World Wide Web (2021) 24:351–373
https://doi.org/10.1007/s11280-020-00845-4

LoG: a locally-global model for entity disambiguation

Kexuan Xin[1] · Wen Hua[1] · Yu Liu[1] · Xiaofang Zhou[1]

Received: 26 December 2019 / Revised: 23 July 2020 / Accepted: 21 September 2020 /
Published online: 22 October 2020

Abstract
Entity disambiguation (ED) aims to link textual mentions in a document to the correct named entities in a knowledge base (KB). Although global ED models usually outperform local models by collectively linking mentions based on the topical coherence assumption, they may still incur incorrect entity assignment when a document contains multiple topics. Therefore, we propose a Locally-Global model (LoG) for ED which extracts global features locally, i.e., among a limited number of neighboring mentions, to combine the respective superiority of both models. In particular, we derive mention neighbors according to the syntactic distance on a dependency parse tree, and propose a tree connection method CoSimTC to measure the cross-tree distance between mentions. We also recognize the importance of keywords in a document for collective entity disambiguation, which reveal the central topic information of the document. Hence, we propose a keyword extraction method Sent2Word to detect keywords of each document. Furthermore, we extend the Graph Attention Network (GAT) to integrate both local and global features to produce a discriminative representation for each candidate entity. Our experimental results on six widely-adopted public datasets demonstrate better performance compared with state-of-the-art ED approaches. The high efficiency of the LoG model further verifies its feasibility in practice.

Keywords Entity linking · Dependency parse tree · Cross-sentence distance · Keyword extraction · Graph attention network

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2019*
Guest Editors: Reynold Cheng, Nikos Mamoulis, and Xin Huang

✉ Wen Hua
w.hua@uq.edu.au

Kexuan Xin
ke.xin@uqconnect.edu.au

Yu Liu
yu.liu@uq.edu.au

Xiaofang Zhou
zxf@itee.uq.edu.au

[1] School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia

1 Introduction

Entity disambiguation (ED), which is also known as entity linking (EL), is one of the fundamental preprocessing tasks in Natural Language Processing (NLP), which can benefit various applications such as information retrieval, question answering, machine translation, etc. ED aims to resolve the semantic ambiguity of a mention and link it to the correct entry in a given knowledge base (KB), as illustrated in Figure 1.

Generally, ED approaches can be classified into two categories: local model and global model. Local model [5, 26, 38] resolves mention ambiguity by utilizing features such as surface form and local context, while global model (also called collective entity linking) [11, 12, 15, 18] achieves better performance by finding the best alignment of all the mentions in a document to maximize topical coherence. However, collective linking may end up with incorrect entity assignment due to the extremely strong assumption of topical coherence at document level. Consider the example in Figure 1. Global model prefers to link both "England" to a rugby union team which is obviously incorrect for the former one. Moreover, it usually incurs high computational complexity when the document contains a large number of mentions. Therefore, the main trend of current ED approaches [3, 10, 19, 23] is to combine both local and global features, or in other words, consider coherence locally. In fact, [3] claimed that topical coherence only need to hold among neighboring mentions.

1.1 Challenges and contributions

Determining mention adjacency is a non-trivial task. Traditional sequence-based approach [3, 30] measures the distance between mentions by simply counting the number of words in between, which results in semantically inconsistent neighbors sometimes. In this work, we resort to *dependency parse tree* to incorporate the syntactic structure of a sentence for more accurate distance estimation. It is commonly believed that the closer two mentions are on a parse tree, the higher linguistic relatedness they have. However, there are still two issues we need to address carefully. First, it is highly possible for multiple mentions to have the same tree-distance to a target mention, since each word can connect to several other words on a parse tree. This calls for a neighbor selection mechanism to deal with such situation. Considering that sequence-distance evaluates word closeness from a different perspective

Figure 1 An illustration of entity linking. The correct candidate entity for each mention is labeled by red dash arrow

with tree-distance and it causes less distance conflict among words, we propose to combine both distance measures when determining mention neighbors. Furthermore, parse tree is constructed for a single sentence, which makes the cross-sentence tree-distance unmeasurable directly. Therefore, we introduce an algorithm *CoSimTC* to connect the parse trees of adjacent sentences, which derives a whole tree structure for each document, and then extract mention neighbors based on the document-level parse tree.

Moreover, we observe that the topic distribution of a document is still crucial for entity disambiguation, since the correct entities should align with the main topics of that document. Although Latent Dirichlet Allocation (LDA) [2] has achieved proved effectiveness in determining topics of a document, it is not applicable to the ED tasks as the topics are represented as word distributions which cannot be directly encoded into model features. Furthermore, discovering topics through word-by-word analysis of the whole document is quite time-consuming, which would degrade the efficiency of entity linking. Therefore, in this work, we pay attention to the keywords of a document, which reflect the central topic information of the document. We propose a *Sent2Word* keyword extraction algorithm that can detect keywords precisely and efficiently by identifying key sentences from the document and extracting keywords at sentence level.

Another problem that needs to be considered is how to combine local features and global coherence together. Graph-based methods have been successfully applied in this area, such as the Graph Convolutional network (GCN) [22] utilized in [3]. However, GCN is highly dependent on the graph structure, which usually leads to low generalization ability. We believe that Graph Attention Network (GAT) [36] is an ideal alternative to integrate neighbor and keyword coherence into local features. Specifically, based on the mention neighbors extracted from the document parse tree, we utilize *distance decay attention* to encode the mention distance information into deep neural network. In this way, the graph attention layer can successfully extract the discriminative features and explore relatedness between entities to infer the best entity linking result.

The main contributions of our ED method can be summarized as below:

- We derive meaningful local neighbors for each mention in a more linguistic way by utilizing dependency parse tree. We propose a tree connection method CoSimTC to generate a whole tree structure for each document, which can measure cross-sentence distance between mentions. Our distance metric further combines both sequence-distance and tree-distance to reflect mention closeness from different perspectives. The proposed tree-based neighbors can help each mention to make more accurate linking decision.
- We propose an accurate and efficient keyword extraction algorithm Sent2Word, which locate key sentences in the document and then detect keywords at sentence level. The topical coherence with both mention neighbors and keywords is integrated to collectively determine gold entity linkings.
- Our neural ED approach combines basic deep neural network model with Graph Attention Network (GAT), which integrates the discriminative features for each candidate entity on both local and global aspects. The distance decay attention produces better representation for each candidate entity.
- We evaluate the proposed LoG model on six widely-adopted public datasets and compare with state-of-the-art ED models. The experimental results verify the superiority of our proposal in terms of both accuracy and efficiency.

The rest of the paper is organized as follows: We introduce the details of the LoG model in Section 2, report our experimental results in Section 3, review the related literature in Section 4, and finally conclude our work in Section 5.

2 Entity disambiguation model

A typical ED framework consists of three main modules: candidate generation, feature extraction, and neural network model. For each mention m_i, a set of candidate entities denoted as $\phi(m_i)$ are selected from the knowledge base according to the prior probability $p(e_j|m_i)$. Discriminative features for each $e_j \in \phi(m_i)$ are extracted and fed into the neural network model to learn the final probability of linking m_i to e_j.

Identifying representative and discriminative features is crucial to the performance of ED framework. In this work, we combine both local and global features to learn the linking probability of each candidate entity given its context information. Local features reveal how compatible the candidate entity with its local information (e.g., surface form, surrounding words of the mention). Following [3, 27], we conclude these local features for each $e_j \in \phi(m_i)$:

- Lexical match features between m_i's surface form and e_j's entity name, which include the edit distance between them and the Boolean features that measure if they are totally matched or if one is the prefix or suffix of the other and vice versa.
- Context-based features which measure the word overlapping between e_j's entity name and the surrounding words of its corresponding mention. More specifically, it records the lexical matching position between the entity name and the context words.
- Attention-based context information, which reflects the compatibility between e_j and its local context. It is computed as $cos(e_j, c_{m_i,e_j})$, where cos is the cosine similarity. $c_{m_i,e_j} = \sum_{\omega_k \in C(m_i)} \alpha_k * \omega_k$ is the weighted sum of context words, which integrates the valuable information of context words $C(m_i)$ that is the most important to e_j. Here, $\alpha_k \propto cos(\omega_k, e_j)$ is the attention weight that reveals the importance of each context word in supporting e_j.

Our main contributions in this work are the carefully-designed global features to better identify correct linking, i.e., neighbor-based global features (Section 2.1) and keyword-based global features (Section 2.2), as well as the neural network model to integrate all the proposed features (Section 2.3). We will elaborate on each module in the following.

2.1 Neighbor-based global features

Different from the document-level coherence assumption adopted in most collective ED models, we propose that a mention only needs to be topically coherent with mentions that are close, and we call this set of mentions as **mention neighbors**. In our model, part of the global features is based on mention neighbors. Specifically, mention neighbors are formally defined as the set of mentions that have the top-k minimum distances to the current mention: $\mathcal{N}(m_i) = \{m_{i1}, m_{i1}, ..., m_{ik}\}$.

Intuitively, we can treat the number of words between two mentions in the document as the distance between them, which is called *sequence distance*. However, the number of words between mentions sometimes cannot reveal their closeness accurately. Taking the

sentence in Figure 2 as an example, *Beijing* locates closer than *Taiwan Strait* in the raw text with respect to the mention *Lien Chan*, while we can notice that *Taiwan Strait* is more relevant to *Lien Chan* than *Beijing* because *Lien Chan* is the Kuomintang chairman in Taiwan. Therefore, we aim to develop a better distance measure with the aid of parse tree to estimate the syntactic distance between mentions.

Tree distance and tree connection Dependency parse tree is a type of tree-based structure that reveals the syntactic structure of a sentence. The **tree distance** between two tree nodes is defined as the length of the shortest path from one node to the other node within the same parse tree. It is noticeable that the dependency parse tree is generated from a sequence of words rather than mentions. Therefore, to estimate the tree distance of mentions whose surface form includes multiple words (e.g., *Australian National swimming team*), we take the LCA (Lowest Common Ancestor) node to represent this mention (which is *team* in this case). We show an example dependency parse tree structure in Figure 2, which is generated by the Stanford CoreNLP toolkit. According to the definition of the tree distance, *Taiwan Strait* is 3 hops away to *Lien Chan* while *Beijing* is 6 hops away, which accords with the syntactic closeness, as *Taiwan Strait* is more related to *Lien Chan*. Overall, compared with sequence distance, tree distance can generally reveal more accurate semantic relevance between mentions.

Nevertheless, the distance between mentions located in different sentences cannot be directly measured by the parse tree distance we defined above, because the dependency parse tree can only be generated from a single sentence. Hence, an effective **tree connection** method is necessary to deal with this issue. In this work, we introduce five possible solutions to connect consecutive parse trees, which are explained as follows.

1. *Root-based*: Since each parse tree has only one root that serves as the syntactic center of the corresponding sentence, we can have a straightforward tree connection strategy, which connects consecutive parse trees by adding edges between their root nodes, as illustrated with the orange lines shown in Figure 3.
2. *Mention-based*: Considering the influence of sequence distance between mentions, we come up with another heuristic distance measure that adds edges between the adjacent mentions in consecutive sentences, as illustrated in Figure 3, where the adjacent mentions (*University of Sydney* of the former sentence and *Peter* at the latter sentence) are connected by the purple dash line.
3. *Coreference-based*: Co-reference is one of the most effective features of semantic relation between textual expressions. The definition of co-reference is: the two expressions

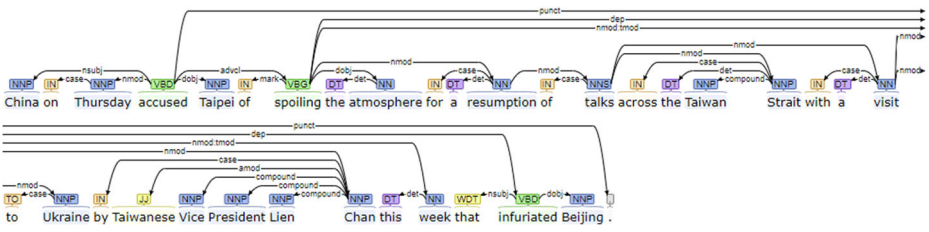


Figure 2 The dependency parse tree of a sample sentence

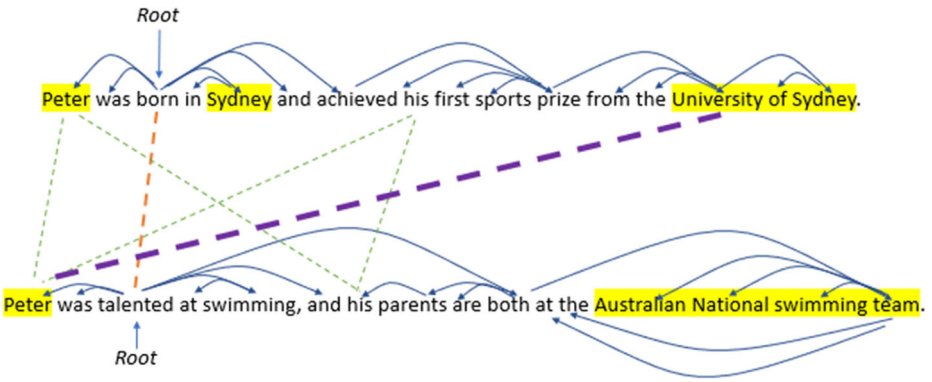


Figure 3 An illustration of multiple parse trees for two sentences where highlighted words are mentions and blue arcs are dependency edges. Green lines represent cross-sentence co-reference connection, orange dash lines represent root connection, and purple dash lines represent the adjacent cross-sentence mention connection

that are claimed to stand for the identical real-world entity in a given context are treated as in co-reference relation. For instance, the expression "Peter" and "his" in the sentences shown in Figure 3 both refer to the same person. Hence, we can connect these two co-reference expressions to encode their correlation, as shown with the green lines. One thing that needs to mention is that, since we aim to connect consecutive parse trees, we only care about the co-reference expressions lying at adjacent sentences, while ignoring the co-reference relations within the same sentence. By doing this, we can successfully push semantically relevant mentions closer even though they are in different sentences.

4. *Similarity-based*: Sometimes consecutive sentences may not have co-reference expressions. Therefore, we need to design another algorithm to measure the relatedness for cross-sentence mentions. Based on the intuition that similar mentions should be pushed closer, we add edges between the pair of mentions located in adjacent sentences that have the highest pairwise semantic similarity scores. In this work, the semantic similarity score involves the similarity between their surface forms and context words:

$$Sim(m_i, m_j) = \theta * cos(m_i, m_j) + \frac{1-\theta}{2} * ctx\_sim(m_i, m_j) \tag{1}$$

where $ctx\_sim(m_i, m_j)$ is related to mentions' left context embeddings ($l_{ctx}(m)$) and right context embeddings ($r_{ctx}(m)$). The embeddings of mention and context are the average of the word vectors contained in surface form and context words respectively. The context similarity is:

$$ctx\_sim(m_i, m_j) = cos(l_{ctx}(m_i), l_{ctx}(m_j)) + cos(r_{ctx}(m_i), r_{ctx}(m_j)) \tag{2}$$

Algorithm 1 CoSimTC.

Input: a set of parse trees $T = (t_1, t_2, \ldots, t_n)$, and their corresponding sentences $S = (s_1, s_2, \ldots, s_n)$

Output: the whole Document Tree DT with all trees connected

```
1  for each adjacent sentence pair (s_i, s_{i+1}) do
2      coref_{s_i,s_{i+1}} ← cross_coref(s_i, s_{i+1}) ;
3      if coref_{s_i,s_{i+1}} ≠ ∅ then
4          DT.add_edge(coref_{s_i,s_{i+1}})
5      else
6          sim_{s_i,s_{i+1}} ← cross_sim(s_i, s_{i+1});
7          if sim_{s_i,s_{i+1}} ≠ ∅ then
8              DT.add_edge(sim_{s_i,s_{i+1}})
9          else
10             adj_{s_i,s_{i+1}} =cross_adj(s_i, s_{i+1}) ;
11             DT.add_edge(adj_{s_i,s_{i+1}})
12         end
13     end
14 end
```

5. *CoSimTC*: Based on the previous four types of methods, we propose our final tree connection algorithm **CoSimTC** (Coreference-Similarity Tree Connection) on the basis of an integration of coreference-based and similarity-based approaches. As illustrated in Algorithm 1, the general procedure of our algorithm runs as follows: We first connect together each pair of co-reference expressions lying in adjacent sentences (lines 2–4). For those sentences without co-reference relations, we add edges between mentions lying in adjacent sentences and with pairwise highest similarity scores defined in (1) (lines 6–8). Finally, we connect adjacent words for sentences that have no mentions (lines 10–11). After executing the above operations, all consecutive sentences have been connected and a complete document tree has been generated.

After producing the whole document tree DT for a series of continuous sentences, we compute the tree distance between each pair of mentions based on DT and choose the top-k mentions that have the smallest tree distance as the set of neighbors that are most coherent to the current mention.

Neighbor-based global features Following [3], we extract two global features for each candidate entity of the mention $e_j^i \in \phi(m_i)$ given mention neighbors $\mathcal{N}(m_i)$ based on the assumption that the mention should be topically coherent with any neighboring mention $m \in \mathcal{N}(m_i)$.

Intuitively, the gold entity of the target mention should be coherent to all its neighboring mentions, and the closer the neighboring mention locates to the mention, the higher importance it has to disambiguate the target mention. Hence, we propose a **distance-decay neighbor mention similarity** and apply it to measure the relatedness between the candidate entity and the surface form of each mention neighbor:

$$\{cos(e_j^i, m_j) * p^r | m_j \in \mathcal{N}(m_i)\} \tag{3}$$

where p is the distance decay factor and $r \in [0, k-1]$ is the ranking of the corresponding neighbor. For each candidate entity, we concatenate its local features, entity embedding and the distance-decay neighbor mention similarity to form the feature vector $F \in \mathbb{R}^{d_0}$, which is the initial representation of the candidate entity node.

Finally, global features are integrated for each candidate entity by building its subgraph on the basis of neighbor mentions. The subgraph is defined as $G = (E, R)$, where $E = \{e_1, e_2, ..., e_n\}$ is the set of entities which represent the graph vertices, and $R = \{r_j^i | r_j^i = Rel(e_i, e_j)\}$ which serve as edges in the graph and reflect the relevance between connected entities. According to the assumption that all mentions should be coherent to their mention neighbors, their gold entity should also be coherent with each other. To figure out the optimal subset of candidate entities for each mention, we connect each entity with the candidate entities of its neighboring mentions. In other words, the nodes connected to $e_j^i \in \phi(m_i)$ can be represented as:

$$\mathcal{N}(e_j^i) = \{e_k^l | e_k^l \in \phi(m_k), m_k \in \mathcal{N}(m_i)\} \tag{4}$$

2.2 Keyword-based global features

With respect to document-level topical coherence, we believe the gold entity should not only be coherent with its mention neighbors, but also align with the central topics of the whole document. Therefore, we further extend our ED model by identifying topics of a document and integrating compatibility features between candidate entities and the extracted topics.

Keyword extraction Latent Dirichlet Allocation (LDA) [2] is a commonly-used unsupervised topic model, which can accurately recognize topics of a document. However, LDA is a statistical model which requires a large number of documents to learn meaningful representations of topics. Moreover, it is not applicable to the ED tasks as the topics are represented as word distributions which cannot be directly encoded into model features. In this work, we pay attention to the keywords which reflect the central topic information of a document.

Naturally, we can adopt an existing keyword extraction method, e.g., CoreRank [35] which is a well-performed unsupervised word importance scoring algorithm and has been successfully applied in various applications such as document summarization. CoreRank is a graph-based algorithm to detect keywords that have influence across the whole document. It first constructs a directed weighted graph from the document, where vertices represent words and edges are weighted based on co-occurrence frequency of words within a sliding window of a fixed size. It then computes the score of each node as a summation of the core numbers of its neighbors, where the core number of a node is the highest order of the k-core subgraph that contains the node. A k-core of graph G is a maximal subgraph of G in which every vertex v has at least degree k [34] and the degree of v is the sum of weights of its incident edges. Finally, it takes the top-k words as the keywords of the document. The aim of the algorithm is to find the most influential words within the document, where a word is considered to be influential if it is located at the core of the network. A general framework of CoreRank is illustrated in Figure 4, where nodes 'x' and 'xx' have the same core number 2, but 'x' node has CoreRank score $3 + 2 + 2 = 7$ which is higher than that of node 'xx' $(2+2+1 = 5)$. Hence, the algorithm regards node 'x' as more influential than node 'xx', or in other words, node 'x' is located closer to the core of the graph compared to node 'xx'. In

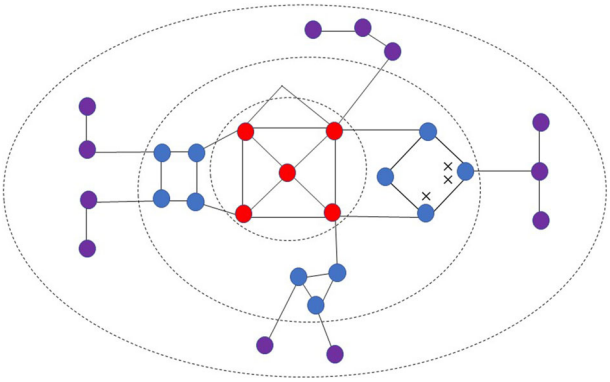


Figure 4 Illustration of CoreRank, where all red nodes have core number $c = 3$, blue nodes $c = 2$ and purple nodes $c = 1$

our entity linking task, we can utilize these keywords with high CoreRank scores to ensure the compatibility between candidate entities and the most influential topics of the document.

Sent2Word: from key sentences to keywords Unfortunately, we observe that directly extracting keywords from the whole document using CoreRank has several limitations. On one hand, CoreRank requires the most influential keywords to have a high co-occurrence frequency with the remaining words in the whole document, which is obviously too strict. Moreover, the raw document may contain noisy content that is likely to mislead the unsupervised model when capturing crucial topical information. On the other hand, CoreRank discovers topics by building a weighted graph among all the words in a document and traversing the graph to calculate word scores. Such a word-by-word analysis of the whole document is quite time-consuming, which would degrade the efficiency of keyword extraction and entity linking. Therefore, we propose the **Sent2Word** algorithm that first locates the key sentences in a document and then applies CoreRank only on the key sentences to extract keywords.

Obviously, the most important question is how to rank sentences of a document so as to identify key sentences. Our basic idea is that a sentence is more important if it has a higher similarity with the other sentences of the document. To this end, we construct a weighted sentence-graph from the given document, where nodes represent sentences and edge weights measure the similarity between sentences, and then conduct a PageRank [29] on the sentence-graph to obtain the final scores of all the sentences denoted as $Score_{pagerank}(s)$. In order to calculate sentence similarity, we convert each sentence s into a bag-of-word representation with the words weighted by term frequency-inverted document frequency (tf-idf) metric and measure the similarity between sentences s_i and s_j as the cosine of the corresponding weighted word vectors. A straightforward way of applying tf-idf is to calculate the sentence-level tf and idf values. In other words, it assumes that a word is very important for expressing a sentence if it occurs frequently in the target sentence while rarely in the other sentences. However, this assumption contradicts with our intuition for extracting keywords from a document, since the topic words should still appear frequently in the document. Hence, in order to capture the most influential words in a sentence and meanwhile eliminate the misleading information introduced by unimportant words that occur commonly in all the documents, we adopt sentence-level tf and document-level idf

values for word weighting. In particular, tf measures the frequency of a word within the sentence:

$$TF(w_i, s_j) = \frac{Count(w_i, s_j)}{\sum_{w_k \in s_j} Count(w_k, s_j)}, w_i \in s_j, s_j \in d, d \in \mathcal{D} \tag{5}$$

where d is the target document and $\mathcal{D}$ is the whole document dataset. Idf is computed at document level:

$$IDF(w_i) = \log \frac{|\mathcal{D}|}{1 + |\mathcal{D}_{w_i}|}, w_i \in d, d \in \mathcal{D} \tag{6}$$

where $|\mathcal{D}_{w_i}|$ represents the number of documents containing w_i, and $|\mathcal{D}|$ is the document dataset size.

The above PageRank-based algorithm might not perform well when the document contains only a limited number of sentences. To remedy the lack of sentences for some short documents, we introduce another two sentence features, position information and length information, which are quite effective for measuring the importance of sentences. We observe that people tend to express their main topics at certain positions when writing a document. Based on our statistical analysis of the sentence distribution on a sample of public datasets, the possibility of key sentences appearing at the beginning or the end of the document is usually much higher. Hence, we assign different weight to each sentence according to its relative position in a document and the observed distribution, reflecting the probability of choosing it as a key sentence, denoted as $Score_{pos}(s)$. Furthermore, to eliminate the score dominance caused by some really long sentences, we adopt the length score which reveals the difference between the target sentence length and the ideal key sentence length (e.g., the average sentence length of a document):

$$Score_{len}(s) = \frac{length_{avg} - |length_{avg} - length_s|}{length_{avg}} \tag{7}$$

The final score of each sentence s is defined as a weighted linear combination of all the three aforementioned scores, where the parameters α, β, and γ represent the importance of each corresponding score with $\alpha + \beta + \gamma = 1$:

$$Score(s) = \alpha * Score_{pagerank}(s) + \beta * Score_{pos}(s) + \gamma * Score_{len}(s) \tag{8}$$

After calculating the final score of each sentence, we select sentences with top-l overall scores as the key sentences and then apply CoreRank algorithm to score each word within the key sentences. The highest ranked k words are regarded as the keywords of the target document, denoted as $\mathcal{W}(d)$.

Add keywords to feature vector and graph vector After extracting keywords for each document d, denoted as $\mathcal{W}(d) = \{w_1, \ldots, w_k\}$, we define a coherence set, which includes both the neighbor mentions and the keywords: $\mathcal{C}(m_i) = \mathcal{N}(m_i) + \mathcal{W}(d)$ where $m_i \in d$. We believe the gold entity of a mention should be coherent with the whole coherence set that covers the locally-global information (i.e., neighbor mentions) and the global topic information (i.e., keywords) at the same time. Hence, in addition to the neighbor mention

similarity defined in (3), we also concatenate the similarity between the candidate entity and each keyword into the feature vector F:

$$\{cos(e^i_j, w_j)|w_j \in \mathcal{W}(d_i)\} \tag{9}$$

Similarly, we need to add keywords to each sub-graph vector G as well, since we intend to find the best entity assignment within a set of mentions close to each other and meanwhile guarantee the coherence between the entity assignment and the topical keywords across the whole document. In particular, the updated subgraph vector of each candidate entity e_j for mention m_i (unlike (4)) contains both the candidate entities of the neighbor mentions and the extracted keywords of the document as vertices, and the edges connect the candidate entity to every node belonging to its neighbor node set $e \in \mathcal{N}(e^i_j)$ which is defined as below:

$$\{\mathcal{N}(e^i_j) = n^l_k | n^l_k \in \phi(m_k) \cup \mathcal{W}(d), m_k \in \mathcal{N}(m_i)\} \tag{10}$$

2.3 Neural network model

The overall ED framework is illustrated in Figure 5. We extract mention neighbors for each mention based on the document tree built by our CoSimTC measure and extract keywords for each document. The mention neighbors and the keywords together consist of the coherence set, based on which the entity subgraph is constructed as global features. We also extract a series of local discriminative features, concatenated the compatibility between entity and the coherence set to generate the feature vector for each candidate entity. We then implement and extend the Graph Attention Network (GAT) [36] to explore the coherence between entities, which integrates both local information and global structure information for entity disambiguation. The key difference with the original GAT structure

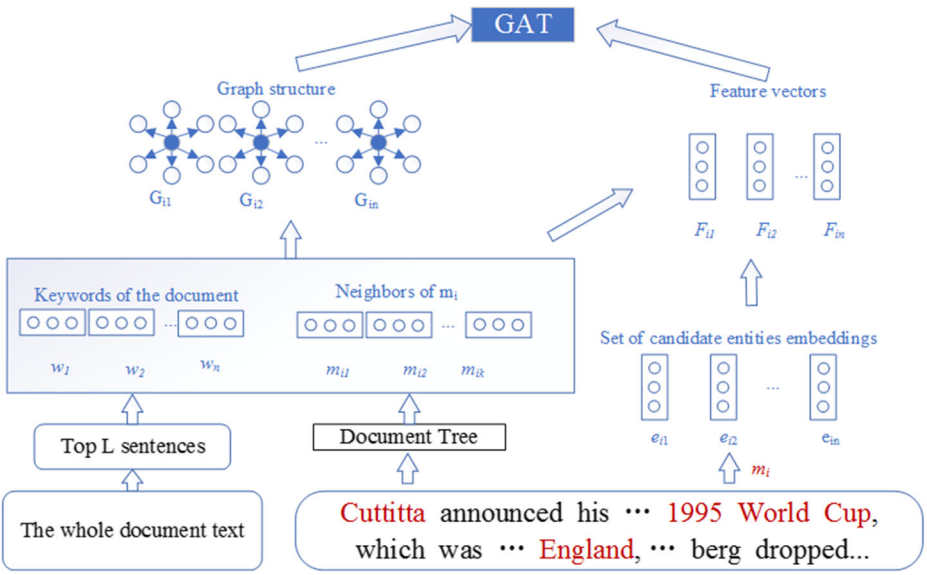


Figure 5 Framework overview

in [36] is that our input graph is the subgraph structure built on top of the mention neighbors and topical keywords, instead of the whole graph containing all mentions within the document.

The objective of the neural network model is to find the best linking candidate for each mention, which satisfies that:

$$\Gamma(m) = \arg\max_{e_i \in \phi(m)} p(e_i|m) \tag{11}$$

where $p(e_i|m)$ is the probability that m refers to candidate e_i, and $p(e_i|m) \propto exp(score(m, e_i))$. $score(m, e_i)$ is the score function which is learned by our neural network model. We introduce the detailed computation process below.

Encoding features In our model, each entity serves as a node and the relations between two entity serve as edges. For any $e_i \in \phi(m)$, we have feature vector $F_i \in \mathbb{R}^{d_0}$, which is the initial representation of the entity. Firstly, we will encode the feature vector using a perception (MLP): $h_i^1 = g(F_i)$, where h_i^1 is the hidden state of entity e_i, g is a two-layer MLP.

Entity relation scores Then, we use graph attention layer to compute the relation score between e_i and each of $n_j \in \mathcal{N}(e_i)$. Different from conventional GAT, we use multiplicative attention to compute the relation between two entities:

$$Rel(e_i, n_j) = p^r \left(\frac{\mathbf{e}_i^T \mathbf{R} \mathbf{n}_j}{|\mathbf{e}_i||\mathbf{n}_j|} \right) \tag{12}$$

where $\mathbf{e}_i, \mathbf{n}_j \in \mathbb{R}^{d_e}$ is the embedding of the two entities (n_j is encoded as either word embedding or entity embedding depending on whether it is a word or an entity), $R \in R^{d_e \times d_e}$ is the diagonal matrix that represents the relation between the two entity nodes. Here, we also add weight decay p^r which is the same value as defined in Section 2.1. Note that if n_j is a word embedding, we ignore the weight decay for it. This relation score contains the relatedness between e_i and n_j in KB. It also reflects the closeness in text, because it weakens the weight for candidates that have long distance between their corresponding mentions. After calculating $Rel(e_i, n_j)$for each $n_j \in \mathcal{N}(e_i)$, we normalize them to guarantee that all the $Rel(e_i, n_j)$ of neighbor nodes sum to one. Then we can derive the new representation of e_i as:

$$h_i' = \sigma \left(\sum_{e_j \in N(e_i)} Rel(e_i, n_j) W h_j \right) \tag{13}$$

Decoding After above computations, the hidden state of e_i now contains the features of itself, integrating with the features of all $e_j \in \mathcal{N}(e_i)$. Finally, we map the hidden state h_i^t to the confidence score of choosing e_i:

$$score = W^t h_i^t + b^t \tag{14}$$

Training objective The training objective of our neural network model is to minimize the cross-entropy loss. The objective can be described as:

$$\mathcal{L}_m = -\sum_{j=1}^{n} P(e_g|m) log(P(e_j|m)) \tag{15}$$

where e_g is the ground truth candidate entity. Suppose that we have D mentions per document and there are totally $\mathcal{D}$ documents in our training dataset. The overall objective function is the loss of all mentions within the training corpus:

$$\mathcal{L} = \sum_{D \in \mathcal{D}} \sum_{m \in D} \mathcal{L}_m \tag{16}$$

3 Experiments

In this experiment, we evaluate the overall performance of our LoG model for entity disambiguation, as well as the corresponding effect of different strategies for extracting neighbor-based global features (introduced in Section 2.1) and keyword-based global features (described in Section 2.2).

3.1 Experimental setup

Datasets Following previous work, We verify the performance of our ED model on six widely-used publicly-available datasets. For in-domain scenario, we utilize AIDA-CoNLL dataset [18] which includes AIDA-train for training, AIDA-A for validation and AIDA-B for testing. For out-domain scenario, we still train and validate on the AIDA dataset and test on other five popular datasets: MSNBC, AUIAINT, and ACE2004 [33] are released and cleaned by [13]; WNED-CWEB [9] and WNED-WIKI [13] are two datasets with larger size but less reliability, which are automatically extracted from ClueWeb and Wikipedia respectively. We report the statistics of these datasets in Table 1.

Compared methods We compare our LoG model with the following existing state-of-the-art entity linking systems: Hoffart et al. [18] and Guo and Barbosa [13] are both effective graph-based ED approaches; Ganea and Hofmann [10] is a novel global ED model based on the Loopy Belief Propogation (LBP); Phong and Ivan [23] considers correlations between entities for collective entity linking. Fang and Cao [7] and Yang and Gu [39] both link entities in sequential order and utilize reinforcement learning to boost performance globally; Chen and Wang [4] incorporates latent entity type information to achieve further improvement of the linking accuracy. For a fair comparison of these models, we report the best performance achieved on the same datasets from their original papers. We only consider in-KB mentions and micro-F1 measure in this experiment, where micro-F1 evaluates the

Table 1 Statistics of datasets used in the experiments

Dataset	Total mentions	Total docs	Mentions/doc	Gold recall
AIDA-train	18448	946	19.5	–
AIDA-A	4791	216	22.1	96.9%
AIDA-B	4485	231	19.4	98.2%
MSNBC	656	20	32.8	98.5%
AQUIAINT	727	50	14.5	94.2%
ACE2004	257	36	7.1	90.6%
WNED-CWEB	11154	320	34.8	91.1%
WNED-WIKI	6821	320	21.3	92%

average linking accuracy per mention. All the reported performance are based on Wikipedia and YAGO mention-entity index. We run the models five times and report the average performance. We also show the standard deviation in the ablation study for significance test.

Hyper-parameter settings To construct the document tree, we utilize the commonly-used dependency parsing toolkit from Stanford CoreNLP in this experiment. The dimensions of the word and entity embeddings are both $d = 300$. Word embeddings are from glove [31] and entity embeddings are from [23]. We choose the top-6 mentions as the mention neighbors, and the distance decay for mention neighbors is 0.95. As for similarity-based CoSimTC measure, the parameter θ used in (1) is 0.7. We fetch surrounding words of each mention at window size of 50 as context words $c_{e,m}$, and window size of 3 for (1).

With respect to the related parameters of keyword extraction , we set number of key sentences $l = 3$, and the number of keywords $k = 2$. The parameter setting for (8) is $\alpha = 0.8$, $\beta = 0.1$, $\gamma = 0.1$ respectively. These parameters are determined by grid search. Before building the graph-of-words and feeding it to the CoreRank algorithm, we pre-process the text by the following steps: 1) Remove stopwords. 2) Use Part-of-Speech tagging to keep only nouns and adjectives. The POS tool is from Stanford CoreNLP. 3) Stem each word using WordNetLemmatizer in nltk package.

The hyper-parameter settings and training procedure of our GAT model mostly follow [3] and [10]. We use early-stop in the training stage, and make each document as a mini-batch. We set a 2-layer MLP encoder, which has 2000 and 1 hidden units respectively. We set the number of graph attention layer as 2 to capture richer global entity coherence. The $diag(\mathbf{R})$ is sampled from $\mathcal{N}(1, 0.1)$.

Candidate generation and selection The candidate generation and selection step setting is as follows: we pick top-30 entities for each mention with highest $\hat{p}(e|m)$. Then, we preserve the candidate entities based on two standards: (1) top-4 entities with highest $\hat{p}(e|m)$; (2) top-4 entities that have the largest context-entity similarity scores. The similarity is measured as the cosine similarity between entity and context words: $\mathbf{e}^T(\sum_{\omega \in W} \mathbf{w})$, where $\mathbf{e}, \mathbf{w} \in \mathbb{R}^d$ are entity and word embeddings respectively, and W is the surrounding words of m_i inside the 50-word window.

3.2 Overall performance

The overall performance of entity disambiguation on the six public datasets is shown in Table 2. We report the in-domain accuracy, the cross-domain accuracy, and the average performance over all cross-domain datasets respectively in the table. It can be shown that our LoG model, which combines both mention neighbors and keywords information, has a promising generalization ability in the out-domain scenario and achieves superior performance compared with most of the existing state-of-the-art methods. Considering the overall performance on the cross-domain datasets, LoG model is comparable with Fand and Cao [7] and achieves the highest F1 score on average. In particular, it has the best performance with respect to F1 score on most datasets including MSNBC, AQUIAINT, ACE2004 and ClueWeb datasets. Although other models have higher accuracy on the WIKI dataset, their performance on the other cross-domain datasets is still incomparable with our LoG model, thus leading to lower average F1 score. Moreover, it is worth noting that these models to some extent incorporate the Wikipedia corpus into their feature representation [23, 39] or training process [4, 7, 13]. For example, Yang and Gu [39] utilizes Wikipedia's hyperlink structure to incorporate entities that are closely associated with the target entity into feature

Table 2 F1 scores over all six public datasets, where the bold and underlined scores are the best and second best accuracy achieved on corresponding dataset, respectively

Methods	AIDA-B	MSNBC	AQUIAINT	ACE2004	ClueWEB	WIKI	Avg
Hoffart et al	81.82	79	56	80	58.6	63	69.74
Guo and Barbosa	89	92	87	88	77	**84.5**	86.25
Ganea and Hofmann	92.22	93.7	88.5	88.5	77.9	77.5	86.39
Phong and Ivan	93.07	93.9	88.3	89.9	77.5	78.0	86.77
Fang and Cao	94.3	92.8	87.5	91.2	**78.5**	82.8	**87.85**
Yang and Gu	**94.64**	**94.57**	87.38	89.44	73.47	78.16	86.27
Chen and Wang	93.54	93.4	89.8	88.9	77.9	80.1	87.27
LoG-Sent2Word	91.74	94.16	**90.90**	**92.92**	76.96	75.02	86.95
LoG	92.02	94.32	90.75	**92.92**	77.91	76.24	87.37

representation; Fang and Cao [7] filters candidate entities based on Wikipedia and exploits additional training data from the Wikipedia corpus; With the help of powerful BERT model, Chen and Wang [4] also achieves good performance on the WIKI dataset by pre-training the BERT model on Wikipedia articles. However, such a setting limits their generalization ability to the other cross-domain datasets, resulting in inferior linking performance. Comparing LoG and LoG-Sent2Word [37] (the LoG model that only considers mention neighbors while ignoring topical keywords), we can observe higher accuracy achieved by the final LoG model, which verifies the importance of integrating topical keywords for entity disambiguation.

For the in-domain evaluation, our LoG model is slightly outperformed by several state-of-the-art solutions [4, 7, 39]. These models integrate all mentions within a document to capture topical coherence, while our idea of mention neighbors is still effective as a trade-off of document-level coherence. When a large amount of mentions are incorporated, more information could be included in the model but it is also more likely to introduce misleading information. Besides, limiting topical coherence among mention neighbors can effectively prune the training space and hence achieve higher model efficiency and feasibility in practice. Furthermore, these state-of-the-art models all incorporate auxiliary information in their training and testing process. Specifically, Chen and Wang [4] utilizes the latent type information via BERT model, which is quite useful in the AIDA-CONLL dataset. Fang and Gu [7] achieves a high-quality candidate entity set with the help of Wikipedia and improves the feature representation by incorporating additional training data from Wikipedia. Although Yang and Gu [39] achieves the best performance on the in-domain AIDA-CONLL dataset, it has poorer generalization ability and performs not well on most of the cross-domain datasets especially AQUIAINT, ACE2004 and ClueWeb. Overall, our LoG model achieves comparable or even superior performance on the public datasets. Nevertheless, this experiment also inspires us to leverage extra knowledge for entity disambiguation (e.g., type information, Wikipedia articles and hyperlink structure) in our future work.

3.3 Component analysis

Effect of global features In this part, we illustrate the effectiveness of adding global information from neighbor nodes and keywords via our GAT-based network model. The local results are generated by only implementing the MLP encoder described in Section 2.3 which

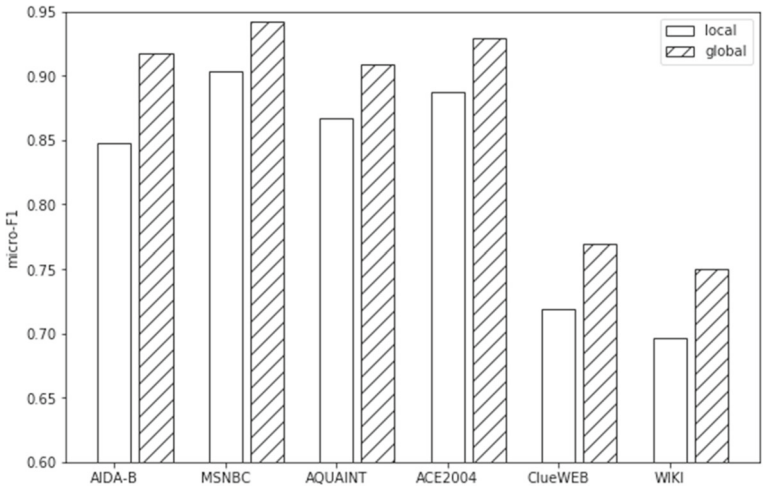


Figure 6 Effect of using GAT to integrate global information

encodes each entity as node based on its feature vector, and the global model represents for the LoG model. Figure 6 demonstrates the effectiveness of our GAT neural network model, which increases the accuracy on each dataset by at least 4% F1 score. It is noticeable that the accuracy increase of our GAT model is smaller on the first three cross-domain datasets: MSNBC, AQUIAINT and ACE2004. The reason is that the entity disambiguation cases are easier, 85% of which can be solved by just utilizing the local features. Whereas the AIDA-B, ClueWeb and WIKI datasets are much more challenging because they include noise and more ambiguous cases that our encoding features within feature vector cannot handle. Therefore, the influence of the global information integrated by our GAT-based model becomes more important for ClueWeb and WIKI datasets.

Comparison of distance measures Table 3 shows the performance difference of the four kinds of distance measures in our neural network model. Except seq-dist, the latter three distance methods all measure the distance between mentions within the same sentence by dependency parse tree. We exclude the keywords information from our LoG model in this experiment to have a fair comparison. We can conclude from Table 3 that the CoSimTC distance measure significantly beats the other three distance measures and achieves the highest accuracy across all these six datasets, since it integrates not only the co-reference information but also the compatibility between mention and context words in adjacent sentences. It can also be observed that mention-based measure has satisfactory results, even if it obtains a slightly worse performance than CoSimTC. The reason is that the mention-based method

Table 3 Comparison of seq-dist, root-based, mention-based and CoSimTC measures

Methods	AIDA-B	MSNBC	AQUIAINT	ACE2004	ClueWEB	WIKI
seq-dist	91.45 ± 0.18	93.96 ± 0.13	89.79 ± 0.53	87.32 ± 0.7	75.07 ± 0.13	73.60 ± 0.14
root	91.33 ± 0.21	93.80 ± 0.15	89.51 ± 0.42	86.92 ± 0.82	74.98 ± 0.13	74.15 ± 0.12
mention	91.62 ± 0.14	93.88 ± 0.15	89.65 ± 0.41	90.83 ± 0.44	76.34 ± 0.13	74.2 ± 0.11
CoSimTC	91.74 ± 0.20	94.16 ± 0.16	90.90 ± 0.63	92.92 ± 0.4	76.96 ± 0.13	75.02 ± 0.24

also measures inter-sentence mention distance at sentence level. Root-based distance has a poor performance, and even worse than seq-dist sometimes. This is because such distance is only able to figure out intra-sentence mention closeness, while simply adding edges to the root of each sentence may involve mistakes when measuring cross-sentence mention distance in some cases. For instance, the words that are closer to the sentence root are likely to have a smaller distance between each other, even if their sequence distance is large.

It can also be observed that the four types of distance measures have the largest difference of performance on the ACE2004 dataset. The reason is that this dataset is smaller, thus the changes by even a few cases can contribute to evident differences in the overall accuracy of the whole dataset. Another reason is that this dataset is cleaner, hence the benefit of methods based on parse tree can be expressed totally without the influence of noise. However, for AIDA-B dataset, the four distance measures have a negligible difference. This is because this dataset has few multi-mention sentences, which means most sentences have a smaller number of mentions, and even using seq-dist is enough to reflect the closeness of these mentions. Furthermore, we cannot observe obvious benefit for parse tree based distance measures for ClueWeb and WIKI datasets, because the noise contained in these two datasets has a bad effect on producing parse tree for each sentence. For example, they may contain some invalid sentences that have incorrect linguistic structure or short fragment sentences deriving from breaking long sentences, such as messy symbols, website addresses, news titles, which cannot be parsed into valid parse trees. In addition, the co-reference tool we used may produce some incorrect co-references across sentences, which may bring bad influence to the benefit of the co-reference relations when connecting adjacent sentences.

Comparison of keyword extraction methods Table 4 illustrates the result comparison of three keyword extraction algorithms described in Section 2.2, i.e., LDA, CoreRank, and Sent2Word. Overall, it can be concluded that adding keywords information is useful for entity disambiguation since the performance on the two large-scale and challenging datasets, ClubWEB and WIKI, is significantly improved for all the three keyword extraction algorithms compared with the LoG model which only considers mention neighbors using CoSimTC. Keyword information also has effect on the in-domain AIDA-B dataset because it can capture the overall topic when linking entities. The improvement in MSNBC, AQUIAINT and ACE2004 is marginal since these three datasets are relatively easy to process and they can already achieve good performance based on local features and mention neighbors. Therefore, adding keyword information does not bring too much improvement in these cases.

LDA is a commonly-used topic word extraction method. We apply the LDA model in the gensim package of python and select the top-2 words of the identified topics as the extracted keywords. As we can see from Table 4, LDA achieves a slight performance improvement over CoSimTC on the ClueWEB and WIKI datasets as it can capture the important

Table 4 Comparison of keyword extraction methods

Methods	AIDA-B	MSNBC	AQUIAINT	ACE2004	ClueWEB	WIKI
CoSimTC	91.74 ± 0.2	94.16 ± 0.2	$\mathbf{90.90 \pm 0.63}$	$\mathbf{92.92 \pm 0.4}$	76.96 ± 0.13	75.02 ± 0.24
+LDA	91.77 ± 0.27	92.93 ± 0.35	88.48 ± 0.83	90 ± 0.8	77.31 ± 0.17	75.51 ± 0.33
+CoreRank	91.93 ± 0.16	92.78 ± 0.2	88.76 ± 0.52	90.8 ± 0.7	77.49 ± 0.14	75.49 ± 0.17
+Sent2Word	92.02 ± 0.14	$\mathbf{94.32 \pm 0.16}$	90.75 ± 0.41	$\mathbf{92.92 \pm 0.3}$	$\mathbf{77.91 \pm 0.12}$	$\mathbf{76.24 \pm 0.11}$

topical information to some extent which helps to promote entity disambiguation. However, the accuracy is much worse on the other three relatively easy datasets. This is mainly because the inaccurately extracted keywords information could mislead the disambiguation model and end up with an incoherent entity assignment. We also validate the effect of Sent2Word in Table 4. The difference between the last two rows is that CoreRank extracts keywords directly from the whole document while Sent2Word locates key sentences and extract topical keywords from the key sentences only. It can be shown that CoreRank is also a powerful topic extraction method as it achieves comparable performance compared to LDA. Our Sent2Word algorithm obviously achieves the best results compared with the other two keyword extraction methods. According to our observation, some of the keywords identified by CoreRank directly on the whole document actually do not occur in the topical sentences. Consequently, Sent2Word can achieve much higher accuracy than CoreRank by filtering noisy non-topical sentences, locating the most important part of a document, and restricting keyword extraction only on the key sentences, let alone that Sent2Word can naturally improve the efficiency of extracting keywords by avoiding the construction and traversal of a large word-graph of the whole document in CoreRank.

Recall that we introduce two sentence features, i.e., position and length information, and combine them with the PageRank-based algorithm for selecting topical keywords from a document. In (8), we use three parameters, α, β and γ respectively, to determine the importance of each corresponding component. In this part, we conduct an ablation study to evaluate the effectiveness of adding these sentence features. Table 5 reports the experimental results, where $\beta = 0$ and $\gamma = 0$ represent removing position and length information respectively from our LoG model. We observe that both sentence position and length information can boost the overall performance of entity disambiguation on all the six datasets, and the accuracy can be further enhanced when both features are combined together. It can also been shown that sentence position is more effective than the length information in ranking topical keywords, as a larger performance degradation is observed for $LoG(\beta = 0)$ compared with $LoG(\gamma = 0)$. Without position information ($LoG(\beta = 0)$ vs. LoG), all the results suffer a significant drop especially on the AQUIAINT, ACE2004 and WIKI datasets ($\approx 1\%$). This is because most of these datasets obey the key sentence distribution observed in our statistical analysis. In the ClubWeb dataset, length information is slightly more important than the sentence position (a larger drop of accuracy for $LoG(\gamma = 0)$ than $LoG(\beta = 0)$), since ClubWeb contains many long and noisy sentences while the length score can help to eliminate such noise in the process of keyword selection using the proposed Sent2Word algorithm.

Qualitative illustration of CoSimTC We illustrate the effect of utilizing CoSimTC to generate mention neighbors compared to utilizing seq-dist qualitatively, which is shown in Figure 7. Our target mention is shown as the red word in this figure. We have also

Table 5 Effect of adding sentence position and length information

Methods	AIDA-B	MSNBC	AQUIAINT	ACE2004	ClueWEB	WIKI
LoG ($\beta, \gamma = 0$)	91.81	93.76	89.84	91.80	77.51	75.52
LoG ($\beta = 0$)	91.93	93.79	89.88	92.05	77.83	75.50
LoG ($\gamma = 0$)	91.89	94.08	90.53	92.50	77.56	75.78
LoG	92.02	94.32	90.75	92.92	77.91	76.24

1. 'West Indian all-rounder Phil Simmons took four for 38 on Friday as Leicestershire beat Somerset by an innings and 39 runs in two days to take over at the head of the county championship.'
2. 'Their stay on top, though, may be short-lived as title rivals Essex, Derbyshire and Surrey all closed in on victory while Kent made up for lost time in their rain-affected match against Nottinghamshire.'
3. 'After bowling Somerset out for 83 on the opening morning at Grace Road, Leicestershire extended their first innings by 94 runs before being bowled out for 296 with England discard Andy Caddick taking three for 83.'

Mention neighbors(CoSimTC): Essex, Nottinghamshire, Surrey, Derbyshire Somerset, Somerset		Mention neighbors(seq-dist): Surrey, Derbyshire, Essex Nottinghamshire, Somerset, Grace Road	
'Kent'	0.09%	'Kent'	0.09%
'Kent_County_Cricket_Club'	99.46%	'Kent_County_Cricket_Club'	97.68%
'Kent_(band)'	0.03%	'Kent_(band)'	0.03%

Figure 7 A demonstration of using parse tree to produce mention neighbors

highlighted the mention neighbors derived by these two distance measures in other colors, where the green words represent the common mention neighbors generated by both distance measures. We can notice that the mention neighbors produced by these two distance methods have one different mention neighbor. From the raw text, we can see that *Somerset* contributes more to choosing the correct candidate entity, as it is highly relevant to the cricket mentioned in the context of the *Kent*, while *Grace Road* is not as relevant to it and contributes little to the linking decision. We can verify the effectiveness of our algorithm design by the output that represents the probability of linking to each candidate. It can be seen that even though these two distance measures can successfully link to the gold entity because of the effective local context information, our CoSimTC method still achieves higher possibility to link to the correct candidate by integrating more accurate global information from the generated mention neighbors.

Qualitative illustration of adding keywords We also make a qualitative illustration of the effect of adding keywords shown in Figure 8, where the red word is the mention and green words are the top-2 keywords of this document extracted from key sentences 1, 3 and 4. We can see the keywords information successfully helps the model pick the gold entity ('Electoral_division_of_Murchison'), while simply using CoSimTC method without the keywords information is not sufficient to link this mention to the correct candidate entity.

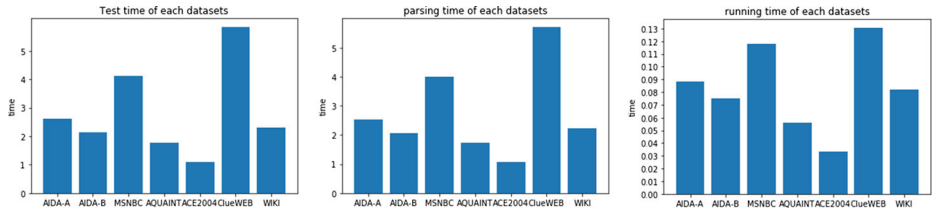
3.4 Efficiency and feasibility

Finally, we report the running time (in seconds) of our LoG model to demonstrate its feasibility in practice. The overall testing process can be roughly divided into two major steps: 1) We select mention neighbors via dependency parsing, co-reference linking and the CoSimTC algorithm (Section 2.1), and determine document keywords using the Sent2Word

1. The Electoral division of Apsley is one of the 15 electorates or seats in the Tasmanian Legislative Council.
2. It is the second-largest upper house electorate in the state by area after Murchison.
3. The total area of the Apsley division is 19,204 km² .
4. The electorate created in 1999 is named after the Apsley River and the .
5. These were named by William Lyne in honour of Lord Apsley, earl of Bathurst.
6. As of October 2010, there were 22,593 enrolled voters in Apsley.
7. The current member in the is who was elected in 2004 and re-elected unopposed in 2010. Members serve six-year terms, before facing re-election: the next election in Apsley is due in 2016.
8. Towns within the division include: Pipers River, Scottsdale, Evandale, Swansea, Derby, Lilydale, Bridport, Campbell Town, , St Helens, Branxholm, Avoca, Fingal, Bicheno, Bagdad, Bellingham, Tomahawk, Ross, St Marys, Rossarden

CoSimTC		CoSimTC + Keywords	
'Electoral_division_of_Murchison'	12.68%	'Electoral_division_of_Murchison'	13.20%
'Electoral_district_of_Murchison-Eyre'	13.09%	'Electoral_district_of_Murchison-Eyre'	12.95%
'Murchison,_Victoria'	12.38%	'Murchison,_Victoria'	12.90%

Figure 8 A demonstration of adding keyword information using Sent2Word



(a) Total time (b) Time of neighbor and keyword selection (c) Time of NN model

Figure 9 Running time of LoG model on the public datasets

strategy (Section 2.2). 2) Based on the extracted neighbors and keywords, we construct feature and subgraph vectors for each entity and feed them into our GAT-based network model to obtain the final entity linking results (Section 2.3). Therefore, we display the total running time as well as the cost of each step separately in Figure 9. The experiments are conducted on GPU 2080ti and the average performance per document is reported. We observe that the first step (i.e., dependency parsing and context selection) dominates the total running time, and the overall efficiency degrades with the increase of document length and complexity (e.g., number of mentions). To better understand this phenomenon, we further investigate the correlation between the context selection time and the number of mentions contained in the document. We notice that keyword detection using our Sent2Word algorithm is quite efficient, whereas the time required for dependency parsing and co-reference linking increases dramatically when the document is longer and contains more mentions. This is inevitable in our framework. However, the LoG model can still finish entity linking for each document in less than 6 seconds for all the public datasets, which verifies its feasibility in practice.

4 Related work

In this section, we will summarize existing entity disambiguation models and discuss the difference of our LoG framework.

Local ED model disambiguates each mention independently. The features implemented by local models mostly include string features, context words features and entity properties. Early local ED models measure the string distance between the candidate entity and the surface form of the mention. Distance measures includes Dice coefficient score [24], edit distance [3, 25], Hamming distance [6], etc. These name string comparison features are fundamental and effective for disambiguating different candidate entities and they are still commonly used in modern ED systems. With the development of deep learning, ED model begins to train embeddings of both words and entities. Recent work has proposed a series of effective context-entity similarity features based on embeddings. ETHZ-attention mechanism [10] is one of the classic local model, which effectively selects words which contribute to the disambiguation of candidate entities. Berkeley-CNN [8] is also an effective local model that captures the similarity between entity and context words by implementing CNNs at different granularities. He et al. [17] learns the representation of each document applying Stacked Denoising Auto-encoders. Gupta et al. [14], Nie et al. [28] both integrated entity description and entity type information to the disambiguation model, where [14] adopted CNN and LSTM-encoder for these source of information, and [28] implemented

co-attention for mention-entity description and it Incorporated entity type to bridge the gap between entity description and mention context.

As local models only consider each mention linking decisions individually, while ignoring the coherence between mentions, researchers begin to explore global model to disambiguate entities collectively, which is called collective entity linking. It is based on the intuition that mentions within the same document should be coherent with each other. Hachey et al. [15] connects each mention to only one entity via dense sub-graph estimation algorithms, and [1, 12, 15, 16] implement PageRank or Random Walk to perform collective entity linking. Recently, [4, 7] utilize different kinds of auxiliary knowledge (e.g., latent type information, Wikipedia articles and structures, etc.) to further improve the performance of collective entity linking. However, these "all mention coherence" solutions could potentially introduce noisy contextual information into the model, and they often have expensive computational cost because they need to measure the pairwise coherence between the candidate entities of different mentions. Recent works [3, 20, 21, 30] propose that the topical coherence only needs to be achieved within a subset of mentions within one document, which improves the computational efficiency. Similarly, [32] claims that each linking decision only needs to be topically coherent with one of the other linking in a document, and it proposes the Pair-Linking algorithm to efficiently resolve the document-level coherence. Yang et al. [39] also focuses on solving the problems caused by "all mention coherence", which leads to high computation complexity and sub-optimal coherence, and it proposes a dynamic context augmentation strategy to dynamically augment coherent entities when linking the current mention.

Existing work of ED offers diverse useful methods to perform entity disambiguation. Different from these methods, we propose an ED model which constructs the mention graph on the basis of semantic parse tree to measure the syntactic structure closeness between mentions. We also locate topical content from each document and extract keywords from the limited important part of text, which improves the effectiveness of topical keyword extraction process. In addition, we implement GAT to build entity graph to capture entity coherence to improve the performance of our ED model.

5 Conclusion

In this paper, we propose a novel distance measure CoSimTC based on parse tree to produce mention neighbors. It combines the benefit of parse tree distance and sequence distance and solves the cross-tree problem at document level. The mention neighbors derived by CoSimTC are helpful for the target mention to integrate useful global information within the document. Furthermore, we observe that, besides the local neighbors extracted by CoSimTC, the correct entity should also be coherent to the keywords of the document. We propose a keyword extraction approach Sent2Word which first locates the key sentences of each document and then extracts keywords from only the topical content, thus detecting keywords information more accurately and faster. We also propose a novel neural network model based on graph attention network (GAT) to integrate both local and global information and explore the relatedness between entities flexibly. Compared to existing state-of-the-art entity disambiguation methods, our Locally-Global (LoG) model achieves competitive performance and has the best average F1 score on five widely-used public datasets. The experimental results also demonstrate the benefit of the mention neighbors produced by CoSimTC, compared to using sequence distance directly, as well as the necessity of adding keywords information for entity disambiguation.

References

1. Alhelbawy, A., Gaizauskas, R.: Graph ranking for collective named entity disambiguation. In: ACL, vol. 2, pp. 75–80 (2014)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
3. Cao, Y., Hou, L., Li, J., Liu, Z.: Neural collective entity linking. In: COLING, pp. 675–686 (2018)
4. Chen, S., Wang, J., Jiang, F., Lin, C.Y.: Improving entity linking by modeling latent entity type information. In: AAAI (2020)
5. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: EMNLP-CoNLL (2007)
6. Dredze, M., McNamee, P., Rao, D., Gerber, A., Finin, T.: Entity disambiguation for knowledge base population. In: COLING, pp. 277–285 (2010)
7. Fang, Z., Cao, Y., Li, Q., Zhang, D., Zhang, Z., Liu, Y.: Joint entity linking with deep reinforcement learning. In: The World Wide Web Conference, pp. 438–447 (2019)
8. Francis-Landau, M., Durrett, G., Klein, D.: Capturing semantic similarity for entity linking with convolutional neural networks. In: NAACL, pp. 1256–1261 (2016)
9. Gabrilovich, E., Ringgaard, M., Subramanya, A.: Facc1: Freebase annotation of clueweb corpora Note: http://lemurproject.org/clueweb09/FACC1/ Cited by 5 (2013)
10. Ganea, O.E., Hofmann, T.: Deep joint entity disambiguation with local neural attention. In: EMNLP, pp. 2619–2629 (2017)
11. Ganea, O.E., Ganea, M., Lucchi, A., Eickhoff, C., Hofmann, T.: Probabilistic bag-of-hyperlinks model for entity linking. In: WWW, pp. 927–938 (2016)
12. Guo, Z., Barbosa, D.: Robust entity linking via random walks. In: CIKM, pp. 499–508. ACM (2014)
13. Guo, Z., Barbosa, D.: Robust named entity disambiguation with random walks. Semant. Web **9**(4), 459–479 (2018)
14. Gupta, N., Singh, S., Roth, D.: Entity linking via joint encoding of types, descriptions, and context. In: EMNLP, pp. 2681–2690 (2017)
15. Hachey, B., Radford, W., Curran, J.R.: Graph-based named entity linking with wikipedia. In: WISE, pp. 213–226. Springer (2011)
16. Han, X., Sun, L., Zhao, J.: Collective entity linking in Web text: a graph-based method. In: SIGIR, pp. 765–774. ACM (2011)
17. He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., Wang, H.: Learning entity representation for entity disambiguation. In: ACL(Short Papers), vol. 2, pp. 30–34 (2013)
18. Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G.: Robust disambiguation of named entities in text. In: EMNLP, pp. 782–792 (2011)
19. Hua, W., Wang, Z., Wang, H., Zheng, K., Zhou, X.: Short text understanding through lexical-semantic analysis. In: ICDE, pp. 495–506 (2015)
20. Hua, W., Zheng, K., Zhou, X.: Microblog entity linking with social temporal context. In: SIGMOD, pp. 1761–1775 (2015)
21. Hua, W., Wang, Z., Wang, H., Zheng, K., Zhou, X.: Understand short texts by harvesting and analyzing semantic knowledge. IEEE Trans. Knowl. Data Eng. **29**(3), 499–512 (2017)
22. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2016)
23. Le, P., Titov, I.: Improving entity linking by modeling latent relations between mentions. In: ACL, pp. 1595–1604 (2018)
24. Lehmann, J., Monahan, S., Nezda, L., Jung, A., Shi, Y.: Lcc approaches to knowledge base population at tac 2010. In: TAC (2010)
25. Liu, X., Li, Y., Wu, H., Zhou, M., Wei, F., Lu, Y.: Entity linking for tweets. In: ACL, vol. 1, pp. 1304–1311 (2013)
26. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: CIKM, pp. 233–242. ACM (2007)
27. Mueller, D., Durrett, G.: Effective use of context in noisy entity linking. In: EMNLP, pp. 1024–1029 (2018)
28. Nie, F., Cao, Y., Wang, J., Lin, C.Y., Pan, R.: Mention and entity description co-attention for entity disambiguation. In: AAAI (2018)
29. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the Web. Tech. rep., Stanford InfoLab (1999)
30. Pappu, A., Blanco, R., Mehdad, Y., Stent, A., Thadani, K.: Lightweight multilingual entity extraction and linking. In: WSDM, pp. 365–374. ACM (2017)

31. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
32. Phan, M.C., Sun, A., Tay, Y., Han, J., Li, C.: Neupl: Attention-based semantic matching and pair-linking for entity disambiguation. In: CIKM, pp. 1667–1676. ACM (2017)
33. Ratinov, L., Roth, D., Downey, D., Anderson, M.: Local and global algorithms for disambiguation to wikipedia. In: ACL, pp. 1375–1384 (2011)
34. Seidman, S.B.: Network structure and minimum degree. Social Netw. **5**(3), 269–287 (1983)
35. Tixier, A., Malliaros, F., Vazirgiannis, M.: A graph degeneracy-based approach to keyword extraction. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1860–1870 (2016)
36. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
37. Xin, K., Hua, W., Liu, Y., Zhou, X.: Entity disambiguation based on parse tree neighbours on graph attention network. In: 523–537 (2019)
38. Yamada, I., Shindo, H., Takeda, H., Takefuji, Y.: Joint learning of the embedding of words and entities for named entity disambiguation. In: CoNLL, p. 250 (2016)
39. Yang, X., Gu, X., Lin, S., Tang, S., Zhuang, Y., Wu, F., Chen, Z., Hu, G., Ren, X.: Learning dynamic context augmentation for global entity linking. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 271–281 (2019)