An Attention Factor Graph Model for Tweet Entity Linking

Chenwei Ran
Department of Computer Science and Technology
Tsinghua University
rcw14@mails.tsinghua.edu.cn

Wei Shen*
College of Computer and Control Engineering
Nankai University
shenwei@nankai.edu.cn

Jianyong Wang†
Department of Computer Science and Technology
Tsinghua University
jianyong@tsinghua.edu.cn

ABSTRACT

The rapid expansion of Twitter has attracted worldwide attention. With more than 500 million tweets posted per day, Twitter becomes an invaluable information and knowledge source. Many Twitter related tasks have been studied, such as event extraction, hashtag recommendation, and topic detection. A critical step in understanding and mining information from Twitter is to disambiguate entities in tweets, i.e., tweet entity linking. It is a challenging task because tweets are short, noisy, and fresh. Many tweet-specific signals have been found to solve the tweet entity linking problem, such as user interest, temporal popularity, location information and so on. However, two common weaknesses exist in previous work. First, most proposed models are not flexible and extendable to fit new signals. Second, their scalability is not good enough to handle the large-scale social network like Twitter. In this work, we formalize the tweet entity linking problem into a factor graph model which has shown its effectiveness and efficiency in many other applications. We also propose selective attention over entities to increase the scalability of our model, which brings linear complexity. To adopt the attention mechanism in the factor graph, we propose a new type of nodes called pseudo-variable nodes to solve the asymmetry attention problem caused by the undirected characteristic of the factor graph. We evaluated our model on two different manually annotated tweet datasets. The experimental results show that our model achieves better performance in terms of both effectiveness and efficiency compared with the state-of-the-art approaches.

CCS CONCEPTS

• **Information systems → Information extraction**;

KEYWORDS

Twitter, knowledge graph, entity linking, factor graph model, attention-based model

*Corresponding author
†Also with Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University.

ACM Reference Format:
Chenwei Ran, Wei Shen, and Jianyong Wang. 2018. An Attention Factor Graph Model for Tweet Entity Linking. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186012

1 INTRODUCTION

The rapid expansion of Twitter[1], an online social networking and microblogging service, has attracted worldwide attention. While tweets (messages posted in Twitter) are restricted to 140 characters, there are more than 330 million monthly active users and 500 million tweets per day about topics ranging from daily life to breaking news in Twitter. Researchers also take note of this invaluable information and knowledge source. Many Twitter related tasks have been studied, such as event extraction [14, 24], hashtag recommendation [6, 13], and topic detection [19].

A critical step in understanding and mining information from Twitter is disambiguating entities in tweets, i.e., linking mentions in tweets to their corresponding entities in the knowledge graph (e.g., Wikipedia). It is a challenging task. Characteristics of tweets can be summarized to three points [9]: 1. **short**, each tweet is restricted to 140 characters; 2. **noisy**, informal acronyms, spoken language writing style and typos are common in tweets; 3.**fresh**, the newly emerging entity-related information in tweets may have not been included in the knowledge graph. All these characteristics make it hard to compute the similarity between text around the mention and the document describing the entity, which is an important feature when linking entities in traditional document entity linking. It also means there are fewer entities in the same tweet (as shown in [27], each tweet averagely contains only 0.76 entities), while topical coherence between different entities within the same tweet is another important feature in previous work.

Although it seems hard to leverage these traditional features in tweet entity linking, various novel tweet-specific signals have been found. For example, user interest is shown to be a powerful signal in tweet entity linking. Shen et al. [27] show that users in Twitter have their own interests. Thus, tweets from the same user have topical coherence which is helpful for tweet entity linking. Hua et al. [11] consider the scenario that some information seeking users (rather than content generators) post tweets rarely, and claim that users having followee-follower relationships also have similar interests, i.e., their tweets are topical coherent. Temporal signal is also helpful to achieve a better performance due to the fresh characteristic of tweets. For instance, Fang and Chang [5] and Hua et al. [11] both use the temporal popularity of entities in a specific time period instead of the general prior popularity. In addition, location

[1]https://twitter.com

information [5], hashtags [20] and extra posts [9] are also shown to be helpful signals for the tweet entity linking task.

However, two common weaknesses exist in these previous work. First, most of these work propose particular models, which are well-designed to leverage some signals. Therefore, these proposed models are not flexible and extendable to fit new signals. That is to say, it is hard to combine various effective signals mentioned above into a single model. Second, most of these proposed models are not scalable. As there are hundred millions of tweets posted every day, scalability is very essential for a tweet entity linking system. When topical coherence between different entities are considered, model learning and inference usually become NP-hard problem. Some work propose approximate algorithms to find the best configuration [17, 27], but they are easy to fall into a poor local optimum.

In order to overcome these weaknesses, we leverage a factor graph model to solve the tweet entity linking problem in this paper. The factor graph model has been successfully applied in many applications, such as knowledge base alignment [30], social relationship mining [29], and social influence analysis [28]. A factor graph model defines a factorization probabilistic distribution. It is an undirected graph comprising variable and factor nodes. Each factor node represents a function over the variables it connects to. As the factor function can be defined over arbitrary sets of variables, we could use almost every signals mentioned above as features. The factorization form of the probabilistic distribution makes the learning and inference in a factor graph model effective and efficient. The most difficult part of the learning and inference in a factor graph model is to calculate marginal probabilities. It is intractable because the graphical structure can be arbitrary and may contain cycles. Loopy belief propagation [23] is a commonly used approximate algorithm to solve this problem. Empirically it can get a great approximation and has polynomial complexity in proportion to the number and average degree of variable nodes.

We emphasize the importance of scalability for a tweet entity linking system. The polynomial complexity when learning and inferring in a factor graph model is efficient for many applications, but it is not good enough for a tweet entity linking system. Here we explain the reason under the scenario only considering one user. User interest is one of the most important signals when disambiguating entities from one user, which means we should measure the topical coherence between every two entities from the same user. When we apply the factor graph model in the tweet entity linking problem, each mention has a corresponding entity variable node in the factor graph. If we consider the topical coherence between every two entities, there are factor nodes connecting every two entity variable nodes. It makes the average degree of variable nodes equal to the number of variable nodes. In other words, the complexity when learning and inferring will be in proportion to the square of the average number of mentions extracted from one user, which is unacceptable for a large-scale social network like Twitter.

To achieve better scalability, we adopt the attention mechanism in the factor graph model. We argue that it is unnecessary to measure the topical coherence between every two entities from the same user. When we disambiguate an entity, we only need to consider the topical coherence between it and the entities it should pay attention to. A remaining problem is how to handle the asymmetry attention in the factor graph model, and we will solve it using the pseudo-variable nodes.

Contributions. The main contributions of this paper are summarized as follows.

- We leverage a factor graph model to deal with the task of tweet entity linking. It is effective, efficient, and flexible enough to combine many different signals together.
- To the best of our knowledge, it is the first time that the attention mechanism is adopted in the factor graph model. With the proposed selective attention over entities, our attention factor graph model can reach linear complexity. We also propose a new type of nodes called pseudo-variable nodes to solve the asymmetry attention problem.
- To verify the effectiveness and efficiency of our model, we evaluated it over two different tweet datasets. The experimental results show that the proposed model clearly achieves better performance than other state-of-the-art competitors.

The rest of this paper is organized as follows. In the next section, we introduce related work. In Section 3, we give the formal definition of the tweet entity linking problem and introduce notations used in this paper. In Section 4, we present a general factor graph model for tweet entity linking and its learning and inference procedures. Then in Section 5, we explain how to add the attention mechanism into the factor graph. We evaluate our model in Section 6. Finally in Section 7 we give a conclusion of this work.

2 RELATED WORK

2.1 Tweet Entity Linking

In the early time, a large amount of research on entity linking [1, 3, 10, 25, 26] have been conducted over news documents. However, these approaches do not work well when they are applied to the tweet data as tweets are short, noisy, and fresh. Researchers turn to find new signals from rich meta-data of tweets. Among various tweet-specific signals, user interest may be the most powerful one. Shen et al. [27] found that the entities from the same user are topical coherent, as users have their own underlying topic distribution based on their interest. Hua et al. [11] further leveraged the followee-follower relationship between users to avoid the difficulty and inaccuracy of modeling user interest from limited amount of tweets. Time stamp is another useful signal for tweet entity linking. Fang and Chang [11] and Hua et al. [5] both found that the temporal popularity of entities is a better feature than the general prior popularity because it can capture the recency characteristic of tweets. They computed the temporal popularity with the occurrence number of entities from the historical tweet data. As the entity mentioned in tweets is unknown due to the ambiguity of mentions, they adopted a variant of the EM algorithm. Another challenge in the calculation of the temporal signal is the sparsity of data. They smoothed the temporal popularity over time binning and entities respectively. Besides, there are many other useful signals such as locations [5], attached hashtags [20] and extra posts [9]. Some studies [5, 8, 31] also found that merging the tasks of mention detection and entity disambiguation into a joint task can achieve improvements. Two common weaknesses exist in these work. First, most of these models are not flexible and extendable

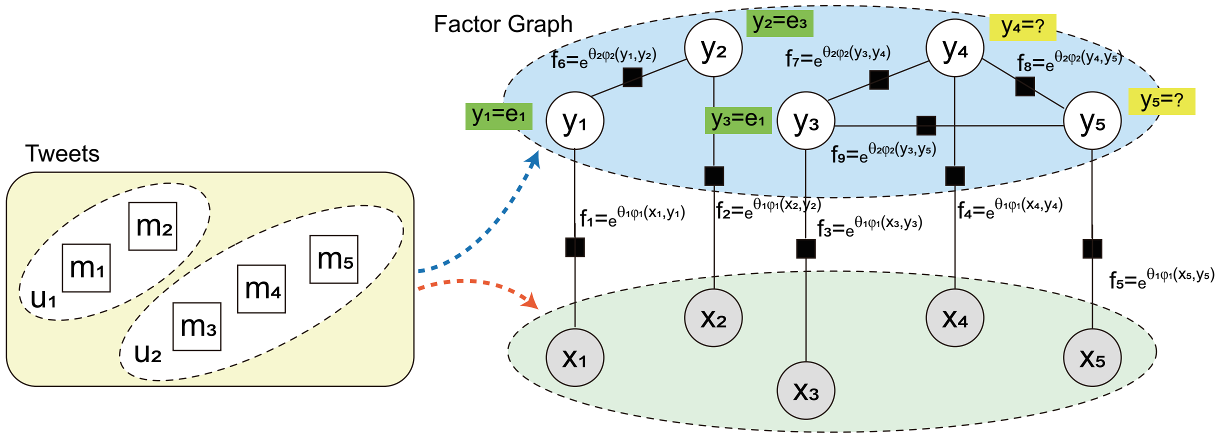


Figure 1: The factor graph model for tweet entity linking

for new features. Second, most of them cannot scale up well to a large social network such as Twitter.

2.2 Factor Graph Model

The factor graph model has been successfully applied in many applications for large-scale data, such as knowledge base alignment [30], social relationship mining [29] and social influence analysis [28]. Tang et al. [29] studied the social relationship classification problem on a publication dataset. Wang et al. [30] studied the cross-lingual knowledge linking problem on a dataset constructed from Wikipedia. The datasets they used are both large-scale. In this paper, we apply the factor graph model to the tweet entity linking problem.

2.3 Attention-based Model

The idea of selective attention is inspired by Lin et al. [16]. They studied the relation extraction task using a sentence-level convolutional neural network. They proposed the selective attention over instances to overcome the wrong labeling problem in distant supervision. Besides, the attention-based model adopted in deep neural network has been applied to various areas such as speech recognition [2] and image classification [22]. To the best of our knowledge, this is the first time that the attention mechanism is adopted in the factor graph model.

3 PRELIMINARIES AND NOTATIONS

In this section, we first briefly introduce some basic concepts and define the task of tweet entity linking. Then we present notations employed in the remaining of the paper.

Tweet. *Tweets are the data source of the tweet entity linking task. The collection of tweets is denoted by T and the collection of users posting tweets is denoted by U.*

The main characteristic of tweets is the restriction of 140 characters which is different from general Web documents. Nonetheless, tweets have rich meta-data including user information, retweet (reposting of a tweet) relationships, posting times and locations, attached hashtags, images, and URLs. We use the content of tweets and the corresponding user information in this work, while other signals could be easily added to our model which will be discussed in details in the next section. For the tweet collection T, we use the lowercase of its denotation to represent an element in it (i.e., $t \in T$) and the subscript to index the element in it (i.e., t_i means the i-th tweet in T). We also represent the element of other collections in this paper following the same way.

Mention. *A mention is a textual phrase referring to some entity extracted from the content of tweets. The collection of mentions is denoted by M.*

Entity. *An entity is a unique real-world object. The collection of entities is denoted by E.*

Although an entity is unique, it may have many different surface forms. For example, the New York City can also be called "Big Apple". In the meanwhile, the same textual phrase may refer to different entities. For instance, "Jordan" could refer to the brand Air Jordan, the basketball player Michael Jordan, or the country Jordan. Thus, there is a many-to-many correspondence between mentions and entities. However, with certain context, a mention has its clear and unique semantically corresponding entity. To find the correct target entity, tweet entity linking is required.

Knowledge Graph. *A knowledge graph is a machine-readable set of knowledge.*

Here knowledge refers to entities, their semantic categories and attributes, and the relationships between entities. The knowledge graph we adopt in this paper is Wikipedia[2].

Tweet Entity Linking. *Given a mention $m \in M$ extracted from a tweet $t \in T$, it could refer to different entities. We call them candidate entities of mention m and denote them as $E_m \subseteq E$. The goal of tweet entity linking is to identify the semantically corresponding target entity $e \in E_m$ for the mention m with the context of m and the meta-data of tweet t.*

A typical entity linking system consists of three modules: mention detection, candidate generation, and candidate ranking. In this

[2]https://en.wikipedia.org

paper we focus on the candidate ranking task and take the detected mentions and the candidate entities for each mention as input. The task of mention detection (also named entity recognition) for tweets has been studied by several work [4, 15, 18]. The candidate entities of a mention can be collected from entity pages, redirect pages, disambiguation pages, and anchor phrases in Wikipedia articles [26]. Specially, mention m may be unlinkable, i.e., the corresponding entity of mention m does not exist in the knowledge graph. Some methods link such a mention to a special denotation NIL. In this paper, we assume that all the detected mentions are linkable with Wikipedia. The method for unlinkable mention detection is left for future research.

Factor Graph. *A factor graph can be represented as $G = (V, F, D)$, where V is the set of variable nodes, F is the set of factor nodes, and D is the set of edges.*

In a factor graph, a variable node $v_i \in V$ represents a random variable, and a factor node $f_j \in F$ represents a function $f_j(S_j)$ where $S_j \subseteq V$ is a set of variable nodes. There is an undirected edge between a factor node f_j and each variable node $v_k \in S_j$. The two types of nodes in a factor graph form a bipartite and undirected graph.

An illustration of formalizing the tweet entity linking problem into a factor graph model is shown in Fig. 1. Under the scenario of tweet entity linking, all variable nodes V in a factor graph G is divided into two subsets X and Y (i.e., $V = X \cup Y$), corresponding to the observed and hidden variables respectively. A mention $m_i \in M$ is mapped to an observed variable x_i and a hidden variable y_i. Thus the factor graph G has $2 \cdot |M|$ variable nodes in total. x_i represents the mention m_i and its context. It has only one state as it is observed. y_i represents the semantically corresponding entity of m_i. It has $K_i = |E_{m_i}|$ possible states each of which represents a candidate entity that m_i may refer to. We also call x as mention variable and y as entity variable in this paper. For learning and inference, the correct corresponding entity of some mention is labeled. Thus Y can be further divided into two subsets Y^L and Y^U (i.e., $Y = Y^L \cup Y^U$), corresponding to the labeled and unlabeled hidden variables respectively.

When learning and inferring with the factor graph model, an important algorithm is the loopy belief propagation algorithm. This algorithm involves passing messages on the factor graph. We denote the message from variable node v_i to factor node f_j as $\mu_{i \to j}(v_i)$. There are K_i possible states that variable v_i can take, thus the message results in a vector of length K_i. Similarly, the message from factor node f_j to variable node v_i is denoted as $\lambda_{j \to i}(v_i)$ which results in a vector of length K_i as well.

We propose a new type of nodes called pseudo-variable nodes for the factor graph in this paper. In our model, each hidden variable has a corresponding pseudo-variable node. We denote the corresponding pseudo-variable node of hidden variable y_i as y'_i.

All notations we use in this paper are summarized in Table 1.

4 FACTOR GRAPH MODEL

In this section, we begin with the description of the model features we use. Then we explain the learning and inference algorithm for the factor graph model in details.

Table 1: Summary of notations

Notation	Definition
T	A tweet collection
U	A user collection
M	A mention collection extracted from tweets T
$M_u \subseteq M$	The mentions extracted from tweets posted by a user u
E	The entity collection in a knowledge graph
$E_m \subseteq E$	The candidate entities of a mention m
$G = (V, F, D)$	A factor graph
$V = X \cup Y$	Variable nodes
F	Factor nodes
D	Edges
$S_j \subseteq V$	The variable nodes connecting to the factor f_j
X	Observed variables
$Y = Y^L \cup Y^U$	Hidden variables
K_i	The number of y_i's possible states
Y^L	The hidden variables with label
Y^U	The hidden variables without label
$\mu_{i \to j}(v_i)$	The message from the variable v_i to the factor f_j
$\lambda_{j \to i}(v_i)$	The message from the factor f_j to the variable v_i
Θ	Weighting parameters
Φ	Feature functions
y'_i	The corresponding pseudo-variable node of the hidden variable y_i
$\mathcal{R}_i(y_j)$	potential topical coherence between the variable y_i and y_j

4.1 Features

The factor functions in a factor graph can be instantiated in different ways. In this paper, we use exponential-linear functions. Specifically, we define the factor function f_j as:

$$f_j(S_j) = exp\{\Theta^T \Phi(S_j)\} \tag{1}$$

where $\Theta = (\theta_1, \theta_2, ...)$ is a weighting vector and Φ is a vector of feature functions. As we mentioned before, the observed variable x_i corresponds to the mention m_i and the hidden variable y_i corresponds to the entity that m_i semantically refers to. In the following we use the terms mention x_i and entity y_i for simplicity. In this paper, we consider two feature functions:

(1) Prior popularity. We leverage the anchor links in Wikipedia to compute the popularity of the entity y_i given the mention x_i and define the first feature function as:

$$\varphi_1(x_i, y_i) = \frac{count(x_i, y_i)}{count(x_i)} \tag{2}$$

where $count(x_i)$ represents the number of mention x_i occurring as the surface form of an anchor link in Wikipedia and $count(x_i, y_i)$ represents the number of anchor links with the surface form x_i pointing to the entity y_i.

(2) Topical coherence based on user interest. As shown in previous work, mentions from the same user have topical coherence. We adopt the Wikipedia Link-based Measure(WLM) described in [21] to measure the topical coherence between

entities. Given y_i and y_j whose corresponding mentions are extracted from the same user's tweets, we define the second feature function as:

$$\varphi_2(y_i, y_j) = 1 - \frac{log(max(|A_{y_i}|, |A_{y_j}|) - log(|A_{y_i} \cap A_{y_j}|)}{log(|A|) - log(min(|A_{y_i}|, |A_{y_j}|))} \quad (3)$$

where A is the collection of all Wikipedia articles and A_y is the collection of articles containing a link to entity y.

Although we only consider two feature functions in this paper, we claim that our model is flexible and extendable for new features. For example, if we want to leverage the temporal popularity, we can add a new feature function with regard to x_i and y_i to capture this signal. If we want to leverage the topical coherence based on hashtags, we can add a new feature function with regard to y_i and y_j whose corresponding mention are extracted from tweets having the same attached hashtag. More generally, the feature function can be defined over arbitrary sets of variables while the learning and inference algorithm can still work.

4.2 Learning

Given a factor graph G, the joint distribution over hidden variables Y is defined as:

$$p(Y|G) = \frac{1}{Z}\prod_j f_j(S_j) = \frac{1}{Z}exp\{\Theta^T \sum_j \Phi(S_j)\} = \frac{1}{Z}exp\{\Theta^T \Psi\} \quad (4)$$

where $Z = \sum_Y exp\{\Theta^T\Psi\}$ is a special normalization factor with zero nodes, $\Psi = \sum_j \Phi(S_j)$ is the aggregation of feature functions over all factor nodes.

Learning a factor graph model is to estimate an optimum parameter configuration Θ^*, so that the log-likelihood of given labeled data is maximized. The log-likelihood objective function is defined as:

$$\begin{aligned} O(\Theta) &= log\, p(Y^L|G) = log \sum_{Y|Y^L} \frac{1}{Z} exp\{\Theta^T\Psi\} \\ &= log \sum_{Y|Y^L} exp\{\Theta^T\Psi\} - log \sum_Y exp\{\Theta^T\Psi\} \end{aligned} \quad (5)$$

where Y^L denotes the hidden variables with known labels and $Y|Y^L$ is a labeling configuration of Y inferred from Y^L. The gradient decent algorithm is a commonly used method to maximize the objective function. The gradient for parameters Θ is calculated as:

$$\begin{aligned} \frac{\partial O(\Theta)}{\partial\Theta} &= \frac{\partial log \sum_{Y|Y^L} exp\{\Theta^T\Psi\} - log\sum_Y exp\{\Theta^T\Psi\}}{\partial\Theta} \\ &= \frac{\sum_{Y|Y^L} exp\{\Theta^T\Psi\}\cdot\Psi}{\sum_{Y|Y^L} exp\{\Theta^T\Psi\}} - \frac{\sum_Y exp\{\Theta^T\Psi\}\cdot\Psi}{\sum_Y exp\{\Theta^T\Psi\}} \\ &= \mathbb{E}_{p_\Theta(Y|Y^L,G)}\Psi - \mathbb{E}_{p_\Theta(Y)}\Psi \end{aligned} \quad (6)$$

where $\mathbb{E}_{p_\Theta(Y|Y^L,G)}\Psi$ and $\mathbb{E}_{p_\Theta(Y)}\Psi$ are two expectations of Ψ with regard to the probabilistic distribution $p_\Theta(Y|Y^L, G)$ and $p_\Theta(Y)$ respectively. To obtain the expectation of Ψ, we need to calculate the marginal probabilities $p(y_i)$ and $p(y_i, y_j)$ and then sum over all hidden variables. Exact marginal probabilities of a factor graph is intractable to be calculated as the graphical structure of the factor graph can be arbitrary and may contain cycles. Loopy belief propagation (LBP for short) [23] is a commonly used algorithm to approximate marginal probabilities for a graphical model. We briefly introduce the LBP algorithm here.

The main idea of the LBP algorithm is passing messages on the factor graph. There are two types of messages:

(1) A message from the variable node v_i to the factor node f_j, denoted as:

$$\mu_{i\to j}(v_i) = \prod_{f_k: v_i\in S_k, f_k\neq f_j} \lambda_{k\to i}(v_i) \quad (7)$$

which means the message from v_i to f_j is the product of the messages from neighboring factor nodes of v_i except f_j. Specially, if f_j is the only neighboring factor node of v_i, the message is set to the uniform distribution.

(2) A message from the factor node f_j to the variable node v_i, denoted as:

$$\lambda_{j\to i}(v_i) = \sum_{S_j: v_i = x_i} f_j(S_j) \prod_{v_k\in S_j\backslash\{v_i\}} \mu_{k\to j}(v_k) \quad (8)$$

which means the message from f_j to v_i is the product of the factor f_j with the messages from all other connecting variable nodes, marginalized over all variable nodes except v_i (i.e., $S_j\backslash\{v_i\}$). Specially, if $S_j = \{v_i\}$, the message $\lambda_{j\to i}(v_i) = f_j(v_i)$.

If the variable v_i has a label v_k, the message it sends to any factor nodes is 1 if $v_i = v_k$ and 0 otherwise. As the messages are defined recursively, we initialize all messages to the uniform distribution and then update the messages according to Eq. 7 and Eq. 8 iteratively. With the final messages, we can compute the marginal probability of the variable v_i as:

$$p(v_i) \propto \prod_{f_k: v_i\in S_k} \lambda_{k\to i}(v_i) \quad (9)$$

We can also compute the marginal probability of the variable set S_j involved in the factor f_j as:

$$p(S_j) \propto f_j(S_j) \prod_{v_i\in S_j} \mu_{i\to i}(v_i) \quad (10)$$

Then we use a two-step variation of the LBP algorithm to obtain the gradient, one step for calculating expectation $\mathbb{E}_{p_\Theta(Y|Y^L,G)}\Psi$ and the other step for calculating expectation $\mathbb{E}_{p_\Theta(Y)}\Psi$. The learning algorithm also can be extended to a distributed learning version with a graph segmentation algorithm such as [12]. Readers could refer to [29] for details of the learning algorithm.

4.3 Inference

After we learned the optimal parameters Θ^*, we can infer the best label of each hidden variable (i.e., the entity that mention semantically refers to). The best label could be the state with the highest marginal probability according to Eq. 9. Alternatively, a better way is to find the best label as a whole, i.e., find a label configuration which maximizes the joint probability defined in Eq. 4:

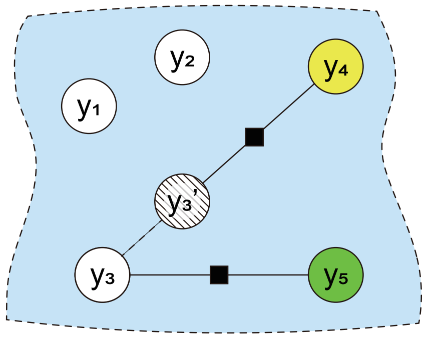


Figure 2: An example for the pseudo-variable node

$$Y^* = argmax_{Y|Y^L} p(Y|G, \Theta^*) \tag{11}$$

We can also use the LBP algorithm to solve this problem just with a little modification. The idea is simple: replace the $\sum$ operator with the *max* operator in the definition of messages. The modified algorithm is also named the max-sum algorithm.

4.4 Complexity Analysis

We analyze the time complexity of the learning and inference algorithm, more specifically, the LBP algorithm. The time cost of the LBP algorithm depends on the message passing process. In our model, the hidden variable corresponding to the mention from user u connects to $|M_u|$ factors. Thus in one iteration, the total number of message passing is

$$\sum_u |M_u|^2 \geq |U|(\frac{\sum_u M_u}{|U|})^2 = |U||\overline{M_u}|^2 \tag{12}$$

where $|\overline{M_u}|$ represents the average number of mentions extracted from the tweets posted by each user. Thus, time complexity of the learning and inference algorithm has a lower bound $\Omega(|U||\overline{M_u}|^2)$.

5 SELECTIVE ATTENTION OVER ENTITIES

In the previous section, we have discussed the intuitive idea of how to leverage the topical coherence based on user interest in the factor graph model, and the time complexity of the learning and inference algorithm has a lower bound $\Omega(|U||\overline{M_u}|^2)$. In this section, we argue that it is unnecessary to measure the topical coherence between every two entity variables corresponding to the same user. Here we consider the scenario that we disambiguate a mention m while the labeled entity of other mentions from the same user is already known. Obviously if a labeled entity is not topically coherent with any candidate entity of m, we do not need to consider this entity. In the factor graph model, that is to say, we do not need to connect these two entity variables with a factor node. Thus, we introduce the selective attention over entities for the factor graph model.

5.1 Attention Measure

When introducing the selective attention over entities for the factor graph model, the first challenge is that most entity variables in the factor graph model are unlabeled which make it hard to decide whether two entity variables are topically coherent. Therefore, we turn to measure the potential topical coherence between two entity variables. Given the entity variable y_i, we propose three ways to measure its potential topical coherence to another entity variable y_j:

Sum: The sum value of topical coherence between the entity variables y_i and y_j:

$$\mathcal{R}_i^1(y_j) = \sum_{y_i, y_j} \varphi_2(y_i, y_j) \tag{13}$$

Average: The average value of topical coherence between the entity variables y_i and y_j:

$$\mathcal{R}_i^2(y_j) = \frac{\sum_{y_i, y_j} \varphi_2(y_i, y_j)}{K_i \cdot K_j} \tag{14}$$

Max: The maximal value of topical coherence between the entity variables y_i and y_j:

$$\mathcal{R}_i^3(y_j) = Max_{y_j} \varphi_2(y_i, y_j) \tag{15}$$

Then we choose the top-$\mathcal{K}$ entity variables with respect to Eq. 13, Eq. 14 or Eq. 15 as the entity variables that y_i should pay attention to.

5.2 Pseudo-variable Node

The second challenge is the asymmetry attention problem, which means the situation that the variable y_i should pay attention to the variable y_j while y_j should not pay attention to y_i. Obviously, we can add a factor connecting to y_i and y_j if they should pay attention to each other. We can remove the factor connecting to y_i and y_j if they should not pay attention to each other. However, we cannot handle the asymmetry attention problem in the factor graph model directly because there is no directed edge in a factor graph.

We have a further discussion about what "attention" exactly means in the factor graph model here. When we say y_i should pay attention to y_j, we are expecting there is a factor whose neighbors are y_i and y_j, and its value will change when the state of y_j changes. When we say y_i should not pay attention to y_j, we are expecting there is no such a factor. Now consider the situation of asymmetry attention. It is required that there is a factor whose value will change when the state of y_j changes and there is no such a factor whose value will change when the state of y_i changes at the same time.

Thus, we propose a new type of nodes called pseudo-variable nodes. In the attention factor graph model, each hidden variable y has a corresponding pseudo-variable node y'. The pseudo-variable node y' is an observed node, whose state is set as:

$$y' = argmax_y P(y|Y^L, G) \tag{16}$$

Now we explain how to use the pseudo-variable node to solve the asymmetry attention problem with the example shown in Fig. 2. In this example, we assume that y_4 should pay attention to y_3 while y_3 should not pay attention to y_4. Thus, we add a factor connecting y_4 with y_3' instead of y_3. When the state of y_4 changes, the value of this factor will change. When the state of y_3 changes, the value of this factor will not change as the state of y_3' does not change.

A remaining problem is how to let the state of y' and the marginal probability $P(y|Y^L, G)$ affect each other. We propose a variation of the LBP algorithm to solve this problem. We first fix the states of the pseudo-variable nodes and approximate the marginal probabilities, and then we update the states of the pseudo-variable nodes iteratively. The proposed algorithm is depicted in Algo. 1.

Algorithm 1 Loopy Belief Propagation with Pseudo-variable Node

```
Input: factor graph $G = (V, F, D)$
Output: marginal probabilities
1: Initialize all messages with the uniform distributions, all
   pseudo-variable nodes with random labels
2: repeat
3:     Update all messages $\mu$ according to Eq. 7:
           $\mu_{i\to j}(v_i) = \prod_{f_k:v_i\in S_k, f_k\neq f_j} \lambda_{k\to i}(v_i)$
4:     Update all messages $\lambda$ according to Eq. 8:
           $\lambda_{j\to i}(v_i) = \sum_{S_j:v_i=x_i} f_j(S_j) \prod_{v_k\in S_j\setminus\{v_i\}} \mu_{k\to j}(v_k)$
5:     Update the states of all pseudo-variable nodes according to
       Eq. 16:
           $y' = argmax_y P(y|Y^L, G)$
6: until convergence
7: Calculate marginal probabilities according to Eq. 9 and Eq. 10
8: return marginal probabilities
```

5.3 Complexity Analysis

We analyze the time complexity of our attention factor graph model. As each variable should pay attention to at most $\mathcal{K}$ other variables, there are at most $\mathcal{K}$ factors that each variable connects to. Thus, time complexity of our attention factor graph model has an upper bound $O(\mathcal{K}|U||\overline{M_u}|)$. When $\mathcal{K} \leq |\overline{M_u}|$, the time cost of our model is technically less than the general factor graph model. As $\mathcal{K}$ can be considered as a constant value, the upper bound of the time complexity can also be represented as $O(\mathcal{K}|U||\overline{M_u}|) = O(|M|)$ which is a linear complexity with respect to the number of mentions in M.

6 EXPERIMENTS

To verify the effectiveness and efficiency of our attention factor graph model, we present a thorough experimental study in this section. We firstly describe the experimental setting in Section 6.1. Then we evaluate the effectiveness of our proposed model in Section 6.2 and the efficiency and scalability in Section 6.3 on two different tweet datasets.

6.1 Experimental Setting

6.1.1 Wikipedia Dataset. In our experiments, we use Wikipedia as our knowledge graph and regard the set of articles in Wikipedia as the entity collection. We downloaded the August 2017 version of English Wikipedia dump, which contains 5.5 million article pages, 7.9 million redirect pages and more than 78.3 million anchor links according to the Wikimedia Statistics project[3]. We also leverage the open source toolkit WikipediaMiner[4] to generate candidate entities for mentions and calculate features including prior popularity (Eq. 2) and topical coherence (Eq. 3).

Table 2: Statistics of tweet datasets

Dataset	#users	#tweets	#mentions	$\lvert\overline{M_u}\rvert$
SHEN13	20	3818	2203	110
FENG14	43	3458	1036	25

6.1.2 Tweet Dataset. The first tweet dataset we use is shared by Shen et al. in [27]. We call this dataset as SHEN13. Shen et al. randomly sampled 20 users from Twitter and annotated at most 200 recent tweets for each user. Finally, SHEN13 contains 3,818 tweets and 2,203 linkable mentions with annotation.

We also use the tweets collected by Feng et al. in [7] to create the second gold standard dataset. Feng et al. crawled 36.7 million tweets posted by 5.26 million users using Twitter's streaming API from October 2014 to December 2014. We call this dataset as FENG14. We randomly sampled a quarter of these tweets and then randomly chose 43 users each of which post more than one tweet. Finally, we obtained 3,458 tweets from 43 users. We manually annotated these tweets and obtained 1,036 linkable mentions with annotation.

The statistics of these two tweet datasets used in the experiments are shown in Table 2. Although both of them are created from tweets, there are some clear difference. Both the average numbers of mentions with respect to each tweet and each user in FENG14 are smaller than SHEN13.

6.1.3 Evaluation Method. Considering different attention measures, our attention factor graph model has several variations:

ATT-FULL: When we set the max attention number $\mathcal{K}$ to a large number, the attention factor graph model will fall back into the general factor graph model (i.e., the factor graph model without the selective attention over entities). We use this variation to evaluate the necessity of the attention mechanism.

ATT-SUM: It selects the attention targets according to the attention measure **Sum** (Eq. 13).

ATT-AVG: It selects the attention targets according to the attention measure **Avg** (Eq. 14).

ATT-MAX: It selects the attention targets according to the attention measure **Max** (Eq. 15).

We compare them with the following baseline methods in terms of accuracy and efficiency:

POP: It is the baseline method that only leverages the prior popularity to disambiguate entities. In this method, we choose the candidate entity which has the max prior popularity as the best label.

KAURI: It is the state-of-the-art method proposed in [27]. This model constructs a graph from all candidate entities. Each node in the graph represents a candidate entity and it has an initial interest score estimated by prior probability, context similarity, and topical coherence. Then a PageRank-like algorithm is proposed to

[3]https://stats.wikimedia.org/EN/TablesWikipediaEN.htm
[4]https://github.com/dnmilne/wikipediaminer

Table 3: Effectiveness performance with different methods

Method	SHEN13		FENG14	
	#Correct	Acc. (%)	#Correct	Acc. (%)
POP	1760	79.9	816	78.8
KAURI	1890	85.8	868	83.7
ATT-FULL	2016	91.5	915	88.3
ATT-SUM	2001	90.8	903	87.2
ATT-AVG	**2022**	**91.8**	912	88.0
ATT-MAX	2018	91.6	**917**	**88.5**

propagate the interest score between candidate entities. Finally the best label for a mention is the entity which has the maximum final interest score.

To quantitatively evaluate the proposed model, we consider two aspects: effectiveness and efficiency. For effectiveness evaluation, we consider two-fold cross-validation. More specifically, we construct the factor graph model with all mentions. Then we hide the labels of half nodes to learn the parameters and infer the best labels for the other part. We evaluate the methods mentioned above in terms of accuracy. For efficiency evaluation, we examine the execution time of the model learning.

Our attention factor graph model needs two inputs: the learning rate η and the max attention number $\mathcal{K}$. The learning rate η for two weighting parameters is set to $(0.01, 0.1)$. The max attention number $\mathcal{K}$ is set to 110 and 25 for SHEN13 and FENG14 respectively. We will also analyze the impact of $\mathcal{K}$ to our model's performance.

All the algorithms were implemented in Java with the support of JAMA[5] for fast matrix manipulations, and the experiments were carried out on a personal computer with Intel Xeon CPU E3-1230v3 (3.30GHz) and 16 GB memory. In this paper, we do not utilize the distributed learning and just consider single machine implementation.

6.2 Effectiveness

In this subsection, we study the effectiveness of our attention factor graph model under different configurations, and compare them with some baselines. The experimental result of effectiveness performance with different methods over two tweet datasets is shown in Table 3. We present both the number of correctly linked mentions and the accuracy. It can be observed that all the variations of our model outperform the baselines significantly. It can be seen that user interest is a very powerful signal, as it helps our model nearly achieve a 10% improvement in terms of accuracy compared with the baseline **POP** on both two datasets. Compared with the state-of-the-art method **KAURI**, our model can achieve a 6% improvement on the dataset SHEN13 and a 4.8% improvement on the dataset FENG14. As they both leverage the user interest signal, the reason of such significant improvements is that the factor graph model can capture the relatedness between different variables better and infer a better global optimal configuration. Besides, we can see that different variations of our attention factor graph model outperform the general factor graph model **ATT-FULL** slightly. Among them, the variation **ATT-AVG** achieves the best performance on the dataset SHEN13 and the variation **ATT-MAX** achieves the best performance on the dataset FENG14. However, the variation **ATT-SUM** results in a performance reduction on both datasets in comparison with the general factor graph model **ATT-FULL**.

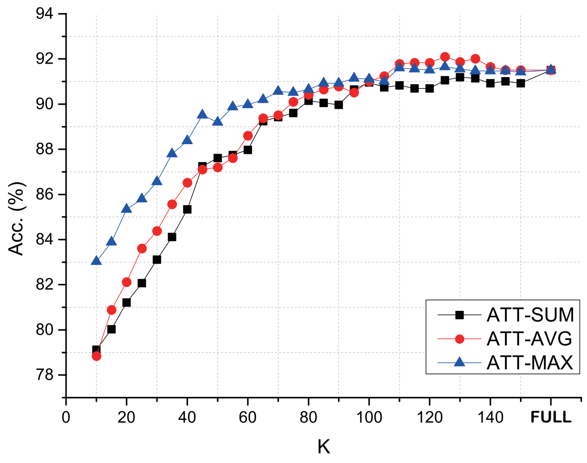
We further investigate the impact of different attention measures and different max attention number $\mathcal{K}$. The experimental result is shown in Fig. 3. As we mentioned before, the variations with different attention measures will fall back into the general factor graph model when $\mathcal{K}$ is set to a large number. It can be seen from Fig. 3 that the accuracy of the variations with different attention measures all converges to the accuracy of the variation **ATT-FULL** on both datasets when the max attention number $\mathcal{K}$ increases. We can have three main observations based on the experimental result. First, our attention factor graph model achieves a better performance when the max attention number $\mathcal{K}$ increases, as more attention targets bring more helpful information for disambiguation. Second, our attention factor graph model can still achieve a relatively good performance when $\mathcal{K}$ is small. Our model outperforms the baseline **KAURI** when $\mathcal{K}$ is set to 30 on the dataset SHEN13 and 10 on the dataset FENG14, as these two numbers are both smaller than the number $|\overline{M_u}|$ of two datasets respectively. Third, the variation **ATT-MAX** outperforms other two variations, especially when $\mathcal{K}$ is set to a small number. We demonstrate that it is an effective measure to find the mentions that we should pay attention to when we disambiguate a mention m.

6.3 Efficiency and Scalability

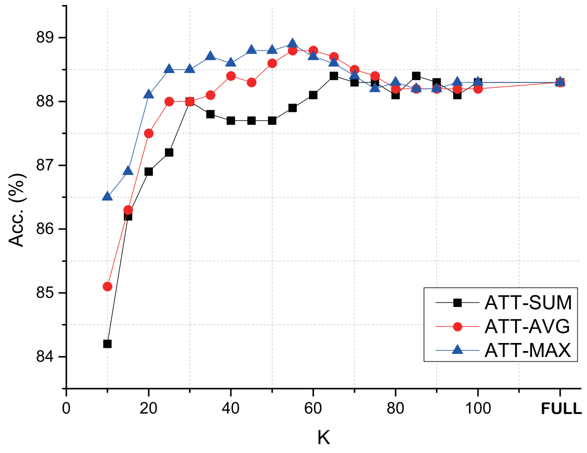
In this subsection, we study the efficiency and scalability of our attention factor graph model. As the time complexity of the attention factor graph model does not depend on the attention measure, we only compare the variation **ATT-MAX** with the general factor graph model **ATT-FULL**. More specifically, we report the average running time for one iteration in the learning algorithm with different experimental settings on both datasets. We first demonstrate the running time with different max attention number $\mathcal{K}$ in Fig. 4. As we mentioned before, the attention factor graph model will fall back into the general factor graph model when $\mathcal{K}$ is set to a large number. We can see that our attention factor graph model is much more efficient than the general factor graph model. The variation **ATT-MAX** with our settings of the max attention number $\mathcal{K}$ described before results in a 16.4% (89.2ms) and 43.6% (41.3ms) relative reduction of the running time compared with **ATT-FULL** on two datasets respectively. We can also observe that the average running time is approximately linear to the max attention number $\mathcal{K}$, which verifies the complexity analysis in Section 5.3. Thus, we can achieve a better efficiency by further decreasing the max attention number just with a slight decrease of accuracy.

Finally, we study the scalability of our attention factor graph model. We compare the average running time of different models with different size of datasets. The experimental result is shown in Fig. 5. It can be observed that the average running time of the model **ATT-MAX** is approximately linear to the size of datasets while the curve of the model **ATT-FULL** is more like the quadratic function. This observation verifies our complexity analysis in both Section 4.4 and Section 5.3. Thus, it can be said that our attention

[5]http://math.nist.gov/javanumerics/jama



(a) SHEN13



(b) FENG14

Figure 3: Effectiveness performance with different attention measures and different max attention number $\mathcal{K}$

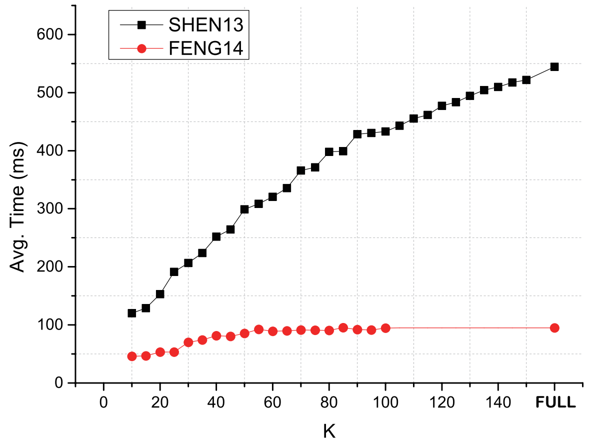


Figure 4: Efficiency performance with different max attention number $\mathcal{K}$

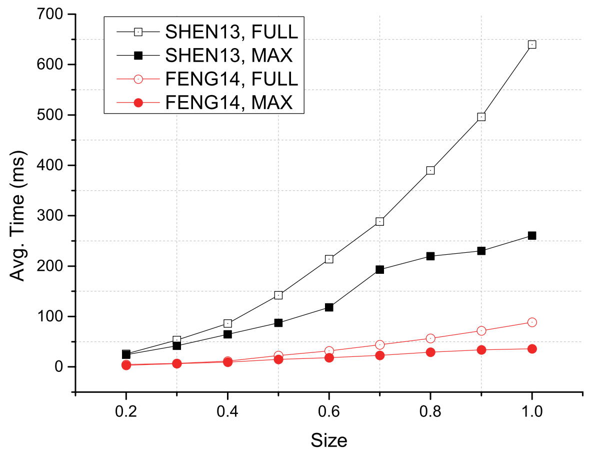


Figure 5: Scalability performance of different methods with different size of datasets

factor graph model can scale up better than the origin factor graph model in the large-scale network like Twitter.

7 CONCLUSIONS

In this paper, we study the problem of tweet entity linking. We formalize the tweet entity linking problem into a factor graph model, which has been successfully applied to many applications. We consider two features (prior popularity and topical coherence based on user interest) in our model. Nevertheless, we claim that our model is flexible and extendable for new features as factor nodes can be defined on arbitrary sets of variables while the learning and inference algorithm still work. To achieve better scalability, we propose to adopt the attention mechanism in the factor graph model. We propose a new type of nodes called pseudo-variable nodes to solve the asymmetry attention problem caused by the undirected characteristic of the factor graph. Experimental results verify the effectiveness and efficiency of our model on two tweet datasets.

There are some important future directions of this work. First, add more tweet-specific signals to the factor graph model. Second, formalize the problem of tweet entity linking into a semi-supervised framework. As we can only label extremely small part of data in Twitter, and Tang et al. [29] has shown the advantage of semi-supervised learning, a semi-supervised framework for tweet entity linking may be also helpful and promising. Finally, find a better attention measure. As the change of variables that need attention can influence the final results directly, a better attention measure is also very important.

ACKNOWLEDGMENTS

Chenwei Ran and Jianyong Wang were supported in part by National Natural Science Foundation of China under Grant No. 61532010 and National Basic Research Program of China (973 Program) under Grant No. 2014CB340505. Wei Shen was supported in part by National Natural Science Foundation of China under Grant No. 61502253.

REFERENCES

[1] Razvan C Bunescu and Marius Pasca. 2006. Using Encyclopedic Knowledge for Named entity Disambiguation.. In *EACL*, Vol. 6. 9–16.

[2] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-End Continuous Speech Recognition Using Attention-Based Recurrent NN: First Results. In *NIPS 2014 Workshop on Deep Learning*.

[3] Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

[4] Diego Marinho de Oliveira, Alberto HF Laender, Adriano Veloso, and Altigran S da Silva. 2013. FS-NER: a Lightweight Filter-Stream Approach to Named Entity Recognition on Twitter Data. In *Proceedings of the 22nd International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 597–604.

[5] Yuan Fang and Ming-Wei Chang. 2014. Entity Linking on Microblogs with Spatial and Temporal Signals. *Transactions of the Association for Computational Linguistics* 2 (2014), 259–272.

[6] Wei Feng and Jianyong Wang. 2012. Incorporating Heterogeneous Information for Personalized Tag Recommendation in Social Tagging Systems. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1276–1284.

[7] Wei Feng, Chao Zhang, Wei Zhang, Jiawei Han, Jianyong Wang, Charu Aggarwal, and Jianbin Huang. 2015. STREAMCUBE: Hierarchical Spatio-Temporal Hashtag Clustering for Event Exploration over the Twitter Stream. In *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 1561–1572.

[8] Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *Proceedings of The 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1020–1030.

[9] Yuhang Guo, Bing Qin, Ting Liu, and Sheng Li. 2013. Microblog Entity Linking by Leveraging Extra Posts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 863–868.

[10] 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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. ACL, 782–792.

[11] Wen Hua, Kai Zheng, and Xiaofang Zhou. 2015. Microblog Entity Linking with Social Temporal Context. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1761–1775.

[12] George Karypis and Vipin Kumar. 1995. METIS–Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0. (1995).

[13] Dominik Kowald, Subhash Chandra Pujari, and Elisabeth Lex. 2017. Temporal Effects on Hashtag Reuse in Twitter: A Cognitive-Inspired Hashtag Recommendation Approach. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1401–1410.

[14] Kathy Lee, Ashequl Qadir, Sadid A Hasan, Vivek Datla, Aaditya Prakash, Joey Liu, and Oladimeji Farri. 2017. Adverse Drug Event Detection in Tweets with Semi-Supervised Convolutional Neural Networks. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 705–714.

[15] Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. 2012. Twiner: Named Entity Recognition in Targeted Twitter Stream. In *Proceedings of the 35th international ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 721–730.

[16] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural Relation Extraction with Selective Attention over Instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. ACL.

[17] Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. 2013. Entity Linking for Tweets.. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. 1304–1311.

[18] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing Named Entities in Tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. ACL, 359–367.

[19] Michael Mathioudakis and Nick Koudas. 2010. Twittermonitor: Trend Detection over The Twitter Stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. ACM, 1155–1158.

[20] Edgar Meij, Wouter Weerkamp, and Maarten De Rijke. 2012. Adding Semantics to Microblog Posts. In *Proceedings of the fifth ACM International Conference on Web Search and Data Mining*. ACM, 563–572.

[21] David Milne and Ian H. Witten. 2008. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia links. In *WIKIAI*.

[22] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent Models of Visual Attention. In *Advances in Neural Information Processing Systems*. 2204–2212.

[23] Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 467–475.

[24] Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. 2015. Weakly Supervised Extraction of Computer Security Events from Twitter. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 896–905.

[25] Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a Knowledge Base: Issues, Techniques, and Solutions. *IEEE Transactions on Knowledge and Data Engineering* 27, 2 (2015), 443–460.

[26] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: Linking Named Entities with Knowledge Base via Semantic Knowledge. In *Proceedings of the 21st International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 449–458.

[27] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2013. Linking Named Entities in Tweets with Knowledge Base via User Interest Modeling. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 68–76.

[28] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social Influence Analysis in Large-Scale Networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 807–816.

[29] Wenbin Tang, Honglei Zhuang, and Jie Tang. 2011. Learning to Infer Social Ties in Large Networks. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 381–397.

[30] Zhichun Wang, Juanzi Li, Zhigang Wang, and Jie Tang. 2012. Cross-Lingual Knowledge Linking across Wiki Knowledge Bases. In *Proceedings of the 21st International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 459–468.

[31] Yi Yang and Ming-Wei Chang. 2015. S-MART: Novel Tree-based Structured Learning Algorithms Applied to Tweet Entity Linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. ACL, 504–513.