# NeuPL: Attention-based Semantic Matching and Pair-Linking for Entity Disambiguation

Minh C. Phan[1] Aixin Sun[1] Yi Tay[1] Jialong Han[1] Chenliang Li[2]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]State Key Lab of Software Engineering, Computer School, Wuhan University, China
phan0050@e.ntu.edu.sg;axsun@ntu.edu.sg;ytay017@e.ntu.edu.sg;jialonghan@gmail.com;cllee@whu.edu.cn

## ABSTRACT

Entity disambiguation, also known as entity linking, is the task of mapping mentions in text to the corresponding entities in a given knowledge base, *e.g.,* Wikipedia. Two key challenges are making use of mention's context to disambiguate (*i.e.,* local objective), and promoting coherence of all the linked entities (*i.e.,* global objective). In this paper, we propose a deep neural network model to effectively measure the semantic matching between mention's context and target entity. We are the first to employ the long short-term memory (LSTM) and attention mechanism for entity disambiguation. We also propose Pair-Linking, a simple but effective and significantly fast linking algorithm. Pair-Linking iteratively identifies and resolves pairs of mentions, starting from the most confident pair. It finishes linking all mentions in a document by scanning the pairs of mentions at most once. Our neural network model combined with Pair-Linking, named **NeuPL**, outperforms state-of-the-art systems over different types of documents including news, RSS, and tweets.

## KEYWORDS

Entity Disambiguation; Semantic Matching; Pair-Linking

## 1 INTRODUCTION

Documents often contain mentions of named entities such as people, places, and organizations, etc. These mentions are usually ambiguous due to its polymorphic nature *i.e.,* the same entity may be mentioned in different surface forms, and the same surface form may refer to different named entities. We use a sentence from Wikipedia as an example: *"Before turning seven, Tiger won the Under Age 10 section of the Drive, Pitch, and Putt competition, held at the Navy Golf Course in Cypress, California".* Without considering its context, the word 'Tiger' can refer to the American golfer *Tiger Woods*, the budget airline *Tiger Air*, or beer brand *Tiger Beer*. Considering its context, the mention 'Tiger' in the given sentence should be linked to golfer *[Tiger_Woods].*

Entity disambiguation is a critical task in bridging the unstructured text and structured knowledge bases. The result of entity disambiguation is beneficial for many tasks, including knowledge base population, information retrieval and extraction, question answering, and content analysis. Therefore, entity disambiguation has constantly gained research attention.

ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3132963

**The Research Problem.** Formally, given a document $d$ containing a set of mentions $M = \{m_1, ..., m_N\}$ and a target knowledge base $K = \{e_1, ..., e_{|K|}\}$, the task of entity disambiguation is to find a mapping $M \mapsto K$ that links each mention to a correct entity in the knowledge base. We denote the output of the matching as an $N$-tuple $\Gamma = (t_1, ..., t_N)$ where $t_i$ is the assigned entity for mention $m_i$ and $t_i \in K$. Similar to most recent studies [14, 34, 47], we do not address the issue of mention extraction or not-in-list identification in this study. That is, a given mention is guaranteed linkable to one entry in the given knowledge base.

**Solution Overview.** Early approaches rely on local context of the mentions, and recent methods utilize both the local context and the coherence between linked entities to improve the mapping result. Shown in Equation 1, the common approach for finding the optimal matching, denoted by $\Gamma^*$, is to maximize the local confidence of each assignment $\phi(m_i, t_i)$, while enforcing the coherence among all linked entities $\psi(\Gamma)$ [8, 36].

$$\Gamma^* = \arg\max_{\Gamma} \left[ \sum_{i=1}^{N} \phi(m_i, t_i) + \psi(\Gamma) \right] \tag{1}$$

Local confidence or local score $\phi(m_i, t_i)$ reflects the likelihood of mapping $m_i \mapsto t_i$ solely based on $m_i$'s context and $t_i$'s profile, without considering any other mappings in the same document. In other words, $\phi(m_i, t_i)$ can be computed in isolation. Various methods have been proposed to model mention's local context, ranging from the simple vector space models, information retrieval based approaches, to supervised models such as binary classification, learning to rank, and probabilistic models [38]. Recently, deep neural network (DNN) has also been investigated for the linking task and promising results have been achieved [6, 12, 20, 47]. These works are based on the general idea that the disambiguation problem is a semantic matching problem. However, the existing solutions do not fully utilize the information embedded in the mention's context. Firstly, the mention's position is ignored in their models. For example, in [20], a DNN learns the representation of the local context without specifying which mention to be focused on. If the context consists of two or more mentions, all mentions are viewed as identical. Secondly, existing approaches ignore the word order in the context (*i.e.,* bag of words model). However, word ordering is critical for natural language understanding.

In this paper, we develop a deep neural network that takes into consideration *mention's positional information*, and *word ordering*.

It is achieved through two Long Short-Term Memory (LSTM) networks modeling the context on the left- and right-sides of a mention. Furthermore, to assist the matching in ambiguous cases where noises present in mention's context, we employ *attention mechanism* into the designed model. To the best of our knowledge, we are the first to build such a comprehensive neural network to model the semantic matching for entity disambiguation.

The coherence of the linked entities $\psi(\Gamma)$ is often computed from the semantic relatedness between entities, denoted by $\psi(t_i, t_j)$. To maximize $\psi(\Gamma)$, state-of-the-art entity disambiguation systems usually use graph-based algorithm to utilize the semantic relatedness between the linked entities, *i.e.,* collective linking (CL). While approaches like dense subgraph [23, 33], PageRank [1, 16–18, 35, 48], and Iterative Substitution [39] can give a boost to disambiguation accuracy, they suffer from high computational cost. On the other hand, human being does not need to consider all other mentions in the same document to disambiguate a mention. Instead, disambiguation can be performed simply based on *a pair of mentions with strong semantic associations*. For example, a pair of mentions like 'Java' and 'Oracle' help both to link to the right entities.

Based on this intuition, we propose a new collective linking algorithm named **Pair-Linking** (PL). In simple words, Pair-Linking resolves the most confident pair of mentions each time. The confidence score of a pair is computed from $\phi(m_i, t_i)$, $\phi(m_j, t_j)$, and $\psi(t_i, t_j)$, assuming $m_i \mapsto t_i$ and $m_j \mapsto t_j$. By always linking the next most confident pair of mentions, Pair-Linking completes collective linking of all mentions in a document *in a single iteration*. Despite its simplicity, Pair-Linking achieves comparable or even better results than the state-of-the-art CL algorithms, while being nearly 100 times faster for long documents in our experiments.

**Our Contributions.** In summary, our main contributions are:

- We present a deep neural network model that measures the semantic similarity between a mention's context and a target entity. Our DNN model is designed in a way to fully utilize the embedded information including mention's position and word order. We also use the attention mechanism to highlight the informative parts in the local context.
- We propose a simple and significantly fast collective linking algorithm named Pair-Linking. The algorithm achieves comparable or even better linking accuracy with state-of-the-art algorithms, while being nearly 100 times faster than the latter.
- We evaluate NeuPL on 7 publicly available datasets using the Gerbil benchmarking framework. The result shows that our system achieves state-of-the-art performance on these datasets with different types of documents including news articles, RSS, and tweets. Our results also show the efficiency of the Pair-Linking algorithm against other collective linking alternatives.

## 2 RELATED WORK

**Mention context matching.** Early studies disambiguate mentions based on lexical matching between the mention's surrounding words and the entity profile [5]. Entity disambiguation problem is viewed as a word-sense disambiguation problem in [30] and the similarity between a mention's local context and an entity is measured by word overlap. Entity disambiguation is modeled as a retrieval task in [9] where the entities are the documents to be ranked, and a query is formed by a mention and its context.

Recently, semantic matching powered by DNN shows promising improvement on entity disambiguation task. The common approach is to learn latent representations of local context and entity, respectively; then to use similarity measure (*e.g.,* cosine similarity) to estimate the semantic matching between a mention and an entity. In [20], stacked denoising auto-encoders are used to learn the representations of documents and entities. The disambiguation is performed by calculating the similarity between the two representations. Zwicklbauer *et al.* [48] learn continuous representation of entity by using word2vec on the corpus created from Wikipedia and Google Wikilinks. Blanco *et al.* [4] use embeddings of the words appearing in entity's description to train the representation of the associated entity. Two recent studies [10, 47] co-train word and entity embeddings in the same vector space. Their results show that the co-trained vectors improve the quality of both word and entity representations which in turn benefits the disambiguation task. To obtain the representation of local context, authors in [4, 10, 47] aggregate the embedding of words in local context while [48] use doc2vec to compute the representation. DNN, specifically convolutional neural network has also been used to model the context surrounding mention in [6, 12, 42].

**Collective Linking.** With the assumption that all the linked entities shall be related, topical coherence in entity linking was introduced [8]. The relatedness between two entities can be computed in multiple ways, *e.g.,* based on the incoming links and categories of the entities [32] and Normalized Google Distance [7, 19, 36].

To perform collective linking, a mention-entity graph is constructed with (i) edges between mention and entity, weighted by score of local context matching, and (ii) edges between entity and entity, weighted by their relatedness. Hoffart *et al.* [23] cast the joint mapping into the problem of identifying dense subgraph that contains exactly one mention-entity edge for each mention. Many other works are based on the Random Walk and PageRank algorithm [1, 16–18, 33, 35, 48]. The stabilized scores of entities obtained by Random Walk or PageRank will be used to select the matching entity for each mention. Although graph-based approaches are shown to produce robust and competitive performance, they are computationally expensive because the graph may contain hundreds of vertices for documents with multiple mentions. Shen *et al.* [39] propose an iterative substitution algorithm to jointly optimize the identification of the mapping entities, so that the sum of pair-wise semantic similarities between all linked entities is maximized. In a very recent study [34], fast linking is achieved by the Forward-Backward algorithm [2], which only considers the adjacent assignments in coherence optimization.

Probabilistic models have also been studied for the entity disambiguation task. Kataria *et al.* [26] use topic model to learn the entity-word association and utilize Wikipedia category hierarchy to capture the co-occurrence patterns between entities. In [14], mentions are jointly disambiguated by combining entity co-occurrence statistics with the local information captured from mention and their surrounding context.

**Discussion.** Existing solutions for semantic matching using DNN ignore word order or mention's location in local context. Different from these methods, our model utilizes LSTMs and attention mechanism to model word sequence as well as mention's positional information in the local context. We also introduce the Pair-Linking algorithm to achieve fast and accurate disambiguation results.

## 3 PRELIMINARIES

Given a document with a set of mentions to be linked, candidate entities for each mention are identified based on the mention's surface form. Then a DNN in NeuPL is used to estimate the semantic matching score between a mention and its candidate entity $\phi(m_i, t_i)$ (see Section 4). $\psi(t_i, t_j)$ is computed from the entity embeddings of $t_i$ and $t_j$ respectively (see Section 3.2). Finally Pair-Linking is applied to collectively link all mentions, detailed in Section 5.

### 3.1 Candidate Generation and Filtering

As a common approach [34, 38, 48], our candidate generation is purely based on the textual similarity between a mention's surface form and an entity's title including all its variants. We used the name dictionary based techniques for candidate retrieval [38]. The dictionary is built by exploiting entity titles, anchor texts, redirect pages, and disambiguation pages in Wikipedia. If a given mention does not present in the dictionary, we use its n-grams to retrieve the candidates. We further improve the recall of candidate generation by correcting the mention's boundary. In several situations, a given mention may contain trivial words (*e.g.,* the, Mr. , CEO, president) that are not indexed by the dictionary. We use an off-the-shelf Named Entity Recognizer (NER) to refine the mention's boundary in these cases.[1] As in [15], we also utilize the NER output to expand the mention's surface form. Specifically, if mention $m_1$ appears before $m_2$ and $m_1$ contains $m_2$ as a substring, we consider $m_1$ as an expanded form of $m_2$, and candidates of $m_1$ will be included to the candidate set of $m_2$.

We train Gradient Boosted Regression Trees model [13] as a candidate ranker to reduce the size of candidate set. For each pair of (mention, candidate) *i.e.,* $(m, e)$, we use the following statistical and textual features for ranking.

- Prior probability $P(e|m)$. $P(e|m)$ is the likelihood that the mention with surface form $m$ being mapped to entity $e$. $P(e|m)$ is pre-calculated based on the hyperlinks in Wikipedia.
- String similarity. We use several string similarity measures including: (i) edit distance, (ii) whether mention $m$ exactly matches entity $e$'s name, (iii) whether $m$ is a prefix or suffix of the entity name, and (iv) whether $m$ is an abbreviation of the entity name. Note that the string similarity features are calculated for the original mention as well as the boundary-corrected mention and the expanded mention described earlier.

We use the IITB labeled dataset [27] to train the ranker and take the top 20 scored entities as the final candidate set for each mention. Taking fewer candidates per mention will lead to low recall while using more candidates degrades disambiguation accuracy in later step. Similar observations are also reported in [14, 48].

[1]We used the Standford NER tool in this work.

### 3.2 Word and Entity Embeddings

Embedding models aim to generate a continuous representation for every word, such that two words that are close in meaning are also close in the embedding vector space. It assumes that words are similar if they co-occur often with the same words [31]. Correspondingly, we can assume two entities to be semantically related if they are found in analogous contexts. The context is defined by the surrounding words or surrounding entities.

Jointly modeling words and entities in the same continuous space have been shown to improve the quality of both word and entity embeddings [46] and benefit entity disambiguation task [10, 47]. In NeuPL, we use the word2vec with skip-gram model [31] to jointly learn the distributional representation of word and entity.

Let $\mathcal{T}$ denote the set of tokens. Token $\tau \in \mathcal{T}$ can be either a word (*e.g.,* Tiger, Wood) or an *entityID* (*e.g., [Tiger_Wood]*). Suppose $\tau_1, ..., \tau_N$ is a given sequence, the model tries to maximize the following average log probability:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{-c \le j \le c, j \ne 0} \log P(\tau_{i+j}|\tau_i) \tag{2}$$

where $c$ is the size of context window, $\tau_i$ denotes the target token, and $\tau_{i+j}$ is a context token. The conditional probability $P(\tau_{i+j}|\tau_i)$ is defined by the softmax function:

$$P(\tau_O|\tau_I) = \frac{\exp({v'_{\tau_O}}^{\mathsf{T}} v_{\tau_I})}{\sum_{\tau \in \mathcal{T}} \exp({v'_{\tau}}^{\mathsf{T}} v_{\tau_I})} \tag{3}$$

where $v_\tau$ and $v'_\tau$ are the 'input' and 'output' vector representations of $\tau$, respectively. After training, we use the 'output' $v'_\tau$ as the embedding for word or entity.

To co-train word and entity embeddings, we create a 'token corpus' by exploiting the existing hyperlinks in Wikipedia. Specifically, for each sentence in Wikipedia which contains at least one hyperlink to another Wikipedia entry, we create an additional sentence by replacing each anchor text with its associated entityID. Then, for each Wikipedia page, we also create a 'pseudo sentence' which is the sequence of entityIDs linked from this page, in the order of their appearances. For example, assume that the Wikipedia page about *Tiger Wood* contains only 2 sentences: "*Woods*$_{\text{[Tiger\_Woods]}}$ was born in *Cypress*$_{\text{[Cypress,\_California]}}$. He has a niece, *Cheyenne Woods*$_{\text{[Cheyenne\_Woods]}}$.", the following sentences are in our 'token corpus'.

- Wood was born in Cypress. He has a niece, Cheyenne Woods.
- *[Tiger_Woods]* was born in *[Cypress,_California]*. He has a niece, *[Cheyenne_Woods]*.
- *[Tiger_Woods] [Cypress,_California] [Cheyenne_Woods]*.

### 3.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a specific Recurrent Neural Network that has been widely used to model variable-length sequence [22, 25]. In NeuPL, we employ LSTM to learn the representations of mention context and entity description. The LSTM variant we are using is similar to the one presented in [25], shown in Figure 2. Specifically, given input sequence $S = \{w_1, w_2, ..., w_n\}$ where $w_t$ is a word embedding to be passed as input at time $t$, the LSTM unit will output hidden vector $h_t$ for each time step $t$, and $h_t$ is calculated using the following equations:

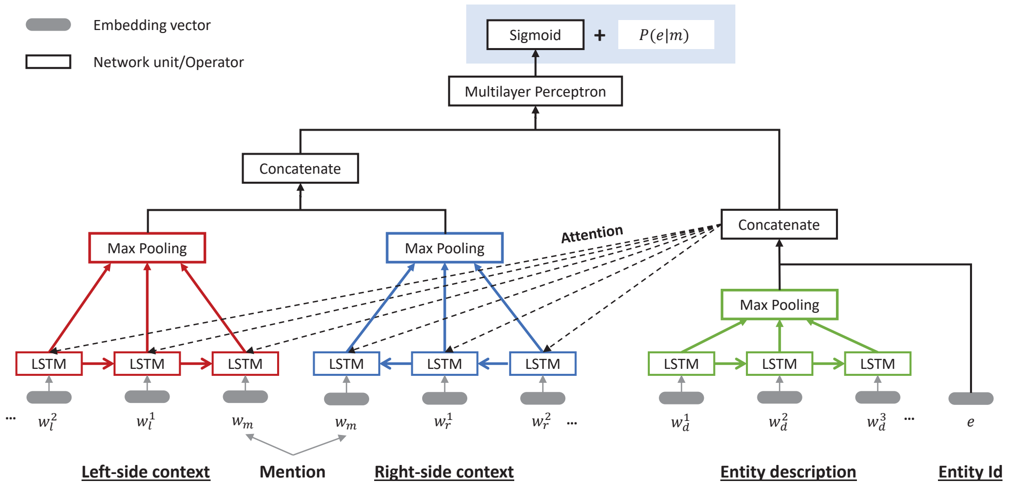


**Figure 1: Neural architecture for leaning the semantic matching between mention context and an entity (best viewed in color).**

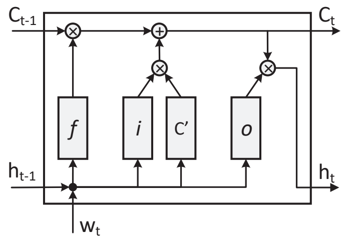


**Figure 2: A long short-term memory block**

$$\begin{array}{l|l} i_t = \sigma(W_i w_t + U_i h_{t-1} + b_i) & C'_t = tanh(W_c w_t + U_c h_{t-1} + b_c) \\ f_t = \sigma(W_f w_t + U_f h_{t-1} + b_f) & C_t = i_t * C'_t + f_t * C_{t-1} \\ o_t = \sigma(W_o w_t + U_o h_{t-1} + b_o) & h_t = o_t * tanh(C_t) \end{array}$$

In above equations, $i$, $f$, $o$, $C$ are input gate, forget gate, output gate, and cell memory, respectively. $\sigma$ is sigmoid function. $W_*$, $U_*$ and $b_*$ are network parameters to be learned during training.

Instead of taking the last hidden state $h_n$, we use the max-pooling result of all hidden states over time steps *i.e.*, max-pooling$(h_1, ..., h_n)$, as the final representation for sequence $S$.

## 4 DNN MODEL FOR SEMANTIC MATCHING

The deep neural architecture used in NeuPL for semantic matching is illustrated in Figure 1. Shown in the figure, two LSTM networks are used to capture the left-side and the right-side contexts of a mention. Another LSTM is used to learn the latent representation of entity profile. The latent vectors are then concatenated and passed as input to a multilayer perceptron (MLP). The MLP outputs a scalar to be used as the semantic matching score. To be detailed shortly, the whole model is trained using Wikipedia data, where a mention is an anchor text and its linked Wikipedia entry is the ground truth.

**Mention's local context.** We use the words within a window size $c$ on both sides of a mention as its local context. Specifically, let $\langle w_\ell^c, \dots, w_\ell^1, m, w_r^1, \dots, w_r^c \rangle$ be the local context for mention $m$, we define its *left-side context* and *right-side context* as $\langle w_\ell^c, \dots, w_\ell^1, m \rangle$ and $\langle m, w_r^1, \dots, w_r^c \rangle$, respectively. For simplicity, we use a single token $m$ to represent a mention here (and also in Figure 1), but it can consist of multiple tokens in implementation.

Shown in Figure 1, two separate LSTM networks are used to encode the mention's left- and right-side contexts respectively. The left-side context is passed in forward direction *i.e.*, $w_\ell^c \rightarrow w_\ell^{c-1} ... w_\ell^1 \rightarrow m$, while the right-side context is passed in backward direction *i.e.*, $m \leftarrow w_r^1 ... w_r^{c-1} \leftarrow w_r^c$. By doing so, we align mention $m$ at the end of each sequence, so that LSTM is aware of its position. This is important because local context may contain more than one mention, and the model needs to focus on the right mention for correct linking. For example, given two mentions in a sentence: '*The Tiger Woods Foundation*$_{m_1}$ was established in 1996 by *Woods*$_{m_2}$', without specifying mentions' locations, the context matches both entities [Tiger_Woods_Foundation] and [Tiger_Woods], leading to wrong disambiguations of $m_1$ and $m_2$.

Compared with the model in [42], where additional positional embedding is used to encode a mention's location, the way we align mention at the end of each side of the context is simpler, and at no extra cost. On the other hand, our strategy is similar to the idea of target-dependent LSTM used for sentiment classification in [44], with two major differences. First, the model in [44] uses the last hidden vector of LSTM's output to represent the context, while we max-pool all hidden vectors of the LSTM on each side of the context. Second, we employ the attention mechanism to improve the quality of representation. Specifically, the hidden vectors obtained from LSTM will pass through an attention module to emphasize the informative parts. Then, max-pooling is applied over new hidden states to produce a fixed-length representation of the local context.

**Entity profile.** To build entity profiles, we exploit the pre-trained embeddings of entities in Section 3.2. Because we treat entities similar to words in training, an entity embedding encodes not only semantic information but also syntactic knowledge about how the associated entity is mentioned. For example, entities about geographic location will be more likely to be placed close to prepositions 'in' or 'at' in the embedding space. We further complete entity profiles by including their descriptions. For each entity, we take the first 150 words from its Wikipedia page as its description.

We again use a single-directional LSTM network with max-pooling (see Figure 1) to learn the description representation. The learned vector is then concatenated with the pre-trained entity embeddings to form the complete entity profile representation.

Because the entire DNN model is pre-trained using Wikipedia data, the training samples are biased toward the standard way of putting hyperlinks in Wikipedia. In the designed model, we do not explicitly specify the entity's name in an entity description, neither declare any mention's boundary in its local context. This prevents the model from over-fitting the training data; hence the model generalizes better for semantic matching with data from outside of Wikipedia.

**Attention mechanism.** Our target is to best estimate the matching score given: (i) a vector representing a mention's local context, and (ii) a vector encoding an entity profile. However local context usually contains irrelevant information that may distract the matching. Next, we present the use of attention mechanism to address this issue.

The idea of attention is to learn to highlight the more informative parts of the inputs to the given attention vector. The mechanism have been successfully applied in many natural language processing tasks including translation [3], summarization [37] and question-answer matching [21, 41, 43]. In NeuPL, we use entity profile's representation as the attention vector to highlight the relevant parts in mention's local context. Specifically, given a hidden vector $h_t$ from a LSTM block at time step $t$, and an attention vector $p$ (*i.e.,* an entity profile's representation, see Figure 1), the re-weighted hidden vector $\bar{h_t}$ is defined by:

$$z_t = tanh(V_p p + V_h h_t) \tag{4}$$

$$s_t = softmax(v, z_t) \tag{5}$$

$$\bar{h_t} = h_t s_t \tag{6}$$

where $V_*$ and $v$ are attention parameters to be learned during training. By the formulas, $h_t$ will be given more weight if it is more relevant to the attention vector $p$. In our context, the attention mechanism works like tf-idf weighting scheme that emphasizes the embedded information in local context (*i.e.,* $h_i$) that are more relevant to the target entity. It helps to eliminate noise hence to improve semantic matching.

**Training objective.** The concatenated vector for a mention's context and the concatenated vector for entity profile are concatenated again, to be the input of a multilayer perceptron (MLP). The MLP consists of two fully connected hidden layers and a single-node output layer. The output of MLP is mapped to a scalar through sigmoid function. Let $0 < o < 1$ denote the final output and $g \in \{0, 1\}$ be the groundtruth label indicating positive/negative sample. The proposed deep neural network is trained with the following binary cross-entropy (CE) loss function, with backpropagation.

$$L_{CE}(o, g) = g \log(o) + (1 - g) \log(1 - o) \tag{7}$$

**Combination with prior probability.** Prior probability $P(e|m)$ is the likelihood of mention $m$ with the given surface form being linked to entity $e$. It is approximated by the hyperlink statistic in Wikipedia [38]. Although $P(e|m)$ completely ignores the surrounding context of $m$, it is recognized as an important feature for entity disambiguation [24]. For example, popular entities (*e.g.,* countries, famous people) generally do not require supporting context when being mentioned. The local contexts in these cases may be general, therefore semantic matching based on context becomes less effective. On the other hand, such mentions can be easily disambiguated by adopting the prior probability. In NeuPL, we linearly combine the semantic matching score computed by DNN and the prior probability:

$$\phi(m_i, e_i) = (1 - \alpha)\sigma(m_i, e_i) + \alpha P(e_i|m_i) \tag{8}$$

In this equation, $\alpha$ is a weight factor, and $\sigma(m_i, e_i)$ is the output of DNN given $m_i$'s context and entity $e_i$ (see Figure 1).

**Algorithm 1:** Pair-Linking algorithm

```
input  : $N$ mentions $(m_1, \ldots, m_N)$. Mention $m_i$ has candidate set
         $\{e_i | e_i \in C(m_i)\}$
output : $\Gamma = (t_1, \ldots, t_N)$
1  $t_i \leftarrow null, \forall t_i \in \Gamma$
2  for each pair $(m_i, m_j) \wedge m_i \neq m_j$ do
3      $Q_{m_i,m_j} \leftarrow$ top_pair($m_i$, $C(m_i)$, $m_j$, $C(m_j)$)
4      $Q$.add( $Q_{m_i,m_j}$ )
5  end
6  while $(\exists t_i \in \Gamma, t_i = null)$ do
7      $(m_i, e_i, m_j, e_j) \leftarrow$ most_confident_pair($Q$)
8      $t_i \leftarrow e_i$
9      $t_j \leftarrow e_j$
10     for $k := 1 \rightarrow N \wedge t_k = null$ do
11         $Q_{m_k,m_i} \leftarrow$ top_pair($m_k$, $C(m_k)$, $m_i$, $\{t_i\}$)
12         $Q_{m_k,m_j} \leftarrow$ top_pair($m_k$, $C(m_k)$, $m_j$, $\{t_j\}$)
13     end
14 end
```

## 5 PAIR-LINKING

We now present Pair-Linking (PL), a new linking algorithm to approximate topic coherence of the linked entities in the same document, *i.e.,* $\psi(\Gamma)$ in Equation 1.

Existing collective linking algorithms usually presume that each mapped entity needs to be related to *all* other linked entities in the same document [23, 39, 48]. This assumption may not hold in the case of long document which consists of mentions referring to different topics. Moreover, it is computational expensive to solve the coherence objective by considering all linked entities.

Pair-Linking works on the intuition that each linking only needs to be coherent with another linking. Specifically, $m_i$ is linked to $t_i$ if there is another supporting assignment $m_j \mapsto t_j$ such that, $t_i$ and $t_j$ are strongly semantically related. Based on the assumption, Pair-Linking iteratively identifies and resolves pairs of mentions, starting from the most confident pair. The confidence score of a pair of assignments $m_i \mapsto t_i$ and $m_j \mapsto t_j$ is defined as follows:

$$conf(i, j) = (1 - \beta)\frac{\left[\phi(m_i, t_i) + \phi(m_j, t_j)\right]}{2} + \beta\psi(t_i, t_j) \tag{9}$$

where $\beta$ is the coefficient reflecting the preference between the local scores and the pair-wise coherence. The coherence $\psi(t_i, t_j)$ of two linked entities $t_i$ and $t_j$ is the cosine similarity of their entity embeddings. In case one entity has no embedding (*e.g.,* a new entity), the coherence is zero.

Considering the set of decisions made by pair-linking, it results in an edge cover on entity graph where each assigned entity is forced to be coherent with at least another entity. Furthermore, Pair-linking resolves mentions in an iterative manner which encourages subsequent assignments to be consistent with previous ones.

Pair-Linking procedure is detailed in Algorithm 1. Specifically, Pair-Linking maintains a priority queue $Q$ and each element $Q_{m_i,m_j}$ tracks the most confident linking pairs involving mentions $m_i$ and $m_j$. $Q_{m_i,m_j}$ is initialized by calling function *top_pair*$(m_i, C_i, m_j, C_j)$, where $C_i$ is the set of candidate entities that mention $m_i$ can link to. The function returns a pair assignment $m_i \mapsto e_i$ and $m_j \mapsto e_j$, such that $e_i \in C_i$, $e_j \in C_j$, and the confidence score of the pair assignment is the highest among $C_i \times C_j$, according to Equation 9. After initialization, Pair-Linking iteratively retrieves the most confident pair assignment from $Q$ (Line 7) and links the pair of mentions to the associated entities (Lines 8-9). Then, Pair-Linking updates $Q$, more precisely, $Q_{m_k,m_i}$ and $Q_{m_k,m_j}$ (Lines 10-13). For $Q_{m_k,m_i}$, the possible pairs of assignments between $m_k$ and $m_i$ are now conditioned by $m_i \mapsto e_i$, and the same applies to $Q_{m_k,m_j}$.

**Early stop.** The most expensive part of the algorithm is the initialization of $Q$ which requires to compute *top_pair* between every two mentions. A straightforward implementation of the function *top_pair*$(m_i, C_i, m_j, C_j)$ will scan through all possible candidate pairs between the two mentions. Since only the pair of candidates with the highest confidence score is recorded for a pair of mentions $m_i$ and $m_j$, Pair-Linking uses *early stop* to prevent checking all possible candidate pairs in $C_i \times C_j$. Specifically, it sorts each candidate set by the local scores and traverses the sorted list in descending order. Early stop is applied if the current score is worse than the highest score by a specific margin, *i.e.,* the largest possible value of $\beta\psi(e_i, e_j)$, see Equation 9. Indeed, early stop significantly reduces the running time of Pair-Linking in experiments while maintains the correctness of the algorithm.

## 6 EXPERIMENT

We use Wikipedia dump on 01-Jul-2016 as the target knowledge base which consists of 5,187,458 entities. The neural network is trained using data solely from Wikipedia. Once trained, it can be used to disambiguate mentions in documents from general domain, *e.g.,* web pages, news, and tweets.

### 6.1 Experimental Setting

**Pre-trained embeddings**. We use Gensim, an efficient and reliable framework to co-train word and entity embeddings from the 'token corpus' described in Section 3.2. The embedding dimension is set to 400, window size is 5, and number of training iteration is 5. In the preprocessing step, we remove all numeric tokens and the tokens that appear less than 5 times in the whole corpus. All word tokens are lowercased. The final vocabulary contains 2,689,534 words and 2,863,704 *entityID*s. As a result, many unpopular entities are given a default embedding, *i.e.,* a zero vector. Disambiguation to these entities will be based on their entity descriptions.

**Neural network setting.** We set the context window size $c = 20$, meaning that the left 20 and the right 20 words around a mention are used as its local context words. Zero padding is used if a context has fewer words. The first 150 words in the entity's Wikipedia page are used as entity description. All words are lowercased and numbers are removed. We use the pre-trained word/entity embeddings as the input to the neural network. The embeddings are fixed during training and only the neural network parameters are learned.

**Table 1: Hyperparameter settings for neural network.**

| Network Unit Setting | | Training Setting | |
|---|---|---|---|
| LSTM hidden state size | 384 | Dropout | 0.3 |
| No of hidden layers in MLP | 2 | Epochs | 30 |
| Size of each hidden layer in MLP | 700 | Batch size | 256 |
| Activation for hidden layer in MLP | *tanh* | Optimizer | *Adam* |

To encode the word sequence, we use unidirectional LSTM (Section 3.3). We also experimented with bidirectional LSTM. It results in negligible improvement while doubling the number of network parameters and taking longer time to train. We hence adopt the unidirectional LSTM. Finally, Dropout layers are added after each input layer and hidden layer to prevent over-fitting. The settings of other neural network hyperparameters are listed in Table 1.

We utilize Wikipedia text and hyperlinks as training resource. For each entity, we randomly collect up to 100 of its anchor texts (*i.e.,* mentions) together with context words as positive training samples. Due to limitation of computational resources, we train the model on the aggregated candidate entities derived from the seven datasets (see Table 2). Because candidate generation for a mention is independent from entity linking, all candidate entities can be pre-determined for a given test dataset (see Section 3.1). The aggregated candidate set contains 27,444 entities, from which we collect 1,108,524 positive samples through Wikipedia hyperlinks. For each positive sample, we create 4 negative samples by replacing the correct entity with another randomly selected entity from the mention's candidate set. As the result, the neural network is trained by using 5,542,620 samples in total.

**NeuPL hyperparameter setting**. We have now learned the embeddings and trained the neural network in NeuPL. To perform entity disambiguation, two more hyperparameters need to be set, namely, $\alpha$ in Equation 8 and $\beta$ in Equation 9. The optimal setting of these two hyperparameters, however, is dataset-dependent. For a fair comparison with other methods, we set $\alpha$ and $\beta$ through 5-fold cross-validation. Specifically, we search for the best $\alpha$ and $\beta$ settings on 4 partitions through grid search from 0 to 1, with a step of 0.05. The best settings are then used to perform disambiguation on the remaining 1 partition. The result on a dataset is the aggregation of the predictions on 5 test partitions, which are then evaluated by the Gerbil framework.

**Rule-based disambiguation**. Because numeric tokens are removed in preprocessing, all numeric mentions are mapped to numeric entities regardless of local context. For example, a mention '12' will be disambiguated to entity *[12_(number)]*. Another rule we used is for mentions of three news agencies. These mentions usually appear at the beginning or the ending of a document. They can be easily detected and disambiguated, and do not contribute to the disambiguation of others. Specifically, the following three mentions have rule-based mappings: 'reuters' $\mapsto$ *[Reuters]*, 'associated press' $\mapsto$ *[Associated_Press]*, and 'afp' $\mapsto$ *[Agence_France-Presse]*.

**Table 2: Statistics of the 7 test datasets used in experiments. $|D|$, $|M|$, $Avg_m$, and $Length$ are number of documents, number of mentions, average number of mentions per document, and document length in number of words, respectively.**

| Dataset | Type | $\|D\|$ | $\|M\|$ | $Avg_m$ | Length |
|---|---|---|---|---|---|
| Reuters128 | news | 111 | 637 | 5.74 | 136 |
| ACE2004 | news | 35 | 257 | 7.34 | 375 |
| MSNBC | news | 20 | 658 | 32.90 | 544 |
| DBpedia | news | 57 | 331 | 5.81 | 29 |
| RSS500 | RSS-feeds | 343 | 518 | 1.51 | 30 |
| KORE50 | short sentences | 50 | 144 | 2.88 | 12 |
| Micro2014 | tweets | 696 | 1457 | 2.09 | 18 |

## 6.2 Datasets and Methods in Comparison

We evaluate NeuPL on 7 benchmark datasets from different domains, including short and long text, formal and informal text (details in Table 2). Note that, we only consider the mentions whose linked entities present in the Wikipedia dump; the same setting has been used in [14, 34, 47, 48]. We describe each dataset as follows:

- **Reuters128** contains 128 economic news articles taken from the Reuters-21587 corpus. There are 111 documents containing linkable mentions in the Wikipedia 01-Jul-2016 dump.
- **ACE2004** is a subset of ACE2004 co-reference documents annotated by Amazon Mechanical Turk. It has 35 documents, each has 7 mentions on average.
- **MSNBC** is created from MSNBC news articles. Each document contains 33 mentions on average.
- **DBpedia Spotlight (DBpedia)** is a news corpus and contains many non-named entity mentions *e.g.,* parent, car, dance, etc.
- **RSS500** is RSS feeds - short formal text collected from a wide range of topics *e.g.,* world, business, science, etc.
- **KORE50** contains 50 short sentences on various topics *e.g.,* music, celebrities, and business. Most mentions are first names referring to persons with high level of ambiguity.
- **Microposts2014 (Micro2014)** is a collection of tweets, introduced in the 'Making Sense of Microposts 2014' challenge. The dataset has train/test partitions. We use the test partition here so that our results can be compared with others.

We compare NeuPL against 5 state-of-the-art disambiguation systems whose results are also reported with the Gerbil framework. We acknowledge that there are other systems whose results are not reported with Gerbil framework. As the evaluation itself is a complicated issue [29, 45], we adopt Gerbil framework following the most recent studies [14, 48], such that all systems can be compared in future through the common protocol. The systems compared in our experiments are AIDA, Kea, WAT, and the most recent works PBoH and DoSeR.

- **AIDA** [23] uses dense subgraph algorithm to identify coherent mention-entity mappings.
- **Kea** [40] builds fine-granular context model and utilizes heterogeneous text sources as well as text created by automated multimedia analysis.
- **WAT** [35] system is an improved version of TagMe [11] with two configurations to approximate the global coherence objective, namely graph-based algorithm and vote-based algorithm.
- **PBoH** [14] is a light-weight disambiguator which is based on probabilistic graphical model to perform collective linking. The model calculates Wikipedia statistics about co-occurrence of words and entities to disambiguate mention.
- **DoSeR** [48] carefully designs the collective disambiguation algorithm using Personalized PageRank on the mention-candidate graph. It heavily relies on the collective linking algorithm to produce good results.

Additionally, we report the result of a baseline embedding model (denoted as **BowPL**) which computes the average embedding of words surrounding a mention to represent its context. The cosine similarity between the representation and candidate entity's embedding is used to replace the neural score $\sigma(m_i, e_i)$ in Equation 8. It is worth mentioning that the bag-of-word-embedding approach is also used in [4, 10, 47] to derive the mention context's representation.

We use the Gerbil benchmarking framework (Version 1.2.4) to evaluate the performance [45]. For fair comparison, we run Gerbil with a fixed local setting for all systems in comparison. Note that some of the results are slightly different from the ones reported in their original paper [14, 48]. The reason is that Gerbil (Version 1.2.4) has improved the entity matching and entity validation procedures to adapt to the knowledge base's changes over time.[2]

We evaluate the disambiguation results by widely used measures: Precision, Recall and F1. For all the measures, we report the micro-averaged score (*i.e.,* aggregated across mentions not documents), and refer the micro-averaged $F1$ as the main metric.

## 6.3 Result and Discussion

**Comparison with baselines.** Table 3 reports the micro-averaged $F1$ of NeuPL and all baseline methods in comparison. Here we also include two configurations of NeuPL and BowPL, named **NeuL** and **BowL** respectively, which perform entity linking purely based on local context matching, without collective linking. A mention is linked to the entity with the highest matching score by Equation 8. We make the following observations from the result.

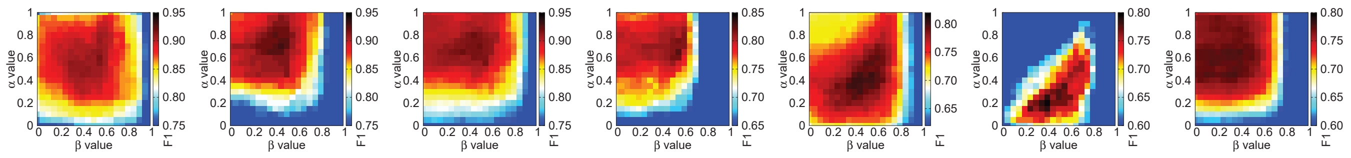
First, our proposed NeuPL achieves the best results on all 7 datasets. It outperforms the second best system DoSeR by 5% based on the averaged $F1$ (the last column in Table 3). DoSeR uses doc2vec [28] to model mention's local context. While doc2vec is simpler, it ignores the word order as well as the mention's location in the local context. NeuPL and NeuL implement two LSTMs to capture the positional information and incorporates attention mechanism. This explains why NeuL, even without collective linking, achieves comparable results with DoSeR and better results than PBoH. Additionally, it is not surprising that our proposed neural model NeuPL outperforms the simple bag-of-word-embedding method BowPL with large margin in two configurations (with and without pair-linking).

Second, PBoH performs better than the first three baselines but worse than the methods powered by neural networks *i.e.,* NeuPL, DoSeR, and NeuL. PBoH makes use of pairwise co-occurrence statistics of words and entities. The semantic similarity between different

[2] http://svn.aksw.org/papers/2016/ISWC_Gerbil_Update/public.pdf

Table 3: Micro-averaged $F1$ of all systems. The best results are in boldface and the second-best are underlined.

| System | Reuters128 | ACE2004 | MSNBC | DBpedia | RSS500 | KORE50 | Micro2014 | Average |
|---|---|---|---|---|---|---|---|---|
| AIDA [23] | 0.599 | 0.820 | 0.759 | 0.249 | 0.722 | 0.660 | 0.433 | 0.606 |
| Kea [40] | 0.654 | 0.796 | 0.854 | 0.736 | 0.709 | 0.620 | 0.639 | 0.715 |
| WAT [35] | 0.660 | 0.809 | 0.795 | 0.671 | 0.700 | 0.599 | 0.604 | 0.691 |
| PBoH [14] | 0.759 | 0.876 | 0.897 | 0.791 | 0.711 | 0.646 | 0.725 | 0.772 |
| DoSeR [48] | 0.873 | 0.921 | 0.912 | 0.816 | 0.762 | 0.550 | 0.756 | 0.798 |
| BowL | 0.793 | 0.896 | 0.823 | 0.780 | 0.708 | 0.419 | 0.725 | 0.735 |
| BowPL | 0.858 | 0.921 | 0.871 | 0.798 | 0.744 | 0.544 | 0.732 | 0.781 |
| **NeuL** | 0.869 | 0.906 | 0.904 | 0.807 | 0.770 | 0.551 | 0.766 | 0.796 |
| **NeuPL** | **0.916** | **0.929** | **0.918** | **0.828** | **0.800** | **0.794** | **0.776** | **0.851** |



(a) Reuters (b) ACE2004 (c) MSNBC (d) Dbpedia (e) RSS500 (f) KORE50 (g) Micro2014

Figure 3: Micro-average F1 performance of NeuPL with varying $\alpha$ and $\beta$ values (best viewed in color).

Table 4: Micro-averaged Precision, Recall, $F1$ score of NeuPL, and the optimal $\alpha$ and $\beta$ settings on each dataset.

| Data set | Precision | Recall | F1 | $\alpha$ | $\beta$ |
|---|---|---|---|---|---|
| Reuters128 | 0.924 | 0.908 | 0.916 | 0.60 | 0.55 |
| ACE2004 | 0.934 | 0.923 | 0.929 | 0.65 | 0.45 |
| MSNBC | 0.919 | 0.917 | 0.918 | 0.65 | 0.35 |
| DBpedia | 0.832 | 0.824 | 0.828 | 0.65 | 0.55 |
| RSS500 | 0.801 | 0.799 | 0.800 | 0.35 | 0.40 |
| KORE50 | 0.794 | 0.794 | 0.794 | 0.20 | 0.40 |
| Micro2014 | 0.783 | 0.770 | 0.776 | 0.45 | 0.35 |

words/entities is not well captured. On the contrary, NeuPL and DoSeR both utilize the continuous representation of word embeddings to improve semantic matching.

Third, among the 7 datasets, NeuPL performs slightly better than other state-of-the-art baselines on easy datasets (*e.g.,* DBpedia, ACE2004 and MSNBC) where the mentions in these datasets are relatively general and the contexts are clear. On the KORE50 dataset which demonstrates higher level of ambiguity in its mentions (*i.e.,* many are first names), NeuPL performs significantly better against baselines. It also show that Pair-Linking improves the linking results from 0.551 to 0.794, comparing NeuPL and NeuL.

**Hyperparameter study for $\alpha$ and $\beta$.** Recall that NeuPL requires two hyperparameters $\alpha$ and $\beta$, where $\alpha$ balances the contribution of prior probability and semantic matching in the local score computation (see Equation 8), and $\beta$ controls the contribution of semantic relatedness between entities in collective linking (see Equation 9).

In our experiments, we set $\alpha$ and $\beta$ through 5-fold cross validation. We observe that, on the same dataset, the behaviours of NeuPL are very consistent across different partitions. In fact, we obtain the same settings of $\alpha$ and $\beta$ through different partitions and list their values in the detailed results of NeuPL in Table 4.

The impacts of setting different $\alpha$ and $\beta$ values to the $F1$ of NeuPL on different datasets are plotted in Figure 3. Observe that, the best settings of $\beta$ are within a narrow range from 0.35 to 0.55 across all 7 datasets, which is also reflected in Table 4. In fact, a common setting of $\beta = 0.45$ on all the 7 datasets leads to comparable results as the ones reported in Table 4. This suggests that NeuPL is not sensitive to $\beta$ on different datasets.

The best settings of $\alpha$ are in the range of 0.20 to 0.60 on the 7 datasets. We observe that NeuPL prefers smaller $\alpha$ ($\leq 0.5$) on short texts (RSS500, KORE50, and Micro2014) than long news articles (Reuters128, ACE2004, MSNBC). The reasons are two fold. First, mentions from news articles in Reuters128, ACE2004, and MSNBC are usually general and less ambiguous than mentions from tweets (Micro2014) or short sentences (KORE50). Therefore, the prior probability $p(e|m)$ is given higher weights on these datasets. Second, general and less ambiguous mentions in a long document may appear in many different contexts and some of the contexts do not provide enough semantic information because the entire document provides enough background information to disambiguate them. On the other hand, short texts like RSS500 and KORE50 need to provide enough context information within the short text itself for good understanding.

**Collective linking.** We compare our Pair-Linking algorithm with the following state-of-the-art collective linking (CL) algorithms. For the evaluated CL algorithms, we use $\phi(m_i, e_i)$ computed by Equation 8 as local matching score, and $\psi(t_i, t_j)$ in Equation 9 as entity-relatedness score. In short, the CL algorithms are evaluated based on the output of our neural architecture,*i.e.,* NeuL.

- **Iterative Substitution** (Itr_Sub) [39] defines the global coherence $\psi(\Gamma)$ (see Equation 1) to be the sum of pair-wise relatedness between all pairs of the linked entities. To solve the objective function, an approximate solution is proposed by iteratively substituting an assignment $m_i \mapsto t_i$ with another mapping $m_i \mapsto t_j$ as long as it increases the objective score.

**Table 5: Micro-averaged $F1$ with different CL algorithms. The best scores are in boldface and the second-best are underlined.**

| NeuL + Collective Linking | Reuters128 | ACE2004 | MSNBC | DBpedia | RSS500 | KORE50 | Micro2014 | Average |
|---|---|---|---|---|---|---|---|---|
| NeuL + PL (*i.e.,* NeuPL) | 0.916 | **0.929** | **0.918** | 0.828 | 0.800 | **0.794** | 0.776 | **0.851** |
| NeuL + ItrSub | 0.906 | 0.916 | 0.913 | 0.828 | 0.794 | 0.627 | 0.759 | 0.821 |
| NeuPL + ItrSub | **0.918** | 0.920 | **0.918** | 0.828 | **0.804** | **0.794** | 0.762 | 0.849 |
| NeuL + FwBw | 0.905 | 0.912 | 0.909 | **0.837** | 0.782 | 0.745 | 0.777 | 0.838 |
| NeuL + DensSub | 0.892 | 0.898 | 0.897 | 0.776 | 0.800 | 0.767 | 0.755 | 0.826 |
| NeuL + PageRank | 0.892 | 0.912 | 0.908 | 0.831 | 0.763 | 0.488 | **0.784** | 0.797 |

**Table 6: Average time to link mentions in one document (in milisecond) for each of 7 benchmark datasets.**

| Collective Linking | Reuters128 | ACE2004 | MSNBC | DBpedia | RSS500 | KORE50 | Micro2014 |
|---|---|---|---|---|---|---|---|
| Pair-Linking (PL) | 1.139 | 1.527 | 54.463 | 1.512 | 0.027 | 0.358 | 0.130 |
| FwBw | 1.490 | 2.984 | 13.277 | 3.140 | 0.174 | 1.560 | 0.600 |
| ItrSub | 28.951 | 29.359 | 5121.481 | 13.435 | 0.134 | 1.281 | 0.737 |
| PL + ItrSub | 17.313 | 20.725 | 6161.708 | 13.249 | 0.137 | 1.477 | 0.765 |
| DensSub | 146.295 | 260.729 | 18382.563 | 160.228 | 1.824 | 19.290 | 9.135 |
| PageRank | 159.574 | 334.742 | 11476.841 | 279.725 | 6.929 | 116.282 | 51.571 |

In their original paper, mentions are first assigned to the candidates with highest local scores. We also evaluate another setting where the initial assignments are the results of our Pair-Linking. We refer this setting as NeuPL+Itr_Sub.

- **Forward-Backward** (FwBw) algorithm [2] limits the topical coherence to consider only the adjacent assignments of a mention and uses dynamic programming to derive the optimal assignments.
- **Densest Subgraph** (DensSub) [23] also uses a similar objective function as Itr_Sub. It applies dense subgraph algorithm to prune the mention-candidate graph into a much smaller subgraph. Afterwards, local search is performed on the subgraph.
- **Personalized PageRank** (PageRank) used by DoSeR [48]. It performs personalized PageRank on the mention-candidate graph and uses the stabilized scores for disambiguation.

Table 5 reports the micro-averaged $F1$ of NeuL with different linking algorithms. Among all settings evaluated, NeuPL and NeuPL+Itr_Sub achieve best performances on most datasets. The differences between NeuPL+Itr_Sub and NeuPL are negligible but NeuPL is much faster in terms of computation. The iterative substitution linking algorithm achieves better $F1$ when being initialized with Pair-Linking results (*i.e.,* NeuPL+Itr_Sub) compared to the version initialized with local scores (*i.e.,* NeuL+Itr_Sub). This result indicates that the results by Pair-Linking are closer to groundtruth. Observe that, even considering only adjacent assignments in coherence computation, FwBw achieves very good results compared with DensSub and PageRank which consider all assignments in a document. As shown in Table 3, no single collective linking algorithm achieves the best performance across all datasets. Therefore, modelling and solving topical coherence in entity disambiguation remains a challenging problem.

The running times of different collective linking algorithms are listed in Table 6. FwBw has the lowest time cost in worst case since it only considers adjacent mentions. Not surprisingly, Itr_Sub and PageRank are the slowest ones. While Itr_Sub requires multiple iterations to refine all assignments, PageRank iteratively operates on the mention-entity matrix for convergence. On the other hand, Pair-Linking only needs to traverse all possible pairs at most once. Furthermore, the worst case of Pair-Linking is the prerequisite of any graph-based algorithm (*e.g.,* PageRank and DensSub) for building the mention-entity graph.

**Table 7: $F1$ of neural network variants with and without PL.**

| Dataset | Full-model | | Single LSTM | | No Attention | |
|---|---|---|---|---|---|---|
| | w/ PL | w/o PL | w/ PL | w/o PL | w/ PL | w/o PL |
| Reuters128 | 0.916 | 0.869 | 0.900 | 0.846 | 0.874 | 0.832 |
| ACE2004 | 0.929 | 0.906 | 0.927 | 0.898 | 0.919 | 0.894 |
| MSNBC | 0.918 | 0.904 | 0.913 | 0.901 | 0.909 | 0.895 |
| DBpedia | 0.828 | 0.807 | 0.798 | 0.792 | 0.807 | 0.795 |
| RSS500 | 0.800 | 0.770 | 0.792 | 0.772 | 0.750 | 0.721 |
| KORE50 | 0.794 | 0.551 | 0.732 | 0.572 | 0.676 | 0.572 |
| Micro2014 | 0.776 | 0.766 | 0.777 | 0.775 | 0.773 | 0.769 |
| Average | 0.851 | 0.796 | 0.834 | 0.794 | 0.815 | 0.783 |

Empirical results show that Pair-Linking is indeed fast, partially due to "early stop" in implementation. Since only few pairs of assignments dominate the Pair-Linking scores, a large number of pairs are ignored by early stop. The running time of Pair-Linking is even faster than FwBw on 6 out of 7 datasets, making Pair-Linking the most effective and efficient linking algorithm.

Consider the most time consuming dataset MSNBC, where on average there are 32.9 mentions per document. Among all algorithms that consider all mentions in the same document for collective linking, Pair-Linking is nearly 100 times faster than the next efficient algorithm Itr_Sub, shown in Table 6. FwBw is faster than PL but it does not consider all mentions in the same document for collective linking, and its linking accuracy is poorer than PL (see Table 5).

**Neural network variants.** In our neural network model, we use two LSTMs to encode positional information of words and mention and exploit attention mechanism. To evaluate their impact on disambiguation accuracy, we compare the network with another two variants. The first variant uses a single LSTM to encode the local context (*e.g.,* both the left and right side contexts) and the second variant does not use attention mechanism.

The micro-averaged $F1$ of the full model and the two variants are reported in Table 7, with two settings *i.e.,* with (w) and without (w/o) Pair-Linking. The result shows that the full model consistently outperforms both variants on all settings across all datasets. Particularly, removing the attention mechanism causes $F1$ to drop from 0.851 to 0.815.

## 7 CONCLUSIONS

In this paper, we present a deep neural network that utilizes LSTM and attention mechanism to model the semantic matching between mention context and target entity. The superior performance of our model hints that neural feature could significantly benefit entity disambiguation task as well as semantic matching task in general. Furthermore, our proposed Pair-Linking algorithm is shown to be among the fastest and most effective linking algorithm. Through extensive experiments, we show that our proposed system outperforms state-of-the-art solutions.

## ACKNOWLEDGEMENTS

This work was supported by Singapore Ministry of Education Research Fund MOE2014-T2-2-066.

## REFERENCES

[1] Ayman Alhelbawy and Robert J. Gaizauskas. 2014. Graph Ranking for Collective Named Entity Disambiguation. In *ACL Volume 2: Short Papers*. 75–80.

[2] Steve Austin, Richard Schwartz, and Paul Placeway. 1991. The forward-backward search algorithm. In *IEEE ICASSP*. 697–700.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014).

[4] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and Space-Efficient Entity Linking for Queries. In *WSDM*. 179–188.

[5] Razvan C. Bunescu and Marius Pasca. 2006. Using Encyclopedic Knowledge for Named entity Disambiguation. In *EACL*.

[6] Rui Cai, Houfeng Wang, and Junhao Zhang. 2015. Learning Entity Representation for Named Entity Disambiguation. In *CCL and NLP-NABD*. 267–278.

[7] Rudi Cilibrasi and Paul M. B. Vitányi. 2007. The Google Similarity Distance. *IEEE TKDE* 19, 3 (2007), 370–383.

[8] Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL*. 708–716.

[9] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity Disambiguation for Knowledge Base Population. In *COLING*. 277–285.

[10] Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity Disambiguation by Knowledge and Text Jointly Embedding. In *CoNLL*.

[11] Paolo Ferragina and Ugo Scaiella. 2012. Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *IEEE Software* 29, 1 (2012), 70–75.

[12] Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing Semantic Similarity for Entity Linking with Convolutional Neural Networks. In *NAACL HLT*. 1256–1261.

[13] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[14] Octavian-Eugen Ganea, Marina Ganea, Aurélien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic Bag-Of-Hyperlinks Model for Entity Linking. In *WWW*. 927–938.

[15] Swapna Gottipati and Jing Jiang. 2011. Linking Entities to a Knowledge Base with Query Expansion. In *EMNLP*. 804–813.

[16] Zhaochen Guo and Denilson Barbosa. 2014. Robust Entity Linking via Random Walks. In *CIKM*. 499–508.

[17] Ben Hachey, Will Radford, and James R. Curran. 2011. Graph-Based Named Entity Linking with Wikipedia. In *WISE*. 213–226.

[18] Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *SIGIR*. 765–774.

[19] Xianpei Han and Jun Zhao. 2009. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *CIKM*. 215–224.

[20] Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning Entity Representation for Entity Disambiguation. In *ACL Volume 2: Short Papers*. 30–34.

[21] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *NIPS*. 1693–1701.

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[23] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *EMNLP*. 782–792.

[24] Heng Ji and Ralph Grishman. 2011. Knowledge Base Population: Successful Approaches and Challenges. In *ACL*. 1148–1158.

[25] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. *JMLR* (2015).

[26] Saurabh Kataria, Krishnan S. Kumar, Rajeev Rastogi, Prithviraj Sen, and Srinivasan H. Sengamedu. 2011. Entity disambiguation with hierarchical topic models. In *SIGKDD*. 1037–1045.

[27] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of Wikipedia entities in web text. In *SIGKDD*. 457–466.

[28] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*. 1188–1196.

[29] Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design Challenges for Entity Linking. *TACL* 3 (2015), 315–328.

[30] Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proc CIKM*. 233–242.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[32] David N. Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *CIKM*. 509–518.

[33] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *TACL* 2 (2014), 231–244.

[34] Aasish Pappu, Roi Blanco, Yashar Mehdad, Amanda Stent, and Kapil Thadani. 2017. Lightweight Multilingual Entity Extraction and Linking. In *WSDM*.

[35] Francesco Piccinno and Paolo Ferragina. 2014. From TagME to WAT: a new entity annotator. In *ACM Workshop on Entity Recognition & Disambiguation*. 55–62.

[36] Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *ACL*. 1375–1384.

[37] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP*. 379–389.

[38] Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *IEEE TKDE* 27, 2 (2015), 443–460.

[39] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. LIEGE: : link entities in web lists with knowledge base. In *KDD*. 1424–1432.

[40] Nadine Steinmetz and Harald Sack. 2013. Semantic Multimedia Information Retrieval Based on Contextual Descriptions. In *ESWC*. 382–396.

[41] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *NIPS*. 2440–2448.

[42] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation. In *IJCAI*. 1333–1339.

[43] Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based Deep Learning Models for non-factoid answer selection. *CoRR* abs/1511.04108 (2015).

[44] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective LSTMs for Target-Dependent Sentiment Classification. In *COLING*. 3298–3307.

[45] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. 2015. GERBIL: General Entity Annotator Benchmarking Framework. In *WWW*. 1133–1143.

[46] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph and Text Jointly Embedding.. In *EMNLP*. 1591–1601.

[47] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *CoNLL*.

[48] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2016. Robust and Collective Entity Disambiguation through Semantic Embeddings. In *SIGIR*. 425–434.