To Link or Not to Link? A Study on End-to-End Tweet Entity Linking

Stephen Guo
Stanford University
sdguo@cs.stanford.edu

Ming-Wei Chang **Emre Kıcıman**
Microsoft Research
{minchang, emrek}@microsoft.com

Abstract

Information extraction from microblog posts is an important task, as today microblogs capture an unprecedented amount of information and provide a view into the pulse of the world. As the core component of information extraction, we consider the task of Twitter entity linking in this paper.

In the current entity linking literature, mention detection and entity disambiguation are frequently cast as equally important but distinct problems. However, in our task, we find that mention detection is often the performance bottleneck. The reason is that messages on micro-blogs are short, noisy and informal texts with little context, and often contain phrases with ambiguous meanings.

To rigorously address the Twitter entity linking problem, we propose a structural SVM algorithm for entity linking that jointly optimizes mention detection and entity disambiguation as a single end-to-end task. By combining structural learning and a variety of first-order, second-order, and context-sensitive features, our system is able to outperform existing state-of-the art entity linking systems by 15% F_1.

1 Introduction

Microblogging services, such as Twitter and Facebook, are today capturing the largest volume ever recorded of fine-grained discussions spanning a huge breadth of topics, from the mundane to the historic. The micro-blogging service Twitter reports that it alone captures over 340M short messages, or *tweets*, per day.[1] From such micro-blogging services' data streams, researchers have reported mining insights about a variety of domains, from election results (Tumasjan et al., 2010) and democracy movements (Starbird and Palen, 2012) to health issues and disease spreading (Paul and Dredze, 2011; Sadilek et al., 2012), as well as tracking product feedback and sentiment (Asur and Huberman, 2010).

A critical step in mining information from a micro-blogging service, such as Twitter, is the identification of entities in tweets. In order to mine the relationship between drugs, symptoms and side-effects, or track the popularity of politicians or sentiment about social issues, we must first be able to identify the topics and specific entities being discussed. The challenge is that messages on microblogs are short, noisy, and informal texts with little context, and often contain phrases with ambiguous meanings. For example, "one day" may be either a set phrase or a reference to a movie. Given such difficulties, current mining and analysis of microblogs lists limits its application to certain domains with easy-to-recognize, unambiguous entities in order to avoid noise in the extraction results.

We begin this paper with a thorough investigation of mention detection and entity disambiguation for social media, focused on the Twitter micro-blogging service. *Mention detection* is the task of extraction surface form candidates that can link to an entity in the domain of interest. *Entity disambiguation* is the task of linking an extracted mention to a specific definition or instance of an entity in a knowledge base.

[1] http://blog.twitter.com/2012/03/twitter-turns-six.html

While mention detection and entity disambiguation are frequently cast as equally important but distinct and separate problems, we find that mention detection is where today's systems and our baseline techniques incur the most failures. Detecting the correct entity mention is a significant challenge given mis-capitalizations, incorrect grammar, and ambiguous phrases. In (Ritter et al., 2011), the authors report their system achieves 0.64 to 0.67 F_1 on named entity segmentation results with 34K tokens of labeled examples. On the other hand, once the correct entity mention is detected, a trivial disambiguation that maps to the most popular entity[2] will achieve 85% accuracy in our set.

Our primary contribution in this paper is a recasting and merging of the tasks of mention detection and entity disambiguation into a single *end-to-end entity linking* task. We achieve significant improvements by applying structural learning techniques to jointly optimize the detection and disambiguation of entities. Treating detection and disambiguation as a single task also enables us to apply a large set of new features, conventionally used only for disambiguation, to the initial detection of mentions. These features, derived from external knowledge bases, include entity popularity and inter-entity relations from external knowledge bases, and are not well utilized in current mention detection systems. For example, consider the following partial tweet:

(1) The town is so, so good. And don't worry Ben, we already forgave you for Gigli. Really.

Determining whether or not "The town" is a mention of a location or other specific entity based solely on lexical and syntactic features is challenging. Knowing "The Town" is the name of a recent movie helps, and we can we be more confident if we know that Ben Affleck is an actor in the movie, and Gigli is another of his movies.

To train and evaluate our system, we created three separate annotated data sets of approximately 500 tweets each. These data sets are hand annotated with entity links to Wikipedia. We evaluate our system by comparing its performance at detecting entities to the performance of two state-of-the-art entity linking systems, Cucerzan (Cucerzan, 2007) and TagMe (Ferragina and Scaiella, 2010), and find that our system outperforms them significantly by 15% in absolute F_1.

The rest of this paper describes related work, our structured learning approach to entity linking, and our experimental results.

2 Related Work

Building an entity linking system requires solving two interrelated sub-problems: mention detection and entity disambiguation. The significant portion of recent work in the literature (Ratinov et al., 2011; Davis et al., 2012; Sil et al., 2012; Demartini et al., 2012; Wang et al., 2012; Han and Sun, 2011; Han et al., 2011) focuses solely upon the entity linking problem. The entity linking systems of these studies assume that entity mentions are provided by a separate mention detection system. In contrast, our study jointly identifies and disambiguates entity mentions within tweets (short text fragments).

A subset of existing literature targets end-to-end linking (Cucerzan, 2007; Milne and Witten, 2008; Kulkarni et al., 2009; Ferragina and Scaiella, 2010; Han and Sun, 2011; Meij et al., 2012), but there are quite a few differences between our work and each of these systems. Some systems (Milne and Witten, 2008; Kulkarni et al., 2009; Han and Sun, 2011) heavily depend on Wikipedia text and might not work well in short and noisy tweets. Many systems (Mihalcea and Csomai, 2007; Cucerzan, 2007; Milne and Witten, 2008; Ferragina and Scaiella, 2010) treat mention detection and entity disambiguation as two different problems. (Meij et al., 2012) is the most related to our paper. While their system also considers mention detection and entity disambiguation together, they do not consider entity-to-entity relationships and do not incorporate contextual words from tweets.

An area of work closely related to the mention detection problem is the Named Entity Recognition (NER) problem, the identification of textual phrases which belong to core categories (Person, Location, Organization). It is well-known that NER systems trained on well-written documents perform very poorly on short, noisy text, such as tweets (Rit-

[2]What we mean here is "the most linked entity". See Section 3 for details.

ter et al., 2011). There have been a few recent studies proposing Twitter-specific NER systems (Li et al., 2012; Ritter et al., 2011).

3 Preliminaries

For performing entity linking on Twitter, we choose Wikipedia as our external knowledge base of entities.

Entity We define an *entity* as a nonambiguous, terminal page (e.g., The Town (the film)) in Wikipedia (i.e., a Wikipedia page that is not a category, disambiguation, list, or redirect page). We define an *anchor phrase* (surface form) as the textual phrase (e.g., the town) which can potentially link to some entities. We define an *entity mention* as an anchor phrase *and* the context ("the town" in the example tweet in Section 1), where its semantic meaning umambiguously represents a specific entity. Note that an entity may be represented by multiple surface forms.

Wikipedia Lexicon Construction Following the assumptions used in most prior entity linking research, we assume that surface forms of entities can be found as anchor phrases in Wikipedia. In order to construct a Wikipedia lexicon, we first collect all anchors phases in Wikipedia. For each anchor phrase (surface form) s, we construct a lexicon entry by gathering the set of entities $\{e_1, e_2, \ldots e_K\}$ that can be linked from s. We also collect the number of times anchor a links to the entity e_i, $d(s, e_i)$. We define $P(e_i|s) = d(s, e_i)/d(s)$, where $d(s)$ represents the number of times s appears in Wikipedia. We refer e' as *the most linked entity* for anchor s if $e' = \arg\max_e P(e_i|s)$.

Candidate Generation Given a tweet t, we extract all k-grams of size $\leq k$. For each k-gram, we find all entities where this k-gram is an anchor phrase. If a k-gram is an anchor phrase for at least one entity, then the k-gram is a *candidate* entity mention. In general, we identify many candidate phrase per tweet; let $U(t) = \{c_1, c_2, \ldots\}$ denote the set of candidates in tweet t. We refer to $s(c)$ as the surface form (e.g., the anchor phrase) of c. Compared to the anchor phrase, the candidate also carries the context and position information. Let $E(c_i) = \{e_1, e_2, \ldots, \textbf{NIL}\}$ denote the set of entities which candidate i may be linked to, plus the additional special token **NIL**. Note that the size of $E(c_i)$ is always at least 2.

Task Definition First, our system generates candidate entity mentions, textual phrases which can possibly be entity mentions. Our system then performs filtering and optimization to process the list of candidates. For each candidate, our system links the candidate to a special **NIL** token or links the candidate to its corresponding entity in Wikipedia. More formally, given a tweet t and its candidate set $U(t)$, the goal of the system is to predict $y_i \in E(c_i), \forall c_i \in U(t)$.

Comparison to the TAC KBP Competition It is important to state that our definition of the entity linking problem differs significantly from the entity linking problem as defined by the TAC KBP competition (Ji et al., 2010; Ji et al., 2011). In the TAC, there is no true mention detection problem; every candidate in the TAC is an entity mention that represents an entity. Another difference is that the TAC allows for an entity mention to map to an entity not in the external knowledge base (Wikipedia); our system does not provide special handling of this case.

Comparison to Named Entity Recognition There are also important differences between our task and the canonical NER task. For example, NER systems identify common names, such as "Robert," as entities. In our task, we only consider a prediction as a success if the system can determine which person in Wikipedia "Robert" is referring to. In other words, our definition of entities depends on the given knowledge base, rather than human judgment. Hence, it is difficult to make a fair system comparison of our system to NER systems.

4 Entity Linking as Structural Learning

In our framework, we use structural learning as a tool to capture the relationship between entities. We define y_i as the output for c_i, where $y_i \in E(c_i)$. Let $T = |U(t)|$ and $\mathbf{y} = \{y_1, y_2, \ldots, y_T\}$. The feature function for the whole assignment can be written as $\Phi(t, U(t), \mathbf{y})$. The score for the assignment $\mathbf{y}$ can be obtained as the linear product between the weight vector $\mathbf{w}$ and the feature vector. For an input example, the prediction can be found by solving the

inference problem:

$$\mathbf{y}' = \arg\max_{\mathbf{y}} \mathbf{w}^T \Phi(t, U(t), \mathbf{y}) \quad (1)$$

We use a Structural SVM (SSVM) (Taskar et al., 2004; Tsochantaridis et al., 2005; Chang et al., 2010) as our learning algorithm. To train the weight vector $\mathbf{w}$, we minimize the objective function of the SSVM

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{l} \xi_i^2 \quad (2)$$

where l is the number of labeled examples and

$$\begin{aligned} &\mathbf{w}^T \Phi(t_i, c(t_i), \mathbf{y}_i) \\ \geq &\Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \Phi(t_i, c(t_i), \mathbf{y}) - \xi_i, \forall i, \mathbf{y} \end{aligned}$$

We denote $\mathbf{y}_i$ as the gold assignment for $\mathbf{x}_i$ and define $\Delta(\mathbf{y}_i, \mathbf{y})$ as the Hamming distance between two assignments $\mathbf{y}_i$ and $\mathbf{y}$.

4.1 Features

Feature definitions are very important as they define the shapes of the structures. Our feature vector is defined as

$$\Phi(t, U(t), \mathbf{y}) = \sum_i \phi(t, c_i, y_i) + \sum_{i<j} \phi(t, c_i, y_i, c_j, y_j)$$

where c_i and c_j is the i-th and j-th candidates in $U(t)$, respectively.

First, we assign $\Phi(t, c_i, \textbf{NIL})$ to be a special bias feature. The corresponding weight value behaves as a threshold to cut-off mentions. Recall in our definition that $y_i = \textbf{NIL}$ represents that the candidate c_i is not a mention.

The first order features for $\Phi(t, c_i, e)$ are described as follows. In general, we can classify our features into two types: mention-specific features and entity-specific features. For a given candidate c_i, mention-specific features only consider the surface form of c_i and the tweet t. Entity-specific features also consider the knowledge base content of the entity e. Prior work in the entity linking literature has primarily focused on entity-specific features, as most prior work solves entity disambiguation with given mentions.

Base and Capitalization Rate Our base features are from two resources. Let $s(c)$ denote the surface form of candidate c. The link probability $P_l(s(c))$ and $P(e|s(c))$ features are extracted from Wikipedia. We explained $P(e|s(c))$ in Section 3. Link probability $P_l(s(c))$ is the probability that a phrase is used as an anchor in Wikipedia. We also add a third feature that captures normalized link count. Besides these three features, we also have a feature to indicate if a is a stop word, and a feature indicating the number of tokens in a. The view count and $P(e|s)$ features are entity-specific, while the other three features are mention-specific.

For each phrase $s(c)$, we also collect statistics about the probability that a phrase is capitalized in Wikipedia. We refer to this feature as the capitalization rate feature, $P_c(s(c))$.

Popularity Feature We have access to 300GBs of Wikipedia page view counts, representing one months worth of page view information, we use this as popularity data.[3] As mentioned in Section 3, we find that the most often linked Wikipedia articles might not be the most popular ones on Twitter. Using page view statistics helps our system correct this bias. We define another probability based on page view statistics $P_v(e_i|c) = v(e_i)/(\sum_{e \in E(c)/\{\textbf{NIL}\}} v(e))$, where $v(e)$ represents the view count for the page e.

Context Capitalization Our context capitalization features indicate if the current candidate, the word before, and the word after the candidate are capitalized.

Entity Type and Tf-idf We use the procedure proposed in (Ratinov et al., 2011) to extract keyword phrases from categories for each Wikipedia page, and then build a rule-based system using keyword phrases to classify if each entity page belongs to one of the following entity types: Person, Location, Organization, TV Show, Book/Magazine and Movie.[4] For a given candidate c and an entity e, the associated binary feature becomes active if the entity belongs to a specific entity type. There are six entity type features in our system.

[3] `http://dammit.lt/wikistats`

[4] The entity type prediction accuracy of our rule-based system on the development set is around 95%.

Features	Descriptions
Base	$P_l(s_i)$, $P(e\|s)$, normalized link counts, stop word, # tokens
Cap. Rate	$P_c(s_i)$
Popularity	$P_v(e\|s)$, normalized page view count, $P_v(e\|s)P(e\|s)$
Context Cap.	Three features indicating if the current candidate and the words before and after are capitalized
Entity Type	Six binary features for each entity type
Tf-idf	Two features for the similarity between the word vectors of the entity and the tweet
Second-Order	$Jac(e_i, e_j)$, $P(e_i\|s_i)P(e_j\|s_j)$, $P_c(s_i)P_c(s_j)$, $P_l(s_i)P_l(s_j)$

Table 1: Summary of the features used in our structural learning systems.

We also include tf-idf features in our system. For each Wikipedia page, we collect the top 100 tf-idf words. We add one feature that is the dot product between the tf-idf word vector of e and the words of tweet t. We include a second feature that represents the average tf-idf score of all words that appear in both e and t.

Second-order features We include four very simple second-order features $\phi(t, c_i, e_i, c_j, e_j)$ to capture more complex relations between entities and candidates. The first feature is the Jaccard distance between two Wikipedia pages e_i and e_j. Let $\Gamma(e_i)$ denote the set of Wikipedia pages that contain a hyperlink to e_i. We define the Jaccard distance between e_i and e_j as:

$$Jac(e_i, e_j) = \frac{|\Gamma(e_i) \cap \Gamma(e_j)|}{|\Gamma(e_i) \cup \Gamma(e_j)|}$$

This feature has a similar effect as the normalized Google distance (Cilibrasi and Vitanyi, 2007), which has been used for many entity linking systems. Let us use the following shorthand: $s_i = s(c_i)$ and $s_j = s(c_j)$. We have also included three features $P(e_i|s_i)P(e_j|s_j)$, $P_c(s_i)P_c(s_j)$ and $P_l(s_i)P_l(s_j)$ to increase the expressivity of our model.

4.2 Mining Additional Contextual Words

Unlike mention detection systems used in other NLP tasks, there are no lexical features in our system. Lexical features are important as they can capture semantic meaning precisely. However, given that we do not have many labeled examples, lexical features can lead to overfitting. The diverse language in tweets also make it more difficult to use lexical features.

Our solution for this problem is to use a very simple method to mine context words for different entities from a large, unlabeled tweet corpus. The algorithm works as follows:

1. Train an end-to-end entity linking system and then apply it to a large, unlabeled tweet corpus
2. Extract contextual words for each entity type based on the pseudo-labeled data.
3. Train the entity linking system again with new contextual features.

In this paper, we only use the word before and the word after as our contextual word for a candidate. Note that while there are ambiguous phrases on the surface (e.g., "friends" can be a TV show or just a regular phrase), certain phrases are unambiguous (e.g., "CSI : Miami"). As contextual words are often shared within the same entity type (e.g. "watching" is likely to appear before a tv show), those words can potentially improve our final system.

Let w_i denote the i-th word in the tweet and t_i denote the entity type for the i-th word.[5] We use a very simple rule to select a set of left context words $Q(R)$ for entity type R.

$$Q(R) = \{w_i \mid P(t_{i+1} = R|w_i) > r, d(w_i) > z\}$$

where $d(w_i)$ represent the number of times the word w_i appears in the unlabeled set. The first rule is to simply find a word which is more likely to be followed by an entity. The second rule filter outs noisy words (e.g., Twitter handles) in the unlabeled set. The right context words are also extracted in a similar way.

To train the second end-to-end entity linking system, we add one additional feature for the contextual words. For the feature vector $\Phi(t, c_i, e)$, the context feature is active if the candidate c_i is capitalized[6] and the context words around c_i belongs to $Q(R)$, given R is the entity type for the entity e.

[5]The tag t_i belongs to the entity type R if our system links a candidate c to an entity with type R and c covers the word w_i.

[6]The word "watching" can be a TV show while most of the time it is not. These common makes this contextual feature noisy. We found that the context feature can only be reliably applied when the candidate is capitalized.

4.3 Cohesiveness Score

There are several ways to consider entity-entity cohesiveness besides using the second-order features directly. In our model, we also consider a modified cohesiveness score proposed in (Ferragina and Scaiella, 2010). The idea behind the cohesiveness score is to estimate the correlations between different entities by using weighted Jaccard scores.[7]

There are two rounds in the procedure of computing the cohesiveness score. We first estimate approximately the most probable entity for each candidate given all the other candidates in the same tweet. In the second round, the cohesiveness score is then produced with respect to the most probable entity computed in the first round.

More formally, in the first round, we compute the relevance score for each candidate and entity pair:

$$Rel(e, c|t) = \frac{\sum_{c' \neq c} \sum_{e' \in E(c')} P(e'|c') Jac(e, e')}{|U(t)|}.$$

Then, the cohesiveness score is computed by

$$S_{coh}(e, c|t) = \frac{\sum_{c' \neq c} Jac(e, \bar{e}(c')) P(\bar{e}(c')|c')}{|U(t)|},$$

where the $\bar{e}(c') = \arg\max_{e \in E(c')} Rel(e, c'|t)$. We then put the cohesiveness score as a feature for each (e, c) pair. In practice, we found that the cohesiveness score in the model can significantly increase the disambiguation ability of the model without using the second-order information.

4.4 Inference

In order to train and test the SSVM model, one needs to solve both the inference problem Eq. (3) and the loss-augmented inference problem. Without second-order features, the inference and loss-augmented inference problems can be easily solved, given that each component can be solved independently by

$$y'_i = \arg\max_{y \in E(c_i)} \mathbf{w}^T \Phi(t, c_i, y) \quad (3)$$

While the inference problem can be solved independently, the training algorithm still considers the whole assignment together in the training procedure.

[7] In our experiments, we only apply the cohesiveness score technique on candidates which pass the filtering procedure. See section 5 for more details for our filtering process.

Data	#Tweets	#Cand	#Men.	P@1
Train	473	8212	218	85.3%
Test 1	500	8950	249	87.7%
Test 2	488	7781	332	89.6%

Table 2: Labeled example statistics. "#Cand" represents the total number of candidates we found in this dataset. "#Men." is the total number of mentions that disambiguate to an entity. The top-1 rate (P@1) represents the proportion of the mentions that disambiguate to the most linked entity in Wikipedia.

With the second-order features, the inference problem becomes NP-hard. While one can resort to using integer linear programming to find the optimal solution, we choose not to do so. We instead use the beam search algorithm. Our beam search algorithm first arranges the candidates from left to right, and then solve the inference problems approximately.

5 Experiments

We collected unlabeled Twitter data from two resources and then asked human annotators to label each tweet with a set of entities present. Our annotators ignored the following: duplicate entities per tweet, ambiguous entity mentions, and entities not present in Wikipedia. We next describe the two sets of Twitter data used as our training data and testing data. In addition to these two datasets, we also randomly sampled another 200 tweets as our development set.

Ritter We sampled 473 and 500 tweets[8] from the data used in (Ritter et al., 2011) to be our training data and test data, respectively. We did not use any labels generated by (Ritter et al., 2011); our annotators completely re-annotated each tweets with its set of entities. We refer to the first set as **Train** and the second set as **Test 1**.

Entertainment To check if our system has the ability to generalize across different domains, we sampled another 488 tweets related to entertainment entities. Our main focus was to extract tweets that contained TV shows, Movies, and

[8] We originally labeled 1000 tweets but then found 27 repeated tweets in the dataset. Therefore, we remove those 27 tweets in the training set.

Books/Magazines. Identifying tweets from a specific domain is a research topic on its own, so we followed (Dalvi et al., 2012), and used a keyword matching method.[9] After sampling this set of tweets, we asked our annotators to label the data in the same way as before (all entities are labeled, not just entertainment entities). We refer to this tweet set as **Test 2**.

After sampling, all tweets were then normalized in the following way. First, we removed all retweet symbols (RT) and special symbols, as these are tokens that may easily confuse NER systems. We treated punctuation as separate tokens. Hashtags (#) play a very important role in tweets as they often carry critical information. We used the following web service[10] to break the hashtags into tokens (e.g., the service will break "#TheCloneWars" into "the clone wars") (Wang et al., 2011).

The statistics of our labeled examples are presented in Table 2. First, note that the average number of mentions per tweet is well below 1. In fact, many tweets are personal conversations and do not carry any entities that can be linked to Wikipedia. Still, many candidates are generated (such as "really") for those tweets, given that those candidates can still potentially link to an entity ("really" could be a TV channel). Therefore, it is very important to include tweets without entities in the training set because we do not want our system to create unnecessary links to entities.

Another interesting thing to note is the percentage of entity mentions that disambiguate directly to their most often linked entities in Wikipedia. If we simply disambiguate each entity mention to its most linked entity in Wikipedia, we can already achieve 85% to 90% accuracy, if mention detection is perfectly accurate. However, mention detection is a difficult problem as only about 3% of candidates are valid entity mentions.

It is worthwhile to mention that, as per (Ferragina and Scaiella, 2010), for computational efficiency, we apply several preprocessing steps before running our entity linking system. First, for each anchor in Wikipedia, we gather all entities it can disambiguate to and remove from that anchor's entity set all entities that are linked less than 2% of the time. Second, we apply a modified filtering procedure similar to that proposed in (Ferragina and Scaiella, 2010) to filter the set of candidates per tweet.

Evaluation Our annotated datasets contain entities from many Wikipedia categories. For evaluation, we primarily focus on entities belonging to a set of six *core* categories (Person, Location, Organization, TV Show, Book/Magazine, Movie). We believe it is necessary to focus upon core entities, rather than considering all possible entities in Wikipedia. Most common words in the English language have their own Wikpedia page, but most words are not important enough to be considered entities. In general, there is a large degree of subjectivity when comparing different entity linking datasets; different researchers have their own interpretation of what constitutes an entity. For example, we examined the annotation used in (Meij et al., 2012) and found it to be extremely lenient, when compared to our own beliefs of what is an entity. Therefore, we believe evaluating performance on restricted entity types is the only fair way to compare different end-to-end entity linking systems.

We evaluate the performance of our system on a per-tweet basis, by comparing the set of annotated "gold" entities with the set of entities predicted by our system, and computing performance metrics (precision, recall, F_1). We choose to evaluate our system on a per-tweet basis, as opposed to a per-entity basis, because we wish to avoid the issue of matching segmentations. For example, it is quite common to observe multiple overlapping phrases in a tweet that should be linked to the same entity (e.g., "President Obama" and "Obama"). When evaluating our system, we compute performance metrics for both all entities and core entities.[11]

Parameters In our implementation, we fixed the regularization parameter $C = 10$. When beam-

[9] We use the following word list :"movie", "tv", "episode", "film", "actor", "actors", "actress", "director", "directors", "movies", "episodes", "book", "novel", "reading", "read", "watch", "watching", "show", "books", "novels", "movies", "author" and "authors".

[10] `http://web-ngram.research.microsoft.com/info/break.html`

[11] To decide if an entity is a core entity or not, we use the following procedure. For the gold entities, the annotators also annotate type of the entity. We decide the entity type of the predicted entities using the procedure described in Section 4.1.

Model	Test 1			Test 2		
	P	R	F_1	P	R	F_1
Cucerzan	64.8	42.2	51.1	64.9	39.7	49.5
TagMe	38.8	69.0	49.7	34.9	70.3	46.7
SSVM	78.8	59.9	**68.0**	75.0	57.7	**65.2**

Table 3: Comparisons between different end-to-end entity linking systems. We evaluate performance on core entities, as it is the only fair way to compare different systems.

Structural SVM	Test 1		Test 2	
	All	Core	All	Core
Base	35.9	42.9	47.7	52.5
+Cap. Rate	38.4	45.6	49.9	53.7
+Popularity	41.3	47.9	50.3	55.1
+Context Cap	43.7	52.0	50.7	54.8
+Entity Type	47.9	57.0	53.5	59.0
+Tfidf	53.2	63.1	56.8	61.9

Table 4: Feature Study: F_1 for entity linking performance. "All" means evaluation on all annotated entities. "Core" means evaluation only on our six entity types. Each row contains all additional features of the row above it.

search is used, the beam size is set to be 50, and we only consider the top 10 candidates for each candidate to speed the inference process. In the context word mining algorithm, $r = 0.5\%$ and $z = 1000$.

5.1 Results

In the following, we analyze the contributions of each component in our system and compare our final systems to other existing end-to-end entity linking systems.

System Comparison We compare our final system to other state-of-the-art systems in Table 3. CUCERZAN represents a modified implementation of the system in (Cucerzan, 2007). TagMe is an end-to-end linking system that focuses on short texts, including tweets. Our system significantly outperforms these two systems in both precision and recall. Note that CUCERZAN's system is a state-of-the-art system on well-written documents with provided entity mentions. The system (Cucerzan, 2007) has been extended by the authors and won the TAC KBP competition in 2010 (Ji et al., 2010).

There are two possible reasons to explain why our system outperforms CUCERZAN. First, their mention detection is a carefully designed system targeted toward documents, not tweets. Their system has segmentation issues when applied to Twitter, as it relies heavily upon capitalization when identifying candidate entity mentions. Second, their system heavily depends on the fact that related entities should appear together within documents. However, given that tweets are very short, some of their most important features are not suitable for the Twitter domain. Our system outperforms TagMe because we use a more sophisticated machine learning approach, as compared to their system. TagMe links too many spurious entity mentions for common words. This is a result of their algorithm's over-emphasis on entity-entity co-occurrence features.

Feature Study We study the contributions of each feature group in our system in Table 4. We summarize our discoveries as follows:

First, we find collecting statistics from a large corpus helps the system significantly. In addition to $P(e|s)$, we find that capitalization rate features offer around 3% to 4% F_1 improvement in Test 1. Similarly, popularity features are also important, as it corrects bias existing in Wikipedia link statistics. Compared to lexical features, using statistical features offers a great advantage of reducing the need for large amounts of labeled data.

We also find entity related features (Popularity, Entity Type, Tf-idf) are crucial. Given that between 85% to 90% of our mentions should directly disambiguate to the most often linked entities, one might think entity-specific features are not important in our task. Interestingly, entity-specific features are among the most important features. The discovery confirms our hypothesis: it is critical to consider mention detection and entity disambiguation as a single problem, rather than as separate problems in a two staged approach used by many other entity linking systems. Note that capitalization rate and context capitalization features are mention-specific. Additionally, we find that mixing mention-specific features and entity-specific features results in a better model.

Entity Type	Words appearing before the mention	Words appearing after the mention
Person	wr, dominating, rip, quarterback, singer, featuring, defender, rb, minister, actress, twitition, secretary	tarde, format, noite, sufferers, dire, admits, senators, urges, performs, joins
TV Show	sbs, assistir, assistindo, otm, watching, nw, watchn, viagra, watchin, ver	skit, performances, premieres, finale, parody, marathon, season, episodes, spoilers, sketch

Table 5: An example of context words that are automatically extracted from 20 million unlabeled tweets. For the sake of brevity, we only display context words for two categories. Note that there are misspelled words (such as "watchn") and abbreviations (such as nw) that do not appear in well-written documents.

Advance Models	Test 1		Test 2	
	All	Core	All	Core
SSVM (Table 4)	53.2	63.1	56.8	61.9
+Context	53.9	64.6	58.6	63.4
+Cohesiveness	55.6	66.5	59.7	65.1
+2nd order	58.1	68.0	60.6	65.2

Table 6: Evaluation results (F_1) of the advanced models. "+ Context" is the model that uses additional context features extracted from 20 millions unlabeled tweets. "+ Cohesiveness" is the model with both additional context and cohesiveness features. "+2nd order" is our final model (which incorporates context, cohesiveness, and second-order features).

Mining Context Words We verify the effectiveness of adding contextual features that are extracted automatically from large unlabeled data. We apply our system (with all first-order features) on a set of 20 million unlabeled tweets we collected. Context words are then extracted using the simple rules described in Section 4. We list the top 10 words we extracted in Table 5. Due to space limitations, we only list the words for the Person and TV Show categories. The results are interesting as we are able to find common misspelled words and abbreviations used in Twitter. For example, we find that "watchn" means "watching" and "nw" means "now watching," and they are usually words found before TV shows. We also find tweeters frequently use abbreviations for people's jobs. For example, "wr" means "wide receiver" and "rb" means "running back." When mined context is added into our system, the performance improves significantly (Table 6). We note that extending context mining algorithms in a large-scale, principled approach is an important next research topic.

Capturing Entity-Entity Relationships In this paper, we use two methods to capture the relationship between entities: adding the cohesiveness score and using second order information. Until now, we only considered features that can be extracted from only one entity. Past research has shown that considering features that involve multiple entities can improve entity linking performance, given that related entities are more likely to appear together in a document. When these type of features are added, we need to perform beamsearch, as the exact inference procedure can be prohibitively expensive.

As displayed in Table 6, we find that either adding the cohesiveness score or using second order information can improve prediction. Using both methods improves the model even more. Comparing computation overhead, computing cohesiveness is significantly more cost-effective than using second-order information.

6 Conclusion

In this paper, we propose a structural SVM method to address the problem of end-to-end entity linking on Twitter. By considering mention detection and entity disambiguation together, we build a end-to-end entity linking system that outperforms current state-of-the-art systems.

There are plenty of research problems left to be addressed. Developing a better algorithm for mining contextual words is an important research topic. It would also be interesting to design a method that jointly learns NER models and entity linking models.

References

S. Asur and B.A. Huberman. 2010. Predicting the future with social media. *arXiv preprint arXiv:1003.5699*.

M. Chang, V. Srikumar, D. Goldwasser, and D. Roth. 2010. Structured output learning with indirect supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*.

R.L. Cilibrasi and P.M.B. Vitanyi. 2007. The google similarity distance. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):370–383.

S. Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference of EMNLP-CoNLL*, pages 708–716.

N. Dalvi, R. Kumar, and B. Pang. 2012. Object matching in tweets with spatial models. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 43–52, New York, NY, USA. ACM.

A. Davis, A. Veloso, A. S. da Silva, W. Meira, Jr., and A. H. F. Laender. 2012. Named entity disambiguation in streaming data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 815–824, Stroudsburg, PA, USA. Association for Computational Linguistics.

G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. 2012. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *The International World Wide Web Conference*, pages 469–478, New York, NY, USA. ACM.

P. Ferragina and U. Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1625–1628, New York, NY, USA. ACM.

X. Han and L. Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 945–954, Stroudsburg, PA, USA. Association for Computational Linguistics.

X. Han, L. Sun, and J. Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 765–774, New York, NY, USA. ACM.

H. Ji, R. Grishman, H.T. Dang, K. Griffitt, and J. Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Proceedings of the TAC 2010 Workshop*.

H. Ji, R. Grishman, and Dang. 2011. Overview of the tac 2011 knowledge base population track. In *Proceedings of the TAC 2011 Workshop*.

S. Kulkarni, A. Singh, G. Ramakrishnan, and S. 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*, Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD), pages 457–466, New York, NY, USA. ACM.

C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. 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*, Proceedings of International Conference on Research and Development in Information Retrieval, SIGIR, pages 721–730, New York, NY, USA. ACM.

E. Meij, W. Weerkamp, and M. de Rijke. 2012. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 563–572, New York, NY, USA. ACM.

R. Mihalcea and A. Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 233–242. ACM.

D. Milne and I. H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 509–518, New York, NY, USA. ACM.

M.J. Paul and M. Dredze. 2011. You are what you tweet: Analyzing twitter for public health. In *Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 2011)*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1375–1384, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. Ritter, S. Clark, Mausam, and O. Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1524–1534, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. Sadilek, H. Kautz, and V. Silenzio. 2012. Modeling spread of disease from social interactions. In *Sixth AAAI International Conference on Weblogs and Social Media (ICWSM)*.

A. Sil, E. Cronin, P. Nie, Y. Yang, A.-M. Popescu, and A. Yates. 2012. Linking named entities to any database. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 116–127, Stroudsburg, PA, USA. Association for Computational Linguistics.

K. Starbird and L. Palen. 2012. (how) will the revolution be retweeted?: information diffusion and the 2011 egyptian uprising. In *Proceedings of the acm 2012 conference on computer supported cooperative work*, pages 7–16. ACM.

B. Taskar, C. Guestrin, and D. Koller. 2004. Max-margin markov networks. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, September.

A. Tumasjan, T.O. Sprenger, P.G. Sandner, and I.M. Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the fourth international aaai conference on weblogs and social media*, pages 178–185.

K. Wang, C. Thrasher, and B.J.P. Hsu. 2011. Web scale nlp: a case study on url word breaking. In *The International World Wide Web Conference*, pages 357–366. ACM.

C. Wang, K. Chakrabarti, T. Cheng, and S. Chaudhuri. 2012. Targeted disambiguation of ad-hoc, homogeneous sets of named entities. In *The International World Wide Web Conference*, pages 719–728, New York, NY, USA. ACM.