Entity Linking for Tweets

Xiaohua Liu[†], **Yitong Li**[‡], **Haocheng Wu**[♯], **Ming Zhou**[†], **Furu Wei**[†], **Yi Lu**[§]
[†]Microsoft Research Asia, Beijing, 100190, China
[‡]School of Electronic and Information Engineering
Beihang University, Beijing, 100191, China
[♯]University of Science and Technology of China
No. 96, Jinzhai Road, Hefei, Anhui, China
[§]School of Computer Science and Technology
Harbin Institute of Technology, Harbin, 150001, China
[†]{xiaoliu, mingzhou, fuwei}@microsoft.com
[‡]tong91222@126.com [♯]v-haowu@microsoft.com [§]v-y@microsoft.com

Abstract

We study the task of entity linking for tweets, which tries to associate each mention in a tweet with a knowledge base entry. Two main challenges of this task are the dearth of information in a single tweet and the rich entity mention variations. To address these challenges, we propose a collective inference method that simultaneously resolves a set of mentions. Particularly, our model integrates three kinds of similarities, i.e., mention-entry similarity, entry-entry similarity, and mention-mention similarity, to enrich the context for entity linking, and to address irregular mentions that are not covered by the entity-variation dictionary. We evaluate our method on a publicly available data set and demonstrate the effectiveness of our method.

1 Introduction

Twitter is a widely used social networking service. With millions of active users and hundreds of millions of new published tweets every day[1], it has become a popular platform to capture and transmit the human experiences of the moment. Many tweet related researches are inspired, from named entity recognition (Liu et al., 2012), topic detection (Mathioudakis and Koudas, 2010), clustering (Rosa et al., 2010), to event extraction (Grinev et al., 2009).

In this work, we study the entity linking task for tweets, which maps each entity mention in a tweet to a unique entity, i.e., an entry ID of a knowledge base like Wikipedia. Entity linking task is generally considered as a bridge between unstructured text and structured machine-readable knowledge base, and represents a critical role in machine reading program (Singh et al., 2011). Entity linking for tweets is particularly meaningful, considering that tweets are often hard to read owing to its informal written style and length limitation of 140 characters.

Current entity linking methods are built on top of a large scale knowledge base such as Wikipedia. A knowledge base consists of a set of entities, and each entity can have a variation list[2]. To decide which entity should be mapped, they may compute: 1) the similarity between the context of a mention, e.g., a text window around the mention, and the content of an entity, e.g., the entity page of Wikipedia (Mihalcea and Csomai, 2007; Han and Zhao, 2009); 2) the coherence among the mapped entities for a set of related mentions, e.g, multiple mentions in a document (Milne and Witten, 2008; Kulkarni et al., 2009; Han and Zhao, 2010; Han et al., 2011).

Tweets pose special challenges to entity linking. First, a tweet is often too concise and too noisy to provide enough information for similarity computing, owing to its short and grass root nature. Second, tweets have rich variations of named entities[3], and many of them fall out of the scope of the existing dictionaries mined from Wikipedia (called OOV mentions hereafter). On

[1]http://siteanalytics.compete.com/twitter.com/

[2]Entity variation lists can be extracted from the entity resolution pages of Wikipedia. For example, the link "http://en.wikipedia.org/wiki/Svm" will lead us to a resolution page, where "Svm" are linked to entities like "Space vector modulation" and "Support vector machine". As a result, "Svm" will be added into the variation lists of "Space vector modulation" and "Support vector machine" , respectively.

[3]According to Liu et al. (2012), on average a named entity has 3.3 different surface forms in tweets.

the other hand, the huge redundancy in tweets offers opportunities. That means, an entity mention often occurs in many tweets, which allows us to aggregate all related tweets to compute mention-mention similarity and mention-entity similarity.

We propose a collective inference method that leverages tweet redundancy to address those two challenges. Given a set of mentions, our model tries to ensure that similar mentions are linked to similar entities while pursuing the high total similarity between matched mention-entity pairs. More specifically, we define local features, including context similarity and edit distance, to model the similarity between a mention and an entity. We adopt in-link based similarity (Milne and Witten, 2008), to measure the similarity between entities. Finally, we introduce a set of features to compute the similarity between mentions, including how similar the tweets containing the mentions are, whether they come from the tweets of the same account, and their edit distance. Notably, our model can resolve OOV mentions with the help of their similar mentions. For example, for the OOV mention "LukeBryanOnline", our model can find similar mentions like "TheLukeBryan" and "LukeBryan". Considering that most of its similar mentions are mapped to the American country singer "Luke Bryan", our model tends to link "LukeBryanOnline" to the same entity.

We evaluate our method on the public available data set shared by Meij et al. (2012)[4]. Experimental results show that our method outperforms two baselines, i.e., Wikify! (Mihalcea and Csomai, 2007) and system proposed by Meij et al. (2012). We also study the effectiveness of features related to each kind of similarity, and demonstrate the advantage of our method for OOV mention linkage.

We summarize our contributions as follows.

1. We introduce a novel collective inference method that integrates three kinds of similarities, i.e., mention-entity similarity, entity-entity similarity, and mention-mention similarity, to simultaneously map a set of tweet mentions to their proper entities.

2. We propose modeling the mention-mention similarity and demonstrate its effectiveness in entity linking for tweets, particularly for OOV mentions.

3. We evaluate our method on a public data set, and show our method compares favorably with the baselines.

Our paper is organized as follows. In the next section, we introduce related work. In Section 3, we give the formal definition of the task. In Section 4, we present our solution, including the framework, features related to different kinds of similarities, and the training and decoding procedures. We evaluate our method in Section 5. Finally in Section 6, we conclude with suggestions of future work.

2 Related Work

Existing entity linking work can roughly be divided into two categories. Methods of the first category resolve one mention at each time, and mainly consider the similarity between a mention-entity pair. In contrast, methods of the second category take a set of related mentions (e.g., mentions in the same document) as input, and figure out their corresponding entities simultaneously.

Examples of the first category include the first Web-scale entity linking system SemTag (Dill et al., 2003), Wikify! (Mihalcea and Csomai, 2007), and the recent work of Milne and Witten (2008). SemTag uses the TAP knowledge base[5], and employs the cosine similarity with TF-IDF weighting scheme to compute the match degree between a mention and an entity, achieving an accuracy of around 82%. Wikify! identifies the important concepts in the text and automatically links these concepts to the corresponding Wikipedia pages. It introduces two approaches to define mention-entity similarity, i.e., the contextual overlap between the paragraph where the mention occurs and the corresponding Wikipedia pages, and a Naive Bayes classifier that predicts whether a mention should be linked to an entity. It achieves 80.69% F1 when two approaches are combined. Milne and Witten work on the same task of Wikify!, and also train a classifier. However, they cleverly use the

[4]http://ilps.science.uva.nl/resources/wsdm2012-adding-semantics-to-microblog-posts/

[5]TAB (http://www.w3.org/2002/05/tap/) is a shallow knowledge base that contains a broad range of lexical and taxonomic information about popular objects like music, movies, authors, sports, autos, health, etc.

links found within Wikipedia articles for training, exploiting the fact that for every link, a Wikipedian has manually selected the correct destination to represent the intended sense of the anchor. Their method achieves an F1 score of 75.0%.

Representative studies of the second category include the work of Kulkarni et al. (2009), Han et al. (2011), and Shen et al. (2012). One common feature of these studies is that they leverage the global coherence between entities. Kulkarni et al. (2009) propose a graphical model that explicitly models the combination of evidence from local mention-entity compatibility and global document-level topical coherence of the entities, and show that considering global coherence between entities significantly improves the performance. Han et al. (2011) introduce a graph-based representation, called Referent Graph, to model the global interdependence between different entity linking decisions, and jointly infer the referent entities of all name mentions in a document by exploiting the interdependence captured in Referent Graph. Shen et al. (2012) propose LIEGE, a framework to link the entities in web lists with the knowledge base, with the assumption that entities mentioned in a Web list tend to be a collection of entities of the same conceptual type.

Most work of entity linking focuses on web pages. Recently, Meij et al. (2012) study this task for tweets. They propose a machine learning based approach using n-gram features, concept features, and tweet features, to identify concepts semantically related to a tweet, and for every entity mention to generate links to its corresponding Wikipedia article. Their method belongs to the first category, in the sense that they only consider the similarity between mention (tweet) and entity (Wikipedia article).

Our method belongs to the second category. However, in contrast with existing collective approaches, our method works on tweets which are short and often noisy. Furthermore, our method is based on the "similar mention with similar entity" assumption, and explicitly models and integrates the mention similarity into the optimization framework. Compared with Meij et al. (2012), our method is collective, and integrates more features.

3 Task Definition

Given a sequence of mentions, denoted by $\vec{M} = (m_1, m_2, \cdots, m_n)$, our task is to output a sequence of entities, denoted by $\vec{E} = (e_1, e_2, \cdots, e_n)$, where e_i is the entity corresponding to m_i. Here, an entity refers to an item of a knowledge base. Following most existing work, we use Wikipedia as the knowledge base, and an entity is a definition page in Wikipedia; a mention denotes a sequence of tokens in a tweet that can be potentially linked to an entity.

Several notes should be made. First, we assume that mentions are given, e.g., identified by some named entity recognition system. Second, mentions may come from multiple tweets. Third, mentions with the same token sequence may refer to different entities, depending on mention context. Finally, we assume each entity e has a variation list[6], and a unique ID through which all related information about that entity can be accessed.

Here is an example to illustrate the task. Given mentions "nbcbightlynews", "Santiago", "WH" and "Libya" from the following tweet "Chuck Todd: Prepping for @nbcnightlynews here in Santiago, reporting on WH handling of Libya situation.", the expected output is "NBC Nightly News(194735)", "Santiago Chile(51572)", "White House(33057)" and "Libya(17633)", where the numbers in the parentheses are the IDs of the corresponding entities.

4 Our Method

In this section, we first present the framework of our entity linking method. Then we introduce features related to different kinds of similarities, followed by a detailed discussion of the training and decoding procedures.

4.1 Framework

Given the input mention sequence $\vec{M} = (m_1, m_2, \cdots, m_n)$, our method outputs the entity sequence $\vec{E}^* = (e_1^*, e_2^*, \cdots, e_n^*)$ according to Formula 1:

[6]For example, the variation list of the entity "Obama" may contain "Barack Obama", "Barack Hussein Obama II", etc.

$$\vec{E}^* = argmax_{\forall \vec{E} \in C(\vec{M})} \lambda \sum_{i=1}^{n} \vec{w} \cdot \vec{f}(e_i, m_i) + (1-\lambda) \sum_{i \neq j} r(e_i, e_j) s(m_i, m_j) \quad (1)$$

Where:

- $C(\vec{M})$ is the set of all possible entity sequences for the mention sequence $\vec{M}$;
- $\vec{E}$ denotes an entity sequence instance, consisting of $e_1, e_2, \cdots, e_n$;
- $\vec{f}(e_i, m_i)$ is the feature vector that models the similarity between mention m_i and its linked entity e_i;
- $\vec{w}$ is the feature weight vector related to $\vec{f}$, which is trained on the training data set; $\vec{w} \cdot \vec{f}(e_i, m_i)$ is the similarity between mention m_i and entity e_i;
- $r(e_i, e_j)$ is the function that returns the similarity between two entities e_i and e_j;
- $s(m_i, m_j)$ is the function that returns the similarity between two mentions m_i and m_j;
- $\lambda \in (0, 1)$ is a systematic parameter, which is determined on the development data set; it is used to adjust the tradeoff between local compatibility and global consistence. It is experimentally set to 0.8 in our work.

From Formula 1, we can see that: 1) our method considers the mention-entity similarly, entity-entity similarity and mention-mention similarity. Mention-entity similarly is used to model local compatibility, while entity-entity similarity and mention-mention similarity combined are to model global consistence; and 2) our method prefers configurations where similar mentions have similar entities and with high local compatibility.

$C(\vec{M})$ is worth of more discussion here. It represents the search space, which can be generated using the entity variation list. To achieve this, we first build an inverted index of all entity variation lists, with each unique variation as an entry pointing to a list of entities. Then for any mention m, we look up the index, and get all possible entities, denoted by $C(m)$. In this way, given a mention sequence $\vec{M} = (m_1, m_2, \cdots, m_n)$, we can enumerate all possible entity sequence $\vec{E} = (e_1, e_2, \cdots, e_n)$, where $e_i \in C(m)$. This means $|C(\vec{M})| = \prod_{m \in M} |C(m)|$, which is often large. There is one special case: if m is an OOV mention, i.e., $|C(m)| = 0$, then $|C(\vec{M})| = 0$, and we get no solution. To address this problem, we can generate a list of candidates for an OOV mention using its similar mentions. Let $S(m)$ denote OOV mention m's similar mentions, we define $C(m) = \bigcup_{m' \in S(m)} C(m')$. If still $C(m) = 0$, we remove m from $\vec{M}$, and report we cannot map it to any entity.

Here is an example to illustrate our framework. Suppose we have the following tweets:

- UserA: Yeaaahhgg #habemusfut.. I love monday night futbol =) #EnglishPremierLeague ManU vs Liverpool$_1$
- UserA: Manchester United 3 - Liverpool$_2$ 2 #EnglishPremierLeague GLORY, GLORY, MAN.UNITED!
- $\cdots$

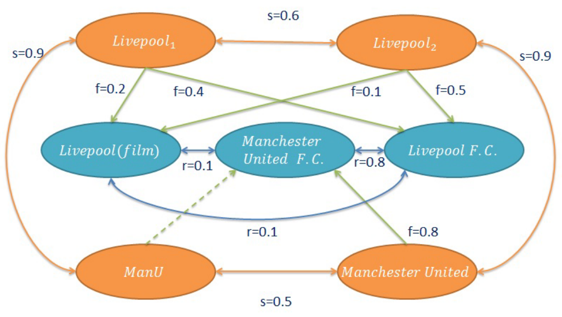

Figure 1: An illustrative example to show our framework. Ovals in orange and in blue represent mentions and entities, respectively. Each mention pair, entity pair, and mention entity pair have a similarity score represented by s, r and f, respectively.

We need find out the best entity sequence $\vec{E}^*$ for mentions $\vec{M}$ = { "Liverpool$_1$", "Manchester United", "ManU", "Liverpool$_2$"}, from the entity sequences $C(\vec{M})$ = { (Liverpool (film), Manchester United F.C., Manchester United F.C., Liverpool (film)), $\cdots$, (Liverpool, F.C.,Manchester United, F.C., Manchester United F.C., Liverpool (film) }. Figure 1 illustrate our solution, where "Liverpool$_1$" (on the left) and "Liverpool$_2$" (on the right) are linked to "Liverpool F.C." (the football club), and "Manchester United" and "ManU" are linked to "Manchester United F.C.". Notably, "ManU" is an OOV mention, but has a similar mention "Manchester United", with which "ManU" is successfully mapped.

4.2 Features

We group features into three categories: local features related to mention-entity similarity ($\vec{f}(e, m)$), features related to entity-entity similarity ($r(e_i, e_j)$) , and features related to mention-mention similarity ($s(m_i, m_j)$).

4.2.1 Local Features

- Prior Probability:

$$f_1(m_i, e_i) = \frac{count(e_i)}{\sum_{\forall e_k \in C(m_i)} count(e_k)} \quad (2)$$

  where $count(e)$ denotes the frequency of entity e in Wikipedia's anchor texts.

- Context Similarity:

$$f_2(m_i, e_i) = \frac{coocurence\_number}{tweet\_length} \quad (3)$$

  where: $coccurence\_number$ is the the number of the words that occur in both the tweet containing m_i and the Wikipedia page of e_i; $tweet\_length$ denotes the number of tokens of the tweet containing mention m_i.

- Edit Distance Similarity:
  If $Length(m_i)+ED(m_i, e_i) = Length(e_i)$, $f_3(m_i, e_i) = 1$, otherwise 0. $ED(\cdot, \cdot)$ computes the character level edit distance. This feature helps to detect whether a mention is an abbreviation of its corresponding entity[7].

- Mention Contains Title: If the mention contains the entity title, namely the title of the Wikipedia page introducing the entity e_i, $f_4(m_i, e_i) = 1$, else 0.

- Title Contains Mention: If the entry title contains the mention, $f_5(m_i, e_i) = 1$, otherwise 0.

[7]Take "ms" and "Microsoft" for example. The length of "ms" is 2, and the edit distance between them is 7. 2 plus 7 equals to 9, which is the length of "Microsoft".

4.2.2 Features Related to Entity Similarity

There are two representative definitions of entity similarity: in-link based similarity (Milne and Witten, 2008) and category based similarity (Shen et al., 2012). Considering that the Wikipedia categories are often noisy (Milne and Witten, 2008), we adopt in-link based similarity, as defined in Formula 4:

$$r(e_i, e_j) = \frac{log|g(e_i) \cap g(e_j)| - log\,max(|g(e_i)|, |g(e_j)|)}{log(Total) - log\,min(|g(e_i)|, |g(e_j)|)} \quad (4)$$

Where:

- $Total$ is the total number of knowledge base entities;

- $g(e)$ is the number of Wikipedia definition pages that have a link to entity e.

4.2.3 Features Related to Mention Similarity

We define 5 features to model the similarity between two mentions m_i and m_j, as listed below, where $t(m)$ denotes the tweet that contains mention m:

- $s_1(m_i, m_j)$: The cosine similarity of $t(m_i)$ and $t(m_j)$; and tweets are represented as TF-IDF vectors;

- $s_2(m_i, m_j)$: The cosine similarity of $t(m_i)$ and $t(m_j)$; and tweets are represented as topic distribution vectors;

- $s_3(m_i, m_j)$: Whether $t(m_i)$ and $t(m_j)$ are published by the same account;

- $s_4(m_i, m_j)$: Whether $t(m_i)$ and $t(m_j)$ contain any common hash tag;

- $s_5(m_i, m_j)$: Edit distance related similarity between m_i and m_j, as defined in Formula 5.

$$\begin{aligned} s_5(m_i, m_j) = 1,\ &if\ min\{Length(m_i), Length(m_j)\} \\ &+ED(m_i, m_j) = max\{Length(m_i), Length(m_j)\}, \\ else\ s_5(m_i, m_j) = 1 - &\frac{ED(m_i, m_j)}{max\{Length(m_i), Length(m_j)\}} \end{aligned} \quad (5)$$

Note that: 1) before computing TF-IDF vectors, stop words are removed; 2) we use the Stanford Topic Modeling Toolbox[8] to compute the topic model, and experimentally set the number of topics to 50.

[8]http://nlp.stanford.edu/software/tmt/tmt-0.4/

Finally, Formula 6 is used to integrate all the features. $\vec{a} = (a_1, a_2, a_3, a_4, a_5)$ is the feature weight vector for mention similarity, where $a_k \in (0,1)$, $k = 1,2,3,4,5$, and $\sum_{k=1}^{5} a_k = 1$.

$$s(m_i, m_j) = \sum_{k=1}^{5} a_k s_k(m_i, m_j) \tag{6}$$

4.3 Training and Decoding

Given n mentions $m_1, m_2, \cdots, m_n$ and their corresponding entities $e_1, e_2, \cdots, e_n$, the goal of training is to determine: $\vec{w}^*$, the weights of local features, and $\vec{a}^*$, the weights of the features related to mention similarity, according to Formula 7 [9].

$$(\vec{w}^*, \vec{a}^*) = arg\ min_{\vec{w},\vec{a}}\{\frac{1}{n}\sum_{i=1}^{n} L_1(e_i, m_i)$$
$$+\alpha_1||\vec{w}||^2 + \frac{\alpha_2}{2}\sum_{i,j=1}^{n} s(m_i, m_j) L_2(\vec{a}, e_i, e_j)\} \tag{7}$$

Where:

- L_1 is the loss function related to local compatibility, which is defined as $\frac{1}{\vec{w}\cdot\vec{f}(e_i,m_i)+1}$;
- $L_2(\vec{a}, e_i, e_j)$ is the loss function related to global coherence, which is defined as $\frac{1}{r(e_i,e_j)\sum_{k=1}^{5} a_k s_k(m_i,m_j)+1}$;
- α_1 is the weight of regularization, which is experimentally set to 1.0;
- α_2 is the weight of L_2 loss, which is experimentally set to 0.2.

Since the decoding problem defined by Formula 1 is NP hard (Kulkarni et al., 2009), we develop a greedy hill-climbing approach to tackle this challenge, as demonstrated in Algorithm 1.

In Algorithm 1, it is the number of iterations; $Score(\vec{E}, \vec{M}) = \lambda \sum_{i=1}^{n} \vec{w} \cdot \vec{f}(e_i, m_i) + (1 - \lambda)\sum_{i \neq j} r(e_i, e_j) s(m_i, m_j)$; $\vec{E}_{ij}$ is the vector after replacing e_i with $e_j \in C(m_i)$ for current $\vec{E}$; sc_{ij} is the score of $\vec{E}_{ij}$, i.e., $Score(\vec{E}_{ij}, \vec{M})$. In each iteration, this rounding solution iteratively substitute entry e_i in $\vec{E}$ to increase the total score cur. If the score cannot be further improved, it stops and returns current $\vec{E}$.

[9]This optimization problem is non-convex. We use coordinate descent to get a local optimal solution.

Algorithm 1 Decoding Algorithm.

```
Input: Mention Set $\vec{M} = (m_1, m_2, \cdots, m_n)$
Output: Entity Set $\vec{E} = (e_1, e_2, \cdots, e_n)$
 1: for $i = 1$ to $n$ do
 2:   Initialize $e_i^{(0)}$ as the entity with the largest prior
      probability given mention $m_i$.
 3: end for
 4: $cur = Score(\vec{E}^{(0)}, \vec{M})$
 5: $it = 1$
 6: while true do
 7:   for $i = 1$ to $n$ do
 8:     for $e_j \in C(m_i)$ do
 9:       if $e_j \neq e_i^{(it-1)}$ then
10:         $\vec{E}_{ij}^{(it)} = \vec{E}^{(it-1)} - \{e_i^{(it-1)}\} + \{e_j\}$.
11:       end if
12:       $sc_{ij} = Score(\vec{E}_{ij}^{(it)}, \vec{M})$.
13:     end for
14:   end for
15:   $(l, m) = argmax_{(i,j)} sc_{ij}$.
16:   $sc^* = sc_{lm}$
17:   if $sc^* > cur$ then
18:     $cur = sc^*$.
19:     $\vec{E}^{(it)} = \vec{E}^{(it-1)} - \{e_l^{(it-1)}\} + \{e_m\}$.
20:     $it = it + 1$.
21:   else
22:     break
23:   end if
24: end while
25: return $\vec{E}^{(it)}$.
```

5 Experiments

In this section, we introduce the data set and experimental settings, and present results.

5.1 Data Preparation

Following most existing studies, we choose Wikipedia as our knowledge base[10]. We index the Wikipedia definition pages, and prepare all required prior knowledge, such as $count(e)$, $g(e)$, and entity variation lists. We also build an inverted index with about 60 million entries for the entity variation lists.

For tweets, we use the data set shared by Meij et al. (2012)[11]. This data set is annotated manually by two volunteers. We get 502 annotated tweets from this data set. We keep 55 of them for

[10]We download the December 2012 version of Wikipedia, which contains about four million articles.

[11]http://ilps.science.uva.nl/resources/wsdm2012-adding-semantics-to-microblog-posts/.

development, and the remaining for 5 fold cross-validation.

5.2 Settings

We consider following settings to evaluate our method.

- Comparing our method with two baselines, i.e., Wikify! (Mihalcea and Csomai, 2007) and the system proposed by Meij et al. (2012) [12];
- Using only local features;
- Using various mention similarity features;
- Experiments on OOV mentions.

5.3 Results

Table 1 reports the comparison results. Our method outperforms both systems in terms of all metrics. Since the main difference between our method and the baselines is that our method considers not only local features, but also global features related to entity similarity and mention similarity, these results indicate the effectiveness of collective inference and global features. For example, we find two baselines incorrectly link "Nickelodeon" in the tweet "BOH will make a special appearance on Nickelodeon's 'Yo Gabba Gabba' tomorrow" to the theater instead of a TV channel. In contrast, our method notices that "Yo Gabba Gabba" in the same tweet can be linked to "Yo Gabba Gabba (TV show)", and thus it correctly maps "Nickelodeon" to "Nickelodeon (TV channel)".

System	Pre.	Rec.	F1
Wikify!	0.375	0.421	0.396
Meij's Method	0.734	0.632	0.679
Our Method	0.752	0.675	0.711

Table 1: Comparison with Baselines.

Table 2 shows the results when local features are incrementally added. It can be seen that: 1) using only Prior Probability feature already yields a reasonable F1; and 2) Context Similarity and Edit Distance Similarity feature have little contribution to the F1, while Mention and Entity Title Similarity feature greatly boosts the F1.

[12]We re-implement Wikify! since we use a new evaluation data set.

Local Feature	Pre.	Rec.	F1
P.P.	0.700	0.599	0.646
+C.S.	0.694	0.597	0.642
+E.D.S.	0.696	0.598	0.643
+M.E.T.S.	0.735	0.632	0.680

Table 2: Local Feature Analysis. P.P.,C.S., E.D.S., and M.E.T.S. denote Prior Probability, Context Similarity, Edit Distance Similarity, and Mention and Entity Title Similarity, respectively.

The performance of our method with various mention similarity features is reported in Table 3. First, we can see that with this kind of features, the F1 can be significantly improved from 0.680 to 0.704. Second, we notice that TF-IDF (s_1) and Topic Model (s_2) features perform equally well, and combining all mention similarity features yields the best performance.

Global Feature	Pre.	Rec.	F1
s_3+s_4+s_5	0.744	0.653	0.700
s_3+s_4+s_5 +s_1	0.759	0.652	0.702
s_3+s_4+s_5+s_2	0.760	0.653	0.703
s_3+s_4+s_5+s_1+s_2	0.764	0.653	0.704

Table 3: Mention Similarity Feature Analysis.

For any OOV mention, we use the strategy of guessing its possible entity candidates using similar mentions, as discussed in Section 4.1. Table 4 shows the performance of our system for OOV mentions. It can be seen that with our OOV strategy, the recall is improved from 0.653 to 0.675 (with $p < 0.05$) while the Precision is slightly dropped and the overall F1 still gets better. A further study reveals that among all the 125 OOV mentions, there are 48 for which our method cannot find any entity; and nearly half of these 48 OOV mentions do have corresponding entities [13]. This suggests that we may need enlarge the size of variation lists or develop some mention normalization techniques.

OOV Method	Precision	Recall	F1
Ignore OOV Mention	0.764	0.653	0.704
+ OOV Method	0.752	0.675	0.711

Table 4: Performance for OOV Mentions.

[13]"NATO-ukraine cooperations" is such an example. It is mapped to NULL but actually has a corresponding entity "Ukraine-NATO relations"

6 Conclusions and Future work

We have presented a collective inference method that jointly links a set of tweet mentions to their corresponding entities. One distinguished characteristic of our method is that it integrates mention-entity similarity, entity-entity similarity, and mention-mention similarity, to address the information lack in a tweet and rich OOV mentions. We evaluate our method on a public data set. Experimental results show our method outperforms two baselines, and suggests the effectiveness of modeling mention-mention similarity, particularly for OOV mention linking.

In the future, we plan to explore two directions. First, we are going to enlarge the size of entity variation lists. Second, we want to integrate the entity mention normalization techniques as introduced by Liu et al. (2012).

Acknowledgments

We thank the anonymous reviewers for their valuable comments. We also thank all the QuickView team members for the helpful discussions.

References

S. Dill, N. Eiron, D. Gibson, D. Gruhl, and R. Guha. 2003. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 178–186, New York, NY, USA. ACM.

Maxim Grinev, Maria Grineva, Alexander Boldakov, Leonid Novak, Andrey Syssoev, and Dmitry Lizorkin. 2009. Sifting micro-blogging stream for events of user interest. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 837–837, New York, NY, USA. ACM.

Xianpei Han and Jun Zhao. 2009. Nlpr-kbp in tac 2009 kbp track: A two-stage method to entity linking. In *Proceedings of Test Analysis Conference*.

Xianpei Han and Jun Zhao. 2010. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: A graph-based method. In *SIGIR'11*.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–465.

Xiaohua Liu, Ming Zhou, Xiangyang Zhou, Zhongyang Fu, and Furu Wei. 2012. Joint inference of named entity recognition and normalization for tweets. In *ACL (1)*, pages 526–535.

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*, SIGMOD '10, pages 1155–1158, New York, NY, USA. ACM.

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*.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceeding of the 17th ACM conference on Information and knowledge management*.

Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2010. Topical clustering of tweets. In *SWSM'10*.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Liege: Link entities in web lists with knowledge base. In *KDD'12*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 793–803, Stroudsburg, PA, USA. Association for Computational Linguistics.