 897.

[75] L. Chen, G. Varoquaux, and F. M. Suchanek, "A lightweight neural model for biomedical entity linking," in *AAAI*, 2021, pp. 12 657–12 665.

[76] M. P. K. Ravi, K. Singh, I. O. Mulang, S. Shekarpour, J. Hoffart, and J. Lehmann, "Cholan: A modular approach for neural entity linking on wikipedia and wikidata," in *EACL*, 2021, pp. 504–514.

[77] L. Zhang, Z. Li, and Q. Yang, "Attention-based multimodal entity linking with high-quality images," in *DASFAA*, 2021, pp. 533–548.

[78] M. C. Phan, A. Sun, Y. Tay, J. Han, and C. Li, "Pair-linking for collective entity disambiguation: Two could be better than all," *TKDE*, vol. 31, no. 7, pp. 1383–1396, 2018.

[79] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL*, 2018, pp. 2227–2237.

[80] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *NAACL*, 2016, pp. 260–270.

[81] M. Pagliardini, P. Gupta, and M. Jaggi, "Unsupervised learning of sentence embeddings using compositional n-gram features," in *NAACL*, 2018, pp. 528–540.

[82] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *AAAI*, 2016, pp. 2741–2749.

[83] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.

[84] E. Charton, M.-J. Meurs, L. Jean-Louis, and M. Gagnon, "Improving entity linking using surface form refinement." in *LREC*, 2014, pp. 4609–4615.

[85] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008, pp. 1096–1103.

[86] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014, pp. 1188–1196.

[87] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2016.

[88] D. Milne and I. H. Witten, "Learning to link with wikipedia," in *CIKM*, 2008, pp. 509–518.

[89] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.

[90] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 1–9.

[91] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013, pp. 1631–1642.

[92] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *ACL*, 2014, pp. 302–308.

[93] P. Xu and D. Barbosa, "Neural fine-grained entity type classification with hierarchy-aware loss," in *NAACL*, 2018, pp. 16–25.

[94] S. Guo, M.-W. Chang, and E. Kiciman, "To link or not to link? a study on end-to-end tweet entity linking," in *NAACL*, 2013, pp. 1020–1030.

[95] W. Zhang, C. L. Tan, Y. C. Sim, and J. Su, "Nus-i2r: Learning a combined system for entity linking." in *Text Analysis Conference 2010 Workshop*, 2010.

[96] T. Štajner and D. Mladenić, "Entity resolution in texts using statistical learning and ontologies," in *ASWC*, 2009, pp. 91–104.

[97] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum, "Kore: keyphrase overlap relatedness for entity disambiguation," in *CIKM*, 2012, pp. 545–554.

[98] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep learning based text classification: A comprehensive review,” *arXiv preprint arXiv:2004.03705*, 2020.

[99] W. Shen, J. Wang, P. Luo, and M. Wang, “Linden: linking named entities with knowledge base via semantic knowledge,” in *WWW*, 2012, pp. 449–458.

[100] W. Shen, J. Wang, P. Luo, and M. Wang, “Liege: link entities in web lists with knowledge base,” in *SIGKDD*, 2012, pp. 1424–1432.

[101] A. Pilz and G. Paaß, “From names to entities using thematic context distance,” in *CIKM*, 2011, pp. 857–866.

[102] S. Monahan, J. Lehmann, T. Nyberg, J. Plymale, and A. Jung, “Cross-lingual cross-document coreference with entity linking.” in *Text Analysis Conference 2011 Workshop*, 2011.

[103] M. Röder, R. Usbeck, S. Hellmann, D. Gerber, and A. Both, “N^3-a collection of datasets for named entity recognition and disambiguation in the nlp interchange format.” in *LREC*, 2014, pp. 3529–3533.

[104] Z. Guo and D. Barbosa, “Robust named entity disambiguation with random walks,” *Semantic Web*, vol. 9, no. 4, pp. 459–479, 2018.

[105] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *JAIR*, vol. 4, pp. 237–285, 1996.

[106] G. R. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel, “The automatic content extraction (ace) program–tasks, data, and evaluation,” in *LREC*, 2004.

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

[108] R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann *et al.*, “Gerbil: general entity annotator benchmarking framework,” in *WWW*, 2015, pp. 1133–1143.

[109] M. Cornolti, P. Ferragina, and M. Ciaramita, “A framework for benchmarking entity-annotation systems,” in *WWW*, 2013, pp. 249–260.

[110] W. Shen, Y. Yin, Y. Yang, J. Han, J. Wang, and X. Yuan, “Toward tweet entity linking with heterogeneous information networks,” *TKDE*, 2021.

[111] W. Hua, K. Zheng, and X. Zhou, “Microblog entity linking with social temporal context,” in *SIGMOD*, 2015, pp. 1761–1775.

[112] S. Zhang, E. Meij, K. Balog, and R. Reinanda, “Novel entity discovery from web tables,” in *WWW*, 2020, pp. 1298–1308.

[113] D. Ritze, O. Lehmberg, Y. Oulabi, and C. Bizer, “Profiling the potential of web tables for augmenting cross-domain knowledge bases,” in *WWW*, 2016, pp. 251–261.

[114] C. S. Bhagavatula, T. Noraset, and D. Downey, “Tabel: Entity linking in web tables,” in *ISWC*, 2015, pp. 425–441.

[115] C. Tan, F. Wei, P. Ren, W. Lv, and M. Zhou, “Entity linking for queries by searching wikipedia sentences,” in *EMNLP*, 2017, pp. 68–77.

[116] R. Blanco, G. Ottaviano, and E. Meij, “Fast and space-efficient entity linking for queries,” in *WSDM*, 2015, pp. 179–188.

[117] M. Cornolti, P. Ferragina, M. Ciaramita, S. Rüd, and H. Schütze, “A piggyback system for joint entity mention detection and linking in web queries,” in *WWW*, 2016, pp. 567–578.

[118] F. Wang, W. Wu, Z. Li, and M. Zhou, “Named entity disambiguation for questions in community question answering,” *KBS*, vol. 126, pp. 68–77, 2017.

[119] X. Lin, H. Li, H. Xin, Z. Li, and L. Chen, “Kbpearl: a knowledge base population system supported by joint entity and relation linking,” *Proc. VLDB Endow.*, vol. 13, no. 7, pp. 1035–1049, 2020.

[120] Y. Liu, W. Shen, Y. Wang, J. Wang, Z. Yang, and X. Yuan, “Joint open knowledge base canonicalization and linking,” in *SIGMOD*, 2021, pp. 2253–2261.

[121] J. Li, A. Sun, J. Han, and C. Li, “A survey on deep learning for named entity recognition,” *TKDE*, 2020.

[122] S. Broscheit, “Investigating entity knowledge in bert with simple neural end-to-end entity linking,” in *CoNLL*, 2019, pp. 677–685.

[123] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[124] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, “Spanbert: Improving pre-training by representing and predicting spans,” *TACL*, vol. 8, pp. 64–77, 2020.

[125] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: generalized autoregressive pretraining for language understanding,” in *NIPS*, 2019, pp. 5753–5763.

[126] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.

Wei Shen received the PhD degree in computer science from Tsinghua University, China, in 2014. He is an associate professor in the College of Computer Science, Nankai University, China. His research interests include entity linking, knowledge base population, and text mining. He was a recipient of ACM China Rising Star Award (Honorable Mention), CCF-Intel Young Faculty Researcher Program, and CAAI Outstanding Doctoral Dissertation Award.

Yuhan Li received his BS degree from Northeast Forestry University, China in 2020. He is currently a master candidate at Nankai University. His research interests include knowledge graph, entity linking and data mining.

Yinan Liu received the BS degree from Northeastern University, China, in 2015 and the MS degree from Nankai University in 2018. He is a PhD student in the College of Computer Science in Nankai University. His research interests include knowledge base population and data mining.

Jiawei Han is Michael Aiken Chair Professor in the Department of Computer Science, University of Illinois at Urbana-Champaign. He received ACM SIGKDD Innovation Award (2004), IEEE Computer Society Technical Achievement Award (2005), IEEE Computer Society W. Wallace McDowell Award (2009), and Japan’s Funai Achievement Award (2018). He is Fellow of ACM and Fellow of IEEE and served as the Director of Information Network Academic Research Center (INARC) (2009-2016) supported by the Network Science-Collaborative Technology Alliance (NS-CTA) program of U.S. Army Research Lab and co-Director of KnowEnG, a Center of Excellence in Big Data Computing (2014-2019), funded by NIH Big Data to Knowledge (BD2K) Initiative.

Jianyong Wang is currently a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received his PhD degree in Computer Science in 1999 from the Institute of Computing Technology, Chinese Academy of Sciences. His research interests mainly include data mining and Web information management. He is serving or ever served as a PC member for some leading international conferences, such as SIGKDD, VLDB, ICDE, WWW, and an associate editor of IEEE TKDE and ACM TKDD. He is a Fellow of the IEEE, a member of the ACM.

Xiaojie Yuan received the BS, MS, and the PhD degree in computer science from Nankai University. She is currently working as a professor of College of Computer Science, Nankai University. She leads a research group working on topics of database, data mining and information retrieval.