TwiNER: Named Entity Recognition in Targeted Twitter Stream

Chenliang Li* [1], Jianshu Weng[2], Qi He[3], Yuxia Yao[2], Anwitaman Datta[1],
Aixin Sun[1], and Bu-Sung Lee[1,2]
[1] School of Computer Engineering, Nanyang Technological University, Singapore
{lich0020,anwitaman,axsun,ebslee}@ntu.edu.sg
[2] Services Platform Lab, HP Labs, Singapore
{jianshu.weng,yuxia.yao,francis.lee}@hp.com
[3] Almaden Research Center, IBM, USA
heq@us.ibm.com

ABSTRACT

Many private and/or public organizations have been reported to create and monitor targeted *Twitter* streams to collect and understand users' opinions about the organizations. Targeted *Twitter* stream is usually constructed by filtering tweets with user-defined selection criteria (*e.g.,* tweets published by users from a selected region, or tweets that match one or more predefined keywords). Targeted *Twitter* stream is then monitored to collect and understand users' opinions about the organizations. There is an emerging need for early crisis detection and response with such target stream. Such applications require a good named entity recognition (NER) system for *Twitter*, which is able to automatically discover emerging named entities that is potentially linked to the crisis. In this paper, we present a novel 2-step unsupervised NER system for targeted *Twitter* stream, called *TwiNER*. In the first step, it leverages on the *global context* obtained from Wikipedia and Web N-Gram corpus to partition tweets into valid segments (phrases) using a dynamic programming algorithm. Each such tweet segment is a candidate named entity. It is observed that the named entities in the targeted stream usually exhibit a *gregarious* property, due to the way the targeted stream is constructed. In the second step, *TwiNER* constructs a random walk model to exploit the *gregarious* property in the *local context* derived from the *Twitter* stream. The highly-ranked segments have a higher chance of being true named entities. We evaluated *TwiNER* on two sets of real-life tweets simulating two targeted streams. Evaluated using labeled ground truth, *TwiNER* achieves comparable performance as with conventional approaches in both streams. Various settings of *TwiNER* have also been examined to verify our *global context + local context* combo idea.

*The work was partially done when Chenliang Li was an intern at HP Labs Singapore.

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

Categories and Subject Descriptors

H.3.4 [**Information Systems**]: Content Analysis and Indexing—*Linguistic processing*

Keywords

Twitter, Tweets, Named Entity Recognition, Wikipedia, Web N-Gram

1. INTRODUCTION

Twitter, as a new type of social media, has seen tremendous growth in recent years. It has attracted great interests from both industry and academia. Many private and/or public organizations have been reported to monitor *Twitter* stream to collect and understand users' opinions about the organizations. Nevertheless, due to the extremely large volume of tweets published every day[1], it is practically infeasible and unnecessary to listen and monitor the whole *Twitter* stream. Therefore, targeted *Twitter* streams are usually monitored instead; each such stream contains tweets that potentially satisfy some information needs of the monitoring organization. Targeted *Twitter* stream is usually constructed by filtering tweets with user-defined selection criteria depends on the information needs. For example, the criterion could be a region so that users' opinions from that particular region are collected and monitored; it could also be one or more predefined keywords so that opinions about some particular events/topics/products/services can be monitored.

There is also an emerging need for early crisis detection and response with such target stream. For example, a cosmetic company is interested in automatically discovering any new named entities (e.g. person names, competitor names, or location names) in a targeted stream it creates for the company and its products, which may link to a potential PR crisis. By doing this, the company is able to acquire first-hand information about the crisis and make early response. Such applications require a good named entity recognition (NER) system for *Twitter*, which is the focus of this paper.

Nevertheless, the nature of tweets brings new challenges. Traditional NER methods on well-formatted documents heavily depend on a phrase's local linguistic features [14], such as capitalization, part-of-speech (POS) tags of previous words, etc. However, tweets are usually informal in nature and short (up to 140 charac-

[1]There are more than 200 million tweets published per day, according to `http://blog.twitter.com/2011/06/200-million-tweets-per-day.html`.

ters). They often contain grammatical errors, misspellings, and unreliable capitalizations. These unreliable linguistic features cause traditional methods to perform poorly on tweets. We use the real examples below to illustrate the challenges when applying traditional NER on tweets.

Table 1: Example Named Entities in Tweet

1	**PAP** POSTERS ARE EVERYWHERE! AND FOR SOME LAMP POLES THERE ARE BOTH **NSP** AND **PAP** POSTERS! #whathappentosavingtheearth
2	ya la!! some of them gg to **potong pasir**. I'm gg to **yio chu kang**

Table 1 shows two real tweets collected during a political event. A POS tagger would fail due to the tweets' abnormal capitalization and grammatical errors. For example, in the first tweet, all words (except "PAP" and "NSP")[2] are mislabeled as *NNP* (singular noun). Similarly, in the second tweet, "potong" and "yio" are mislabeled as *JJ* (adjective) and *VB* (verb) respectively, although both are a part of location names ("potong pasir" and "yio chu kang"). This kind of noisy POS labels would make NER tagger fail to recognize named entities.

To address the above challenges caused by tweets' error-prone and short nature, this paper presents a novel unsupervised NER system for targeted tweet streams, called *TwiNER*. Based on the *gregarious* property of named entities in targeted tweet stream, *TwiNER* recognizes named entities collectively from a batch of tweets in unsupervised manner. More formally, let T be the collection of tweets in question. *TwiNER* receives tweets from T in a batch manner. A batch is the set of tweets posted in the targeted *Twitter* stream within one fixed time interval (e.g. a second). So, $T = \{T_1, T_2, .., T_n\}$ and T_i is the batch of tweets posted in the ith interval. *TwiNER* then recognizes all possible named entities in T_i regardless of their types.

It is noted that currently *TwiNER* does not categorize the type of named entity (*e.g.,* person, location). As conventional NER methods fail to address the new challenges posed by emerging social media like *Twitter*, it is more pressing to be able to discover the presence of named entities in targeted *Twitter* stream before we could categorize their types. Furthermore, even without categorizing the types of named entities, *TwiNER* already enable us to make early crisis response. For example, a cosmetic company may be interested in discovering any new named entity which may directly/indirectly link to the company and subsequently causes a PR crisis, be it a person name, product name, or company name. Moreover, as a targeted Twitter stream is constructed for a particular information need, we assume that the user who constructs the stream has the background knowledge in interpreting the named entities detected. In the following subsections, we give an overview of *TwiNER*.

Figure 1 shows the general system architecture of *TwiNER*, which has two main components, namely *tweet segmentation* and *segment ranking*.

1.1 Tweet Segmentation

As shown in Table 1, traditional linguistic features (e.g., capitalization) are unreliable in tweets. Is there any other feature in tweets we can utilize for the task of NER?

In the same examples in Table 1, people spell "yio chu kang" rather than "chu kang yio" or "kang chu yio". In other words, the *correct collocation of a named entity* is still preserved in tweets. This observation holds stronger if a larger set of tweets are aggregated together. This motivates us to learn a weak phrase segmenter for tweets first.

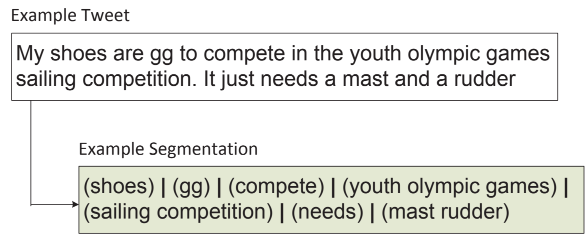

Figure 2: Example of Tweet Segmentation

The idea is to segment an individual tweet into a sequence of consecutive phrases[3], each of which appears "more than chance" [2, 18]. Figure 2 gives an example. In this example, after removing the stop words, a tweet "*My shoes are gg to compete in the youth olympic games sailing competition. It just needs a mast and a rudder*" is segmented into seven parts.

More formally, given a tweet of four words $w_1w_2w_3w_4$, we segment it as $w_1w_2||w_3w_4$ rather than $w_1||w_2w_3w_4$, if $C(w_1w_2)+C(w_3w_4) > C(w_1) + C(w_2w_3w_4)$, where $C(\cdot)$ basically captures *the probability being a valid phrase* of a segment.

A straightforward idea of computing $Pr(\cdot)$ is to count a segment's appearance in a very large corpus. The ideal case is that we use the entire collection of tweets published in *Twitter* to compute the $Pr(\cdot)$ for all possible segments. Unfortunately, to the best of our knowledge, such corpus never exists. Instead, we turn to Microsoft Web N-Gram corpus[4] [19]. This N-Gram corpus is based on all the documents in the World Wide Web indexed by Microsoft Bing in the EN-US market; it provides a good estimate of the statistics of commonly used phrases in English.

Another idea of computing $Pr(\cdot)$ is to look up the segments in a knowledge base where valid segments are more easily recognized. We exploit *Wikipedia* for this purpose, which is by far the largest online encyclopedia in the World Wide Web. We take a snapshot of English *Wikipedia*[5], and build a dictionary by extracting all the article titles, disambiguation pages, redirect pages (synonyms), and wikilinks [8]. If a segment matches any entry in the dictionary, it has a higher prior probability of being a true named entity.

TwiNER combines both ideas in a dynamic programming algorithm to efficiently test various segmentation combinations. Note that in this step, we do not use any local linguistic features of a segment, such as its capitalization. Instead, we leverage on the World Wide Web to derive the segmentation. For ease of presentation, information captured from the World Wide Web for a given segment is called its *global context*.

1.2 Segment Ranking

Each segment extracted in Step (1) is a candidate named entity, e.g. "youth olympic games" and "mast rudder". We now have a huge pool of candidate named entities. Undoubtedly, this pool has a high recall but a very poor precision in identifying the true named entities. For example, among the seven segments extracted in Figure 1, only "youth olympic games" can be considered as a true named entity.

Can we automatically identify the true entities from non-entities in the pool? To address this problem, we learn a function that as-

[2]"PAP" and "NSP" are short forms of political party names. Hashtags such as "#whathappentosavingtheearth" are not considered in this paper.

[3]A segment basically contains a phrase. In the rest of this paper, "segment" and "phrase" are used interchangeably.

[4]http://web-ngram.research.microsoft.com/info/

[5]http://dumps.wikimedia.org/enwiki/

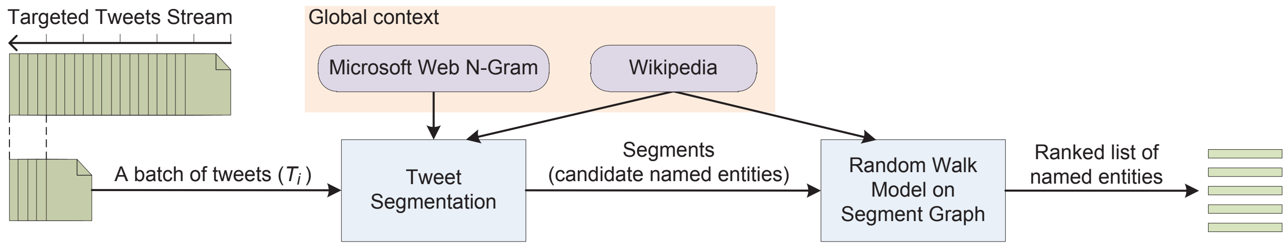

Figure 1: System Architecture of *TwiNER*

signs a confidence score of being a true named entity to each candidate named entity. Candidate named entities are then ranked according to this score. By setting a threshold, we can easily remove the long tails of non-entities with low scores.

Recall there are two types of global context for a given segment in *TwiNER*: Web N-Gram and *Wikipedia*. The former apparently has no clear clue about such score because many common word combinations with high frequency are not named entities, like "there is" and "such a". The latter, on the other hand, provides some hints because many named entities either have corresponding *Wikipedia* pages or have been referenced in *Wikipedia*. However, *Wikipedia* is not as *real-time* as *Twitter*. It usually takes a while for a new named entity appeared in tweets to be captured by *Wikipedia*. For example, in the tweets we use in the experiments, "Vincent Wijeysingha" is the name of a political figure, which appeared in tweets in the early April 2011. Before end of April 2011, there was no mention about this person at all in *Wikipedia*. Furthermore, there is also no guarantee that all named entities in tweets would appear in *Wikipedia* later.

Since the *global context* is insufficient to identify the true named entity, is there any local feature in tweets themselves that we can utilize? It is observed that there exists a *gregarious* property among the named entities in the targeted tweet stream, since the tweets in the targeted tweet stream are normally about similar or related topics/events. Formally, *gregarious* refers to the interaction of named entities with each other and to their collective co-existence in the targeted tweet stream. For example, "Barack Obama" is a named entity. It often co-occurs with other named entities like "United States" and "Michelle Obama" in a targeted stream about *United States*, but seldom co-occurs with "please look", a valid segment extracted from tweets but a non-entity. It is also uncommon that same set of non-entities appear together often.

This *gregarious* property of named entities in *Twitter* motivates us to design a "recursive" algorithm to compute the score of a segment being a named entity. The idea is: an undirected *segment graph* using all the segments extracted in Step (1) is built first, in which nodes are segments and edges are weighed proportional to the co-occurrence similarity; then, a *random walk model* is applied on this graph to derive the probability of a segment being a named entity. Because a segment's confidence is affected by its neighbors in the graph, which only depends on the tweets themselves, we call the *segment graph* as the *local context* of a segment in the tweets. Note that, not only has the *local context* been considered in this model, but also the *global context* is integrated to overcome the limitation of random walk model. Finally, the output of the model is used as the score of a segment being a named entity.

One may wonder that building *local context* (i.e. the *segment graph*) defeats the real-time nature of *Twitter*. Indeed, a buffer of tweets is necessary to construct the local *segment graph*, making *TwiNER* not completely real time (response in a "tweet by tweet" manner). Nevertheless, there are more than 2,000 tweets generated every second[6] in *Twitter*, which is already a big enough buffer to build the *local context* in Step (2). Therefore, *TwiNER* is able to give "near real-time" response practically (in a "second by second" manner).

Contributions

To sum up, we made the following contributions in the paper:

1. We proposed an unsupervised NER system without explicit human label efforts. Our system does not rely on any linguistic features, making it suitable for tweets and potentially other social text streams with unreliable linguistic features.

2. To the best of our knowledge, our *TwiNER* system is the first to exploit both the *local context* (in *tweets*) and the *global context* (from World Wide Web) together for named entity recognition task in Twitter.

3. The proposed system has been successfully evaluated on two different collections of real-life tweets, simulating two types of targeted twitter streams. A region-based stream for tweets published by users from a particular geographical region; and a topic-based stream for tweets potentially relevant for a political event.

The rest of this paper is organized as follows. A review of related work is given in Section 2. Sections 3 and 4 present the design of *TwiNER* in detail. Following that, experimental results are presented in Section 5 to evaluate the correctness and effectiveness of *TwiNER*. Finally, Section 6 concludes this paper with discussion of future work.

2. RELATED WORK

Tweets are infamous for their error-prone and short nature. This leads to failure of many conventional NLP techniques, which heavily depend on local linguistic features, such as capitalization, POS tags of previous words, etc. Also acknowledging the error-prone nature of tweets, Han and Baldwin [6] proposed to normalize ill-formed words in tweets to make the contents more formal. However, this work does not address the problem of NER. NER has attracted renewed interests recently, due to the challenges posed by tweets. Conventionally, NER studies are mainly conducted in a supervised manner. In most of the cases, they depend on the Part-of-Speech (POS) tags, which again need a tagger to be trained with supervised approach based on linguistic features[14, 20, 21].

There are attempts that design linguistic features to capture tweets' unique characteristics and train tweet-specific models. Gimpel et. al trained a POS tagger with the help of a new labeling scheme and a feature set that captures the unique characteristics of tweets [5]. It was reported to outperform the state-of-the-art Stanford

[6] *200 million Tweets per day*: `http://blog.twitter.com/2011/06/200-million-tweets-per-day.html`.

POS tagger on tweets. In [16], Ritter et. al presented an tweet-based NLP framework which contains tweet-specific NLP tools: POS tagger (*T-POS*), shallow parsing (*T-CHUNK*), capitalization classifier (*T-CAP*), and named entity recognition (*T-NER*). *T-POS* and *T-CHUNK* are trained by using condtional random field (CRF) model with conventional and tweet-specific features. These tweet-specific features include retweets, @usernames, hashtags, URLs, and Brown clustering results. Both *T-POS* and *T-CHUNK* were reported with better performance compared to the state-of-the-art methods. *T-NER* is separated into two task: named entity segmenting (*T-SEG*) and named entity classification (*T-CLASS*). *T-SEG* is trained with a CRF model. The features include orthographic, contextual, dictionary features, and the output by *T-POS*, *T-CHUNK*, and *T-CAP*. *T-CLASS* is implemented by applying Labled-LDA [13] with the external knowledge base Freebase[7]. Liu et. al [9] applied a KNN-based classifier to conduct word-level classification, leveraging the similar and recently labeled tweets. Those pre-labeled results, together with other conventional features (e.g. orthographic and lexical features), were then fed into a CRF model to conduct finer-grained NER.

Due to their supervised nature, those approaches require the availability of labeled data, which is usually expensive to come by. Finin et. al. presented a crowd-sourcing way (using services like Mechanical Turk and CrowdFlower) of preparing labeled data for NER studies in *Twitter* [3]. However, it did not propose a solution for NER.

Similar to *TwiNER*, Downey et. al also proposed a *collocation*-based approach, called *LEX* to detect the boundaries of named entities [2]. Nevertheless, it is not designed for tweet-like informal text. It assumes that named entities are either continuous capitalized words or mixed case phrases beginning and ending with capitalized words, which is apparently too strong to hold in tweets. Silva et. al. [1] studied five different types of *collocation* measurements and their variations for phrase extraction task. Besides SCP measurement used in both *TwiNER* and *LEX*, there are another four types of *collocation measure*. And SCP performs the best among others.

Wikipedia is exploited as a source of *global context* in this paper. *Wikipedia* has been utilized in many text mining and NLP tasks, such as text categorization, topic detection, etc. For NER task, *Wikipedia* is mainly used to derive the category label for phrases, including [7] and [15]. [7] only looks for phrases of no more than eight words that start with a word containing at least one capitalized letter in a sentence, and treats phrases with corresponding *Wikipedia* page as named entities. The head noun of the noun phrase just after *be* in the first sentence of the *Wikipedia* page is picked as the phrase's category. This method is not suitable for informal texts such as tweets due to its heavy dependence on local linguistic features. [15] focused on multilingual NER. It depended on the *Category* information in each English *Wikipedia* page to categorize an English named entity. For a non-English phrase, the *Category* information in its corresponding English *Wikipedia* page, if any, is used for categorization. Nevertheless, it is not clear how named entity candidates are identified in this paper [15].

Existing attempts that exploit *Wikipedia* usually assume that named entities should have corresponding *Wikipedia* pages. This assumption makes them unable to identify emerging named entities which are frequently observed in tweets. However, there are many new named entity mentioned in tweets. In *TwiNER*, information in tweets' *local context* and *global context* are aggregated to calculate the probability that a phrase is a named entity. By doing so, *TwiNER* is able to recognize new named entities which may not appear in *Wikipedia* yet. To the best of our knowledge, it is the first to exploit both the *local context* (in *tweets*) and the *global context* (from World Wide Web) together for NER task in Twitter.

[7] `http://www.freebase.com/`

3. TWEET SEGMENTATION

In this section, we detail our solution for tweet segmentation. Given an individual tweet $t \in T_i$, the problem of tweet segmentation is to split t into m consecutive segments, $t = s_1 s_2 ... s_m$; each segment contains one or more words. To obtain the optimal segmentation, we use the following objective function, where C is the function that measures the *stickiness* of a segment or a tweet defined based on word collocation:

$$\arg\max_{s_1,\dots,s_m} C(t) = \sum_{i=1}^{m} C(s_i), \tag{1}$$

A high *stickiness* score of segment s indicates that it is not suitable to further split segment s, as it breaks the correct word *collocation*. In other words, a high *stickiness* value indicates that a segment cannot be further split at any internal position.

If the word length of tweet t is l, there exists 2^{l-1} possible segmentations. It is inefficient to iterate all of them and compute their stickiness. We therefore design a dynamic programming algorithm to tackle the problem, which is presented in the following.

3.1 A Dynamic Programming Algorithm

Algorithm 1 outlines our dynamic programming algorithm for tweet segmentation. The basic idea is to recursively conduct binary segmentations and then evaluate the *stickiness* of the resultant segments. More formally, given any segment s from t (s can be t itself or a part of t) and suppose $s = w_1 w_2 ... w_n$, our solution is to conduct a binary segmentation by splitting it into two adjacent segments $s^1 = w_1 ... w_j$ and $s^2 = w_{j+1} ... w_n$ by satisfying:

$$\arg\max_{s^1,s^2} C(s) = C(s^1) + C(s^2). \tag{2}$$

The complexity of Algorithm 1 is $O(lue\log(ue))$, where l is the average tweet length, u is the upper bound of segment length, and e bounds top sub-segments of a segment. Long segments are rare in tweets because each tweet is limited to 140 characters. We observed that in our data, $u = 5$ is a proper bound as the maximum length of a segment, which largely reduces the number of possible segmentations. We also set $e = 5$ so that the segmentation only focuses on top-quality segments and are not stuck by trivial ones, which leads to a complexity of $O(l)$.

3.2 Segment Stickiness Function

In Algorithm 1, one key factor is the *stickiness* function C. A high *stickiness* score of segment s indicates that further splitting segment s would break the correct word *collocation*. There are a number of *collocation* measurements [10, 12]. However, all these measures were defined for two arguments. That is, they were designed to measure the collocation of the bigram or the n-grams with the particular binary partition. A variety of studies have been conducted to extend these binary collocation measures to the n-grams case (where n is greater than 2) [1, 2, 17]. We define the *stickiness* functions by using the generalization framework proposed in [1]. Specifically, the generalized collocation measures of Point Mutual Information (PMI) and Symmetric Conditional Probability (SCP) are studied here.

Algorithm 1: Tweet Segmentation

input :
A tweet: $t = w_1 w_2 ... w_l$;
u: the maximum length of a segment s;
e: top e segmentations set $\mathbb{S}$ for each segment s;
output:
An optimal tweet segmentation $t = s_1 s_2 ... s_m$;

```
for i = 1 : l do
    initialize a set S_i = {} to store possible segmentation of segment s_i = w_1 w_2 ... w_i;
    if i <= u then
        /* do not split s_i */
        calculate C(s_i);
        add s_i to S_i as a possible segmentation of s_i;
    /* try different possible ways to segment s_i */
    for j = 1 : i − 1 do
        if i − j <= u then
            form two shorter segments of s_i: s_i^1 = w_1 ... w_j and s_i^2 = w_{j+1} ... w_i;
            calculate C(s_i^2);
            foreach S_j ∈ S_j do
                /* S_j contains the top e possible segmentations of s_i^1, and S_j is one of them; */
                concatenate S_j and s_i^2 to form a new segmentation S of s_i;
                add S to S_i;
                C(S) = C(S_j) + C(s_i^2)
        Sort S_i and keep only the top e segmentations;
return S ∈ S_l with the highest score as the optimal segmentation;
```

3.2.1 PMI based Stickiness

PMI measures the degree that two words occur together more often than by chance. Mathematically, PMI for bigram $w_1 w_2$ is defined as follows:

$$PMI(w_1 w_2) = \log \frac{\Pr(w_1|w_2)}{\Pr(w_1)} = \log \frac{\Pr(w_1 w_2)}{\Pr(w_1)\Pr(w_2)} \tag{3}$$

Given a segment, $s = w_1 ... w_n$, PMI is then extended by averaging all binary partitions as follows:

$$PMI(s) = \log \frac{\Pr(w_1 ... w_n)}{\frac{1}{n-1}\sum_{i=1}^{n-1} \Pr(w_1 ... w_i)\Pr(w_{i+1} ... w_n)} \tag{4}$$

If segment s only contains one word w, we have $PMI(s) = \log \Pr(w)$.

Note that PMI defined above falls into the range of $(-\infty, +\infty)$. The *stickiness* of segment s is then defined by mapping the value of Equation 4 to the range of $[0, 1]$ as follows:

$$C(s) = \frac{1}{1 + e^{-PMI(s)}} \tag{5}$$

3.2.2 SCP based Stickiness

Symmetrical Conditional Probability (SCP) was proposed in [1] to measure the "cohesiveness" of bigram $w_1 w_2$ by considering both conditional probabilities for the bigram given each single term:

$$SCP(w_1 w_2) = \Pr(w_1 w_2|w_1)\Pr(w_1 w_2|w_2) = \frac{\Pr(w_1 w_2)^2}{\Pr(w_1)\Pr(w_2)} \tag{6}$$

Given a segment, $s = w_1 ... w_n$, SCP of s is defined similarly as follows:

$$SCP(s) = \frac{\Pr(s)^2}{\frac{1}{n-1}\sum_{i=1}^{n-1} \Pr(w_1 ... w_i)\Pr(w_{i+1} ... w_n)} \tag{7}$$

Here, we smooth SCP value by taking logarithm calculation, Equation 7 is then updated as follows:

$$SCP(s) = \log \frac{\Pr(s)^2}{\frac{1}{n-1}\sum_{i=1}^{n-1} \Pr(w_1 ... w_i)\Pr(w_{i+1} ... w_n)}, \quad SCP(s) \in (-\infty, 0) \tag{8}$$

Similarly, we define SCP for any segment s of unit length as $SCP(s) = 2\log \Pr(w)$. We then define the *stickiness* of s by using the sigmoid function as follows:

$$C(s) = \frac{2}{1 + e^{-SCP(s)}} \tag{9}$$

3.3 Enhanced Stickiness by World Wide Web

By now, the calculation of the stickiness is reduced to estimating $\Pr(s)$, $\Pr(s^1)$, and $\Pr(s^2)$ for any segment $s \subset t$, which are prior probabilities of segments. To accurately estimate these prior probabilities, we need a large enough corpus as the *global context* of each segment. The ideal global context is the entire collection of tweets published in *Twitter*. But unfortunately, to the best of our knowledge, such corpus is not available.

Instead, we exploit the one provided by Microsoft Web N-Gram Services [19] as approximation. This corpus is based on the web documents indexed by Microsoft Bing search engine in the EN-US market. The spam and other low quality documents are excluded. Each indexed document is parsed, tokenized, and the text is lower-cased with the punctuations removed. No stemming, spelling correction or inflections are performed [19], which provides a large enough English corpus to estimate prior probabilities of segments.

One problem of segmentation based on the lexical statistics derived from such corpus is its preference towards frequent patterns. Figure 3 illustrates such an example, with a portion of tweet and three possible segmentations.

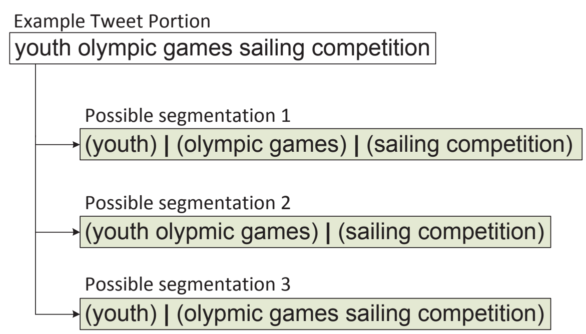

Figure 3: Different Segmentations of a Portion of Tweet

If only Web documents are used as a priori knowledge, then "*youth olympic games sailing competition*" would be segmented into "*youth*" and "*olympic games sailing competition*" (*i.e.,* the possible segmentation 3 in the figure), because both "*youth*" and "*olympic games sailing competition*" are frequent in Web documents. Nevertheless, this tweet is in fact referring to "Youth Olympic Games" held in Singapore in 2010.

We therefore leverage a knowledge base in the World Wide Web, *Wikipedia*, as another source of *global context* to tackle this problem. There are several reasons for the choice of *Wikipedia*. It provides rich a priori knowledge about entity information and is publicly available. Article titles, references to other *Wikipedia* pages, and the disambiguation pages, have often been used as named entity candidates [7, 15]. In detail, we build a large *Wikipedia* dictionary by extracting from a snapshot of *Wikipedia* on January 30, 2010 all the English article titles, disambiguation pages, redirect pages as well as hyperlinks [8]. Articles isolated from the rest are removed. Finally, we have a *Wikipedia* dictionary of 4,342,732 entries as well as their polysemes, synonyms.

Let $Q(s)$ be the probability that s appears as anchor text in its mentioning *Wikipedia* article, which is the number of *Wikipedia* articles containing s as anchor text divided by the number of *Wikipedia* articles s appears in. A segment appearing as anchor text with a high probability in *Wikipedia* is a strong indication that it is a valid name entity[8]. The *stickiness* function is now defined as follows:

$$C'(s) = C(s) \cdot e^{Q(s)} \tag{10}$$

$C(s)$ is measured by exploiting the *global context* captured in Microsoft Web N-Gram. The second component of Equation 10 is introduced so that segments appearing in Wikipedia as anchor texts are attributed with higher *stickiness*.

3.4 Length Normalization

So far, we treat segments of various lengths equally. However, in Twitter NER task, there are only a few long named entities. And it is observed that longer valid segments have higher chances of being named entities than shorter ones. Note that the Web N-gram data has already had a strong preference for short segments. Given this, we introduce length normalization $\mathcal{L}(s)$ to Equation 10 to favor moderately long segments in both Web N-gram and *Wikipedia*. Finally, we have the following *stickiness* function:

$$C''(s) = \mathcal{L}(s) \cdot e^{Q(s)} \cdot C(s) \tag{11}$$

The length normalization factor $\mathcal{L}(s)$ is empirically defined as:

$$\mathcal{L}(s) = \begin{cases} \frac{|s|-1}{|s|}, \text{ for } |s| > 1 \\ 1, \text{ for } |s| = 1 \end{cases} \tag{12}$$

4. SEGMENT RANKING

Section 3 extracts a large number of segments which are valid in the sense of word collocation by leveraging segments' *global context* in the World Wide Web. However, not all segments extracted are named entities. For example, "please look" is a valid segment but not a named entity. In this section, a set of strategies have been developed to rank segments according to their likelihood of being named entities.

4.1 Noise Filtering

We first remove three types of segments which are obviously *not* named entities.

1. Segments containing well-known slang words, e.g. "lol" and "tmr". We use a compilation of Internet slangs provided by `http://www.noslang.com/dictionary/full`.

2. Segments containing words with consecutive repeating characters, e.g."hahahaha", "nooooo", and "gooooooood". These words are frequently used in tweets to represent exaggerative emotions. Regular expressions are used to recognize them.

3. Segments containing words with "#" as prefix. Those words are usually *hashtags* in tweets, which are created by users to highlight keywords/topics. They are not considered named entities in this paper, and can be easily removed by locating the "#" prefix.

4.2 Local Segment Graph

The *global context* alone is insufficient to recognize a named entity. We therefore utilize the *local context* of a segment in tweets to tackle this problem. One intuitive solution is to weigh a segment using its frequency in tweets. However, this method would wrongly favor phrases like "please look", which is not only frequent in World Wide Web, but also frequent in Twitter.

As we discussed in Section 1, there exists a *gregarious* property among named entities in targeted tweet streams. A good example is "Barack Obama". It is a true named entity, and it often co-occurs with other named entities like "United States" or "Michelle Obama" in tweets, but seldom co-occurs with "please look", a valid segment but non-entity. It is also uncommon that same set of non-entities would appear together often.

Based on this property, we propose a "segment graph". At the ith interval (recall that *TwiNER* recognizes named entities in a batch mode), we build an undirected *segment graph* $G(V, E)$ using all segments V extracted from the tweet set T_i on the fly. In this graph, each node is a valid segment after noise filtering, and the edge $e_{ab} \in E$ between two nodes (segments) s_a and s_b is weighed by the *Jaccard Index*:

$$w_{ab} = w(e_{ab}) = \frac{|M(s_a) \cap M(s_b)|}{|M(s_a) \cup M(s_b)|}, \tag{13}$$

where $M(s)$ is the set of tweets in T_i containing segment s.

The *segment graph* $G(V, E)$ provides a good *local context* for each segment in T_i. It does not use the unreliable local linguistics features of tweets but relies on the relations among segments. Because all segments have been parsed once by their *global context* and then filtered with heuristic rules, these relations are relatively more reliable than local linguistics features.

4.3 Random Walk Model

A *random walk model* is then applied on graph $G(V, E)$ to compute the stationary probability of each segment being a true named entity, by considering the graph bidirectional. While random walking, the probability of transiting from node s_a to node s_b (denoted as P_{ab}) is given by

$$P_{ab} = \frac{w_{ab}}{\sum_{c \in V} w_{ac}}. \tag{14}$$

All transition probabilities are then aggregated to form a nonnegative transition matrix P for the whole graph.

To overcome the "dangling links" while conducting a random walk on graph $G(V, E)$, a teleportation vector $\mathbf{e}$ is also introduced to make the random walker jump from a node to any other node in the *segment graph* with a small probability [11]. We observe that $Q(s)$, the probability that s appears as anchor text in *Wikipedia*, is a good teleportation a priori. In other words, we favor those segments that are valid hyperlinks in *Wikipedia*, i.e. those segments are more likely to be named entities. Accordingly, we define the following teleportation probability for node (segment) s:

$$\mathbf{e}_s = \frac{Q'(s)}{\sum_{s_j \in V} Q'(s_j)}, \ \textit{where } Q'(s) = e^{Q(s)}. \tag{15}$$

The exponential function is used here to avoid the situation that the segment is *new* to *Wikipedia* so that its $Q(s) = 0$ and will never be teleported to.

The *Wikipedia*-based teleportation can be considered as an injection of global context into the random walk model of the local context. With the transition matrix P and the *Wikipedia*-based teleportation vector $\mathbf{e}$, the stationary eigenvector π^T of P is calculated iteratively using power method as below:

$$\pi = (\gamma P^T + (1-\gamma)\mathbf{e}\mathbf{1}^T)\pi, \tag{16}$$

where γ controls the probability of teleportation. The lower γ is, the higher probability the random walk will teleport according to $\mathbf{e}_s$.

Then, π^T is used as probabilities of segments being named entities in the local context.

Finally, for balancing the advantages of *global context* and *local context*, given an input segment s, *TwiNER* multiples its stationary probability $\pi(s)$ in the *segment graph* mainly learned from the local context with its teleportation probability learned from Wikipedia:

$$y(s) = \mathbf{e}_s \cdot \pi(s). \quad (17)$$

Equation 17 is used to rank all segments, and only the top K segments are retained as named entities.

5. EVALUATION

In this section, we conduct extensive experiments to evaluate *TwiNER*. In Section 5.1, datasets simulating two targeted twitter streams are described, and performance metrics are introduced. Section 5.2 compares *TwiNER* with existing methods. We then present a performance analysis of *TwiNER* in different settings in Section 5.3.

5.1 Tweets Data and Performance Metrics

Tweets collections. Two collections of tweets are used in the experiments to simulate targeted twitter streams.

The first collection (*SIN*) was collected to simulate targeted twitter stream of one particular geolocation by monitoring a number of Singapore-based[8] users' tweets published in June 2010. The set of users to be monitored was populated by first getting the top-1000 Singapore-based *Twitter* users with the most number of followers from `http://twitaholic.com`, and then expanding the list by including the top users' followers and friends in *Twitter* within two hops. There are a number of real-life events in the data collection period, such as the flash flood in Orchard Road (a premium shopping belt in Singapore), FIFA World Cup 2010 and WWDC 2010, etc. Collection *SIN* contains $4,331,937$ tweets.

The second collection (*SGE*) was collected to simulate targeted twitter stream of one particular event by monitoring a set of predefined keywords related to Singapore General Election 2011. Similar to collection *SIN*, only tweets published by Singapore-based users were collected. The data collection started on April 13, 2011 and ended on May 13, 2011, which covered the duration of Singapore General Election 2011 (nomination day on April 27, 2011, and polling day on May 7, 2011). Collection *SGE* contains $226,744$ tweets.

It is observed that, by collecting tweets based on users, topics covered in collection *SIN* are diverse in nature. Topics covered in collection *SGE*, on the other hand, are more concentrated since most of the discussions are about the general election. Another observation is that, twitter users are more formal in political discussions than casual discussions. In other words, tweets in *SGE* are more formal than those in *SIN*.

Manual Annotation. For both collections, $5,000$ tweets are randomly sampled from the tweets published on one random day. Each tweet is then labeled by two human annotators, who have strong background knowledge about Singapore-related named entities. The BILOU schema is used [9, 14]. After discarding retweets and tweets with inconsistent annotations, we get $4,422$ tweets for *SIN* and $3,328$ tweets for *SGE*. We denote these two randomly sampled tweets collections with groundtruth labeling as *SIN_g* and *SGE_g* respectively. Observe that the annotating agreement is relatively low for collection *SGE*. This is mainly due to the disagreement between the human annotators about the way how to handle the concept *GRC*[9] and *SMC*[10], which refer to different types of electoral divisions in Singapore. Annotators did not have an agreement on whether a GRC/SMC should be labeled as part of a location name. For example, some may label "*aljunied grc*" as "*<U>aljunied</U> grc*", while some may label as "*<B>aljunied</B> <L>grc</L>*".

Performance Metric. Performance metrics used throughout the experiments include: Precision(*Prec*), Recall(*Recall*), and *F1*. *Prec* quantifies the percentage of the extracted phrases that are true named entities. *Recall* quantifies the percentage of the true named entities that are correctly recognized. *F1* is the harmonic mean of *Prec* and *Recall*, *i.e.,* $F1 = 2 \cdot \frac{Prec \cdot Recall}{Prec + Recall}$.

Note that different values of K (the parameter in the *segment ranking* step) would result in different performance of *TwiNER*: larger K will increase *Recall* but decrease *Prec*, and vice versa. *TwiNER*'s performance reported in the following sections are the ones with the highest *F1* score *TwiNER* can achieve using SCP-based *stickiness* function, and various values of K. For a fair comparison, K is set to be larger than 50 (*i.e.,* $K > 50$). The maximum iteration for the random walk is fixed at 500.

5.2 Comparison with Other Methods

In this section, we compare *TwiNER* with two conventional NER systems trained on tweets. Specifically, we train Stanford-NER and LBJ-NER with the labeled tweet data and evaluate their performance[11]. Moreover, we also compare with a tweet-specific NER system (T-NER)[12] proposed in [16] on the two tweet collections.

- LBJ-NER: A NER system based on the regularized averaged perceptron approach which uses gazetteers extracted from *Wikipedia*, word class models derived from unlabeled text, and expressive non-local features [14]. It is reported to have achieved the best result (F1-Measure of 0.908) on the CoNLL 2003 test set.
- Stanford-NER: A NER system based on CRF model which incorporates long-distance information [4]. It achieves good performance consistently across different domains.
- T-NER: a supervised NER system uses CRF model for learning and inference. A set of widely-used effective features are used in T-NER, including orthographic, contextual, and dictionary features [16].

Note that, other than the proposed *TwiNER*, the three methods listed above (*i.e.,* LBJ-NER, Stanford-NER and T-NER) are supervised methods and require labeled examples. For performance comparison, we randomly split both *SIN_g* and *SGE_g* in the ratio of $6:4$ as training and evaluation sets. Stanford-NER and LBJ-NER are trained with default feature settings[13]. While LBJ-NER requires development set for the parameter tuning, we further split the training set in the ratio of $5:1$ for training and development. All the methods are evaluated on the same evaluation set.

[8] A user is considered Singapore-based if she specifies Singapore in the *location* field of her profile.

[9] `http://en.wikipedia.org/wiki/Group_Representation_Constituency`

[10] `http://en.wikipedia.org/wiki/Single_Member_Constituency`

[11] Note that as the entity type classification is not the focus of this paper, we do not differentiate the entity types when both LBJ-NER and Stanford-NER are trained.

[12] `https://github.com/aritter/twitter_nlp`

[13] We use the same settings for training as in the CoNLL-2003 shared task. `http://www.cnts.ua.ac.be/conll2003/ner/`

Table 2: Different NER systems' performance on tweets

System	Dataset	Prec	Recall	F1
LBJ-NER	*SGE_g*	0.933	0.595	0.727
Stanford-NER	*SGE_g*	**0.950**	**0.880**	**0.913**
T-NER	*SGE_g*	0.713	0.359	0.478
TwiNER	*SGE_g*	0.929	0.660	0.772
LBJ-NER	*SIN_g*	**0.764**	0.265	0.393
Stanford-NER	*SIN_g*	0.762	0.293	0.423
T-NER	*SIN_g*	0.429	**0.509**	**0.466**
TwiNER	*SIN_g*	0.419	0.329	0.419

Table 2 shows the evaluation results of different NER systems. It can be observed from Table 2 that:

1. As observed in Table 2, the overall performance on *SGE_g* is much better than on *SIN_g* by all methods. For the case of LBJ-NER and Stanford-NER, the main reason is the adverse impact of error-prone local context of tweets, since *SIN_g* has more tweets of informal style. This results in a low quality of training set based on the linguistic features. *TwiNER* extracts named entities by exploiting the *local context*. Since *SIN_g* is collected by monitoring twitter users of Singapore, diverse topics are collected. This makes *gregarious* property relatively weaker in this collection, which results in the relatively lower performance of *TwiNER* on *SIN_g*.

2. T-NER performs consistently across the two evaluation sets. Since it was developed by taking into consideration tweets' error-prone context, it outperforms Stanford-NER and LBJ-NER and achieves the best performance on *SIN_g* in terms of $F1$. However, the decrease of *Prec* on *SIN_g* from that on *SGE_g* ($0.713 \rightarrow 0.429$) indicates that it still suffers a lot from tweets' error-prone nature.

3. No system outperforms the others on both collections. Stanford-NER performs much better the other systems on *SGE_g*, with F1 performance more than 0.91. However, it only achieves a slightly better $F1$ performance than LBJ-NER and *TwiNER* on *SIN_g*. LBJ-NER performs the worst on *SIN_g* and the third best on *SGE_g* in terms of $F1$. Moreover, the supervised system evaluated here, Stanford-NER, LBJ-NER, and T-NER, achieve relatively lower *Recall* across both collections. This indicates that supervised methods relying on local linguistical features may not generalize well for tweet's error-prone and dynamic nature.

4. *TwiNER*, an unsupervised approach, achieves comparable $F1$ performance with the other supervised systems. *TwiNER* performs much better than LBJ-NER and T-NER on *SGE_g*. It also outperforms LBJ-NER and achieves a comparable performance with Stanford-NER on *SIN_g* in terms of $F1$, where *gregarious* property is relatively weak.

As this set of experiments show, performance of LBJ-NER and Stanford-NER deteriorate significantly when they are applied on tweets, due to tweets' error-prone context. To further illustrate the adverse impacts of tweets' error-prone context, we trained LBJ-NER and Stanford-NER with formal text corpus using the CoNLL-2003 shared task data, and then directly apply the trained models on the evaluation sets of *SIN_g* and *SGE_g*. To distinguish the methods trained using tweet data, we denote the two methods trained on formal text LBJ-NER^F and Stanford-NER^F respectively.

Table 3: Conventional NER systems' performance on tweets

System	Dataset	Prec	Recall	F1
LBJ-NER^F	*SGE_g*	0.674	0.400	0.502
Stanford-NER^F	*SGE_g*	0.691	0.642	0.666
TwiNER	*SGE_g*	**0.929**	**0.660**	**0.772**
LBJ-NER^F	*SIN_g*	0.314	0.314	0.314
Stanford-NER^F	*SIN_g*	0.256	**0.461**	0.329
TwiNER	*SIN_g*	**0.419**	0.329	**0.419**

The results of LBJ-NER^F, Stanford-NER^F and *TwiNER* are summarized in Table 3. It is observed that the performance of LBJ-NER^F and Stanford-NER^F deteriorate significantly with F1-measure of lower than 0.5 on both *SIN_g* and *SGE_g*.

5.3 Analysis of TwiNER

In this section, we investigate the impact of different *TwiNER* components on its performance.

5.3.1 Performance of Tweet Segmentation

Tweet segmentation is used to extract the named entity candidates from tweets, or in other words, to identify the correct boundary of potential named entities in tweets. It is a critical component because the performance of *TwiNER* is heavily affected by the effectiveness of tweet segmentation.

Two *stickiness* functions are defined by using two collocation measures, PMI and SCP, for tweet segmentation. The tweet segmentation algorithm described in Section 3 also incorporates an external knowledge base *Wikipedia*. Further, we normalize the segment length to favor long named entities. In this section, we study the impact of the collocation measures (*PMI* or *SCP*), the *Wikipedia* dictionary (*Wiki*), and the length normalization (*Norm*), based on the ground truth in *SIN_g* and *SGE_g*. We use tweet segmentation with only PMI or SCP measures as the baseline (Equation 5 and 9). We measure the percentage of named entities that are correctly extracted (i.e. split as a segment) as the performance metric, which is denoted as *Prec* as well. The experimental results are listed in Table 4. From Table 4, we observe that:

1. *SCP* significantly outperforms *PMI* for tweet segmentation. We believe this is because PMI returns disproportionately high values for frequent items. This property makes PMI prefer longer segments, which is confirmed by manual investigation of the segmentation result. For example, we observe that "*sdp*" cannot be extracted from "*vote sdp*" by PMI, since PMI-based *stickiness* returns 0.397 for "*vote sdp*", and only 0.017 and 0.004 for "*vote*" and "*sdp*". While SCP-based *stickiness* returns only $5.76E-5$ for "*vote sdp*", and $6.51E-4$ for "*vote*"+"*sdp*".

2. Length normalization (*Norm*) is effective and improves the accuracy of tweet segmentation in the two collections for SCP-based *stickiness*. For example, given a long segment like "*java programming language*", *SCP+Norm* is able to fairly treat it as a named entity, while *SCP* split it as "*java*" and "*programming language*". Since PMI prefers longer segments, further preference introduced by *Norm* does aggravate the problem.

3. *Wikipedia*'s broad coverage and high quality knowledge help reclaim incorrect decisions made by the lexical statistics or reinforce the correct decisions. For instance, *SCP* cannot extract "*pap*" from "*pap team*" given the latter is a frequent

Table 4: Impacts of Tweet Segmentation by SCP

Scheme	Prec@*SIN_g*	Prec@*SGE_g*
SCP	0.721	0.830
SCP+Norm	0.737	0.861
SCP+Wiki	0.739	0.841
SCP+Norm+Wiki	**0.758**	**0.874**
PMI	0.317	0.288
PMI+Norm	0.297	0.288
PMI+Wiki	**0.344**	**0.319**
PMI+Norm+Wiki	0.332	0.308

phrase. With the priori knowledge from *Wikipedia*, where "*pap*" has a $Q(s)$ value of 0.181, "*pap*" is successfully extracted by *SCP+Wiki*.

4. The combination of *Wikipedia* dictionary and length normalization further boosts up the performance of tweet segmentation of SCP-based *stickiness*. This indicates that *Wiki* and *Norm* are complementary to each other. For example, we observe a tweet part "*pap give free iphones*" in a tweet. All of *SCP*, *SCP+Wiki*, and *SCP+Norm* fail to split "*free*" and "*iphones*" apart due to the frequent usage of "*free iphones*" in web documents. However, by combining the both *Wiki* and *Norm*, "*iphones*" is successfully extracted. Also, positive improvement from *PMI+Norm+Wiki* compared to *PMI+Norm* shows the ability of reclaiming incorrect decisions by exploiting *Wikipedia* dictionary.

5.3.2 Impact of Random Walk on Segment Ranking

A *random walk* model is applied to exploit the *gregarious* property of named entities in tweets. The final segment ranking output is an aggregation from the stationary probability of the *random walk* model (*local context*) and the segment's *Wikipedia*-based teleportation *Wiki* probability (*global context*). We analyze their impact on the performance of segment ranking in this section. Specifically, we investigate the following schemes for segment ranking:

- *MFS*: A naive method that ranks the segments based on their frequency in the collection. That is, the most frequent segments are ranked higher.
- *Wiki*: A naive method that ranks the segments based on their *Wikipedia*-based teleportation probability.
- *RW*: A simple random walk with uniform teleportation. The segments are then ranked based on the stationary probability $\pi(s)$.
- *RWW*: A random walk with *Wikipedia*-based teleportation. The segments are then ranked based on the stationary probability $\pi(s)$.
- *RWW+Wiki*: A random walk with *Wikipedia*-based teleportation, while the segments are ranked based on Equation 17.

Table 5 lists the experimental results based on the ground truth in *SIN_g* and *SGE_g*. From Table 5, it can be seen that:

1. While MFS works considerably well on *SGE_g*, its performance degrades significantly on *SIN_g*. The tremendous difference in performance is mainly due to the difference between the nature of the two collections *SGE_g* and *SIN_g*. As discussed earlier, *SGE_g* are tweets published in one day during Singapore General Election 2011. Thus, most of the tweets are about the election and related named entities frequently appear in these tweets. Thus, MFS achieves decent performance. However, MFS hardly recognizes any named entities due to the diverse topics covered on *SIN_g*. This also reflects the degrees of *gregarious* property of the two collections.

Table 5: Impacts of *local context*, *global context* and *random walk* on segment ranking

Scheme	Prec@*SIN_g*	Recall@*SIN_g*	F1@*SIN_g*
MFS	0.039	**0.753**	0.074
Wiki	0.433	0.398	0.415
RW	0.039	0.752	0.074
RWW	0.048	0.336	0.084
RWW+Wiki	**0.576**	0.335	**0.423**
	Prec@*SGE_g*	Recall@*SGE_g*	F1@*SGE_g*
MFS	0.736	0.565	0.639
Wiki	0.738	**0.688**	0.712
RW	0.856	0.536	0.659
RWW	**0.985**	0.536	0.694
RWW+Wiki	0.929	0.646	**0.762**

2. *RW* outperforms *MFS* significantly on *SGE_g* in terms of $F1$. This validates our assumption that named entities are more likely to co-occur with one another than with non-entities. Since *SIN_g* is very diverse, *gregarious* property is too weak to improve the segment ranking. *RWW* significantly outperforms MFS and RW in terms of *Prec* and $F1$ in *SIN_g* and *SGE_g*. The *Wikipedia*-based teleportation priori positively influences the random walk process.

3. An impressive performance is achieved by *Wiki*. It obtains the second best performance in both of the two collections, in terms of $F1$ measure. *RWW+Wiki* achieves the best performance in both *SIN_g* and *SGE_g* in terms of $F1$. We can see that it obtains a large improvement for *Prec*. We believe that the usage of *Wikipedia*-based priori in Equation 17 is the main contributing factor for the improvement of *Prec* compared to *Wiki*. Furthermore, the random walk with *Wikipedia*-based teleportation boost up many named entities that may not be covered by *Wikipedia*, by leveraging the co-occurrence of local context. This results in further improvement of *Prec*.

5.3.3 Impact of K Value

TwiNER's performance is related to the choice of K. Larger K will increase *Recall* and decrease *Prec*, and vice versa. In reality, what matters is rather how many real named entities are in the top list, so that the user can gain direct understanding on what the targeted tweets/Twitter users are concerning about. Thus, we calculate *Prec@K* for each K value from 1 to 200[14]. Here, *Prec@K* is the percentage of top K segments returned by *TwiNER* that are real named entities. Figure 4 shows *Prec@K* curves of *TwiNER* on *SGE_g* and *SIN_g*. Major proportion of segments returned when $K < 50$ are true named entities. *TwiNER* achieves a stable *Prec* performance when K is in the range of $[50, 100]$. After 100, *Prec@K* starts to degrade slowly on *SGE_g*, while *Prec@K* is still stable at

[14]We believe $K <= 200$ is a good range for the tweet collections we studied here.

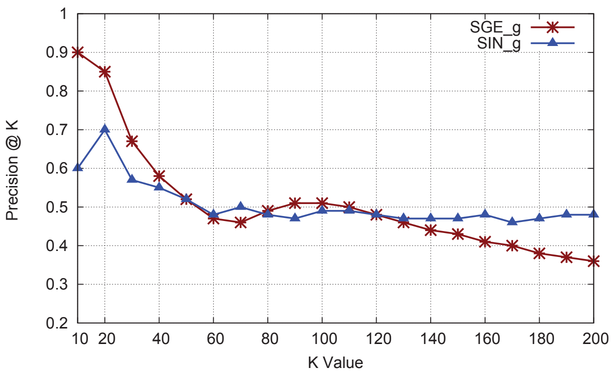

Figure 4: Precision of top *K* named entities

around 0.5 on *SIN_g*. A good strategy of choosing the suitable *K* value in various scenarios is planned for future work. Nevertheless, it is observed that the choice of *K* value depends on the nature of targeted tweet streams, such as topic cohesiveness, *gregarious* property, and size of the tweet collections.

Based on the extensive experiments conducted above, we see that by incorporating the encoded intelligence of World Wide Web and local context of tweets, *TwiNER* shows a promising performance. It provides an unsupervised approach for named entity recognition for Twitter, especially for targeted tweet streams with high *gregarious* property.

6. CONCLUSIONS AND FUTURE WORK

Twitter, as a new type of social media, has attracted great interests from both industry and academia. Many private and/or public organizations have been reported to monitor *Twitter* stream to collect and understand users' opinions about the organizations. Nevertheless, it is practically infeasible and unnecessary to listen and monitor the whole *Twitter* stream, due to it extremely large volume. Therefore, targeted *Twitter* streams are usually monitored instead. Targeted *Twitter* stream is usually constructed by filtering tweets with user-defined selection criteria. There is also an emerging need for early crisis detection and response with such target stream.

Nevertheless, the error-prone and short nature of *Twitter* has brought new challenges to named entity recognition. In this paper, we present a NER system for targeted *Twitter* stream, called *TwiNER*, to address this challenge. Unlike traditional methods, *TwiNER* is unsupervised. It does not depend on the unreliable local linguistics features. Instead, it aggregates information garnered from the World Wide Web to build robust *local context* and *global context* for tweets. Experimental results show promising results of *TwiNER*. It is also shown to achieve comparable performance with the state-of-the-art NER systems in real-life targeted tweet streams.

Despite its promising results, there is still space for improvement. First of all, we plan to study *TwiNER*'s performance in a larger scale. Second, we plan to study the strategy to identify suitable *K* value. Last but not least, this paper does not address the problem of entity type classification. As discussed earlier, this is because we feel this problem is not as pressing as the problem to correctly locate and recognize presence of named entities in tweets, which existing methods largely fail. Extension of *TwiNER* for entity type classification is also planed for future work.

7. REFERENCES

[1] J. F. da Silva and G. P. Lopes. A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. In *Proc. of the 6th Meeting on Mathematics of Language*, 1999.

[2] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *Proc. of IJCAI*, 2007.

[3] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze. Annotating named entites in twitter data with crowdsourcing. In *Proc. of the Workshop on Creating Speech and Language Data With Mechanical Turk at NAACL-HLT*, 2010.

[4] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*, 2005.

[5] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proc. of ACL*, 2011.

[6] B. Han and T. Baldwin. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proc. of ACL*, 2011.

[7] J. Kazama and K. Torisawa. Exploiting wikipedia as external knowledge for named entity recognition. In *Proc. of EMNLP-CoNLL*, 2007.

[8] C. Li, A. Sun, and A. Datta. A generalized method for word sense disambiguation based on wikipedia. In *Proc. of ECIR*, 2011.

[9] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *Proc. of ACL*, 2011.

[10] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[12] P. Pecina and P. Schlesinger. Combining association measures for collocation extraction. In *Proc. of the COLING/ACL on Main conference poster sessions*, 2006.

[13] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora. In *Proc. of EMNLP*, 2009.

[14] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proc. of CoNLL*, 2009.

[15] A. E. Richman and P. Schone. Mining wiki resources for multilingual named entity recognition. In *Proc. of ACL*, 2008.

[16] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *Proc. of EMNLP*, 2011.

[17] P. Schone and D. Jurafsky. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proc. of EMNLP*, 2001.

[18] J. Silva, Z. Kozareva, V. Noncheva, and G. Lopes. Extracting named entities. a statistical approach. In *Proc. of TLAN*, 2004.

[19] K. Wang, C. Thrasher, E. Viegas, X. Li, and P. Hsu. An overview of microsoft web n-gram corpus and applications. In *Proc. of NAACL-HLT*, 2010.

[20] Y. Wang. Annotating and recognising named entities in clinical notes. In *Proc. of the ACL-IJCNLP 2009 Student Research Workshop*, 2009.

[21] G. Zhou and J. Su. Named entity recognition using an hmm-based chunk tagger. In *Proc. of ACL*, 2002.