TwiNER: Named Entity Recognition in Targeted Twitter Stream

Chenliang Li\textsuperscript{*}\textsuperscript{1}, Jianshu Weng\textsuperscript{2}, Qi He\textsuperscript{3}, Yuxia Yao\textsuperscript{2}, Anwitaman Datta\textsuperscript{1}, Aixin Sun\textsuperscript{1}, and Bu-Sung Lee\textsuperscript{1,2}

\textsuperscript{1} School of Computer Engineering, Nanyang Technological University, Singapore
\{lich0020, anwitaman, aksun, ebslee\}@ntu.edu.sg
\textsuperscript{2} Services Platform Lab, HP Labs, Singapore
\{jianshu.weng, yuxia.yao, francis.lee\}@hp.com
\textsuperscript{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.

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\textsuperscript{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 characters)

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

\textsuperscript{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.
PAP ya la!! some of them gg to

1.1 Tweet Segmentation

has two main components, namely
to this paper.

TwiNER.

More formally, let
gerrecognition of named entities collectively from a batch of tweets in un-

ters). They often contain grammatical errors, misspellings, and un-

Table 1: Example Named Entities in Tweet

<table>
<thead>
<tr>
<th>Tweet</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PAP POSTERS ARE EVERYWHERE! AND FOR SOME LAMP POLES THERE ARE BOTH NSP AND PAP POSTERS! #whathappentosavingtheearth</td>
<td></td>
</tr>
<tr>
<td>2 ya la!! some of them gg to potong pasir. I’m gg to yio chu kang</td>
<td></td>
</tr>
</tbody>
</table>

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”) 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 Tweet stream within one fixed time interval (e.g. a second). So, T = {T1, T2, ..., Ti} and Ti is the batch of tweets posted in the ith interval. TwiNER then recognizes all possible named entities in Ti, 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.

Example Tweet

My shoes are gg to compete in the youth olympic games sailing competition. It just needs a mast and a rudder

Example Segmentation

(shoes) { (gg) } (complete) { (youth olympic games) } (sailing competition) { (needs) } (mast rudder)

Figure 2: Example of Tweet Segmentation

The idea is to segment an individual tweet into a sequence of consecutive phrases, 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 w1w2w3w4, we segment it as w1w2[w3w4] rather than w1[w2w3w4]. If C(w1w2) + C(w3w4) > C(w1) + C(w2w3w4), where C(·) basically captures the probability being a valid phrase of a segment.

A straightforward idea of computing Pr(·) 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(·) for all possible segments. Unfortunately, to the best of our knowledge, such corpus never exists. Instead, we turn to Microsoft Web N-Gram corpus [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(·) 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, 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/
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 second6 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 infamously 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

---

6200 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 conditional 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 Freebase7. 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.

3. TWEET SEGMENTATION

In this section, we detail our solution for tweet segmentation. Given an individual tweet $t \in T$, the problem of tweet segmentation is to split $t$ into $m$ consecutive segments, $t = s_1 ... s_m$; 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:

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

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 $s' = w_1 ... w_j$ and $s'' = w_{j+1} ... w_n$, by satisfying:

$$\arg \max_{s', s''} C(s) = C(s') + C(s'').$$

The complexity of Algorithm 1 is $O(lue \log(ne))$, where $l$ is the average tweet length, $n$ 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.

7http://www.freebase.com/
Algorithm 1: Tweet Segmentation

input: A tweet: $t = w_1w_2...w_i$; $u$: the maximum length of a segment $s$; $e$: top $e$ segmentations set $S$ for each segment $s$;

output: An optimal tweet segmentation $t = s_1s_2...s_n$;

for $i = 1 : i < u$ do
    initialize a set $S_i = []$ to store possible segmentation of segment $s_i = w_1w_2...w_i$;
    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 $j < i$ 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 $C(s_i^1)$;
            $\forall S_j \in S_j$ do
                /* $S_j$ contains the top $e$ possible segmentations of $s_i^1$, and $S_j$ is one of them */
                concatenation $S_j$ and $s_i^2$ to form a new segmentation $S_{i,j}$ of $s_i$;
                calculate $C(S_{i,j})$;
            add to $S_i$;
        $C(S) = C(S_j) + C(s_i^2)$
    Sort $S_i$ and keep only the top $e$ segmentations;
return $S \in S_i$ 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 $w_iw_2$ is defined as follows:

$$PMI(w_iw_2) = \log \frac{Pr(w_i|w_2)}{Pr(w_i)} = \log \frac{Pr(w_iw_2)}{Pr(w_i)Pr(w_2)}$$  \hspace{1cm} (3)

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

$$PMI(s) = \log \frac{Pr(w_i...w_n)}{Pr_1 \sum_{i=1}^{n-1} Pr(w_iw_j)Pr(w_{i+1}...w_n)}$$  \hspace{1cm} (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:

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

3.2.2 SCP based Stickiness

Symmetrical Conditional Probability (SCP) was proposed in [1] to measure the “cohesiveness” of bigram $w_1w_2$ by considering both conditional probabilities for the bigram given each single term:

$$SCP(w_1w_2) = Pr(w_1|w_2)Pr(w_2|w_1) = \frac{Pr(w_1w_2)^2}{Pr(w_1)Pr(w_2)}$$  \hspace{1cm} (6)

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

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

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

$$SCP(s) = \log \frac{\prod_{i=1}^{n-1} Pr(w_i...w_n)Pr(w_{i+1}...w_n)}{Pr_1} \in (-\infty, 0)$$  \hspace{1cm} (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:

$$C(s) = \frac{2}{1 + e^{-SCP(s)}}$$  \hspace{1cm} (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 lowercased 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.

Example Tweet Portion

- [youth olympic games sailing competition]

Possible segmentation 1

- [youth] [olympic games] [sailing competition]

Possible segmentation 2

- [youth olympic games] [sailing competition]

Possible segmentation 3

- [youth] [olympic games sailing competition]

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 “Y outh Olympic Games sailing competition” 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
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 4th 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 \( T \) 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 weighted by the Jaccard Index:

\[
\frac{|M(s_a) \cap M(s_b)|}{|M(s_a) \cup M(s_b)|},
\]

where \( M(s) \) is the set of tweets in \( T \) containing segment \( s \).

The segment graph \( G(V, E) \) provides a good local context for each segment in \( T \). 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 transitioning from node \( s_a \) to node \( s_b \) (denoted as \( p_{ab} \)) is given by

\[
p_{ab} = \frac{w_{ab}}{\sum_{c \in V} w_{ac}}.
\]

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 \( e \) 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) \( s \):

\[
e_s = \frac{Q(s)}{\sum_{s' \in V} Q(s')}, \text{ where } Q(s) = e^{Q(s)}.
\]

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 \( e \), the stationary eigenvector \( \pi^T \) of \( P \) is calculated iteratively using power method as below:

\[
\pi = (\gamma P^T + (1 - \gamma) e^T) \pi,
\]

where \( \gamma \) controls the probability of teleportation. The lower \( \gamma \) is, the higher probability the random walk will teleport according to \( e \).
Then, $\pi^T$ 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 multiplies its stationary probability $\pi(s)$ in the segment graph mainly learned from the local context with its teleportation probability learned from Wikipedia:

$$y(s) = e_s \cdot \pi(s). \quad (17)$$

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 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 and SMC, 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., $F1 = \frac{2 \cdot Prec \cdot Recall}{Prec + Recall}$.

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. 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. Moreover, we also compare with a tweet-specific NER system (T-NER) 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 [4]. 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. 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.

A user is considered Singapore-based if she specifies Singapore in the location field of her profile.
Table 2: Different NER systems’ performance on tweets

<table>
<thead>
<tr>
<th>System</th>
<th>Dataset</th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBJ-NER</td>
<td>SGE_g</td>
<td>0.933</td>
<td>0.595</td>
<td>0.727</td>
</tr>
<tr>
<td>Stanford-NER</td>
<td>SGE_g</td>
<td>0.950</td>
<td>0.880</td>
<td>0.913</td>
</tr>
<tr>
<td>T-NER</td>
<td>SGE_g</td>
<td>0.713</td>
<td>0.359</td>
<td>0.478</td>
</tr>
<tr>
<td>TwiNER</td>
<td>SGE_g</td>
<td>0.929</td>
<td>0.660</td>
<td>0.772</td>
</tr>
<tr>
<td>LBJ-NER</td>
<td>SIN_g</td>
<td>0.764</td>
<td>0.265</td>
<td>0.393</td>
</tr>
<tr>
<td>Stanford-NER</td>
<td>SIN_g</td>
<td>0.762</td>
<td>0.293</td>
<td>0.423</td>
</tr>
<tr>
<td>T-NER</td>
<td>SIN_g</td>
<td>0.429</td>
<td>0.509</td>
<td>0.466</td>
</tr>
<tr>
<td>TwiNER</td>
<td>SIN_g</td>
<td>0.419</td>
<td>0.329</td>
<td>0.419</td>
</tr>
</tbody>
</table>

Table 3: Conventional NER systems’ performance on tweets

<table>
<thead>
<tr>
<th>System</th>
<th>Dataset</th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBJ-NER^f</td>
<td>SGE_g</td>
<td>0.674</td>
<td>0.400</td>
<td>0.502</td>
</tr>
<tr>
<td>Stanford-NER^f</td>
<td>SGE_g</td>
<td>0.691</td>
<td>0.642</td>
<td>0.666</td>
</tr>
<tr>
<td>TwiNER</td>
<td>SGE_g</td>
<td>0.929</td>
<td>0.660</td>
<td>0.772</td>
</tr>
<tr>
<td>LBJ-NER^f</td>
<td>SIN_g</td>
<td>0.314</td>
<td>0.314</td>
<td>0.314</td>
</tr>
<tr>
<td>Stanford-NER^f</td>
<td>SIN_g</td>
<td>0.256</td>
<td>0.461</td>
<td>0.329</td>
</tr>
<tr>
<td>TwiNER</td>
<td>SIN_g</td>
<td>0.419</td>
<td>0.329</td>
<td>0.419</td>
</tr>
</tbody>
</table>

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
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 $\text{SIN}_g$ and $\text{SGE}_g$. From Table 5, it can be seen that:

1. While MFS works considerably well on $\text{SGE}_g$, its performance degrades significantly on $\text{SIN}_g$. The tremendous difference in performance is mainly due to the difference between the nature of the two collections $\text{SGE}_g$ and $\text{SIN}_g$. As discussed earlier, $\text{SGE}_g$ 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 $\text{SIN}_g$. This also reflects the degrees of gregarious property of the two collections.

2. RW outperforms MFS significantly on $\text{SGE}_g$ in terms of $F_1$. This validates our assumption that named entities are more likely to co-occur with one another than with non-entities. Since $\text{SIN}_g$ is very diverse, gregarious property is too weak to improve the segment ranking. RWW significantly outperforms MFS and RW in terms of $\text{Prec}$ and $F_1$ in $\text{SIN}_g$ and $\text{SGE}_g$. 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 $F_1$ measure. RWW+Wiki achieves the best performance in both $\text{SIN}_g$ and $\text{SGE}_g$ in terms of $F_1$. We can see that it obtains a large improvement for $\text{Prec}$. We believe that the usage of Wikipedia-based priori in Equation 17 is the main contributing factor for the improvement of $\text{Prec}$ 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 $\text{Prec}$.

5.3.3 Impact of K Value

TwiNER’s performance is related to the choice of $K$. Larger $K$ will increase Recall and decrease Precision, 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 $\text{Prec}@K$ for each $K$ value from 1 to 20016. Here, $\text{Prec}@K$ is the percentage of top $K$ segments returned by TwiNER that are real named entities. Figure 4 shows $\text{Prec}@K$ curves of TwiNER on $\text{SGE}_g$ and $\text{SIN}_g$. Major proportion of segments returned when $K < 50$ are true named entities. TwiNER achieves a stable $\text{Prec}$ performance when $K$ is in the range of $[50, 100]$. After 100, $\text{Prec}@K$ starts to degrade slowly on $\text{SGE}_g$, while $\text{Prec}@K$ is still stable at $K <= 200$ is a good range for the tweet collections we studied here.

---

### Table 4: Impacts of Tweet Segmentation by SCP

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$\text{Prec}@\text{SIN}_g$</th>
<th>$\text{Prec}@\text{SGE}_g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCP</td>
<td>0.721</td>
<td>0.830</td>
</tr>
<tr>
<td>SCP+Norm</td>
<td>0.737</td>
<td>0.861</td>
</tr>
<tr>
<td>SCP+Wiki</td>
<td>0.739</td>
<td>0.841</td>
</tr>
<tr>
<td>SCP+Norm+Wiki</td>
<td><strong>0.758</strong></td>
<td><strong>0.874</strong></td>
</tr>
<tr>
<td>PMI</td>
<td>0.317</td>
<td>0.288</td>
</tr>
<tr>
<td>PMI+Norm</td>
<td>0.297</td>
<td>0.288</td>
</tr>
<tr>
<td>PMI+Wiki</td>
<td><strong>0.344</strong></td>
<td><strong>0.319</strong></td>
</tr>
<tr>
<td>PMI+Norm+Wiki</td>
<td>0.332</td>
<td>0.308</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$\text{Prec}@\text{SIN}_g$</th>
<th>$\text{Recall}@\text{SIN}_g$</th>
<th>$F_1@\text{SIN}_g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFS</td>
<td>0.039</td>
<td><strong>0.753</strong></td>
<td>0.074</td>
</tr>
<tr>
<td>Wiki</td>
<td>0.433</td>
<td>0.398</td>
<td>0.415</td>
</tr>
<tr>
<td>RW</td>
<td>0.039</td>
<td>0.752</td>
<td>0.074</td>
</tr>
<tr>
<td>RWW</td>
<td>0.048</td>
<td>0.336</td>
<td>0.084</td>
</tr>
<tr>
<td>RWW+Wiki</td>
<td><strong>0.576</strong></td>
<td>0.335</td>
<td><strong>0.423</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$\text{Prec}@\text{SGE}_g$</th>
<th>$\text{Recall}@\text{SGE}_g$</th>
<th>$F_1@\text{SGE}_g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFS</td>
<td>0.736</td>
<td>0.565</td>
<td>0.639</td>
</tr>
<tr>
<td>Wiki</td>
<td>0.738</td>
<td><strong>0.688</strong></td>
<td>0.712</td>
</tr>
<tr>
<td>RW</td>
<td>0.856</td>
<td>0.536</td>
<td>0.659</td>
</tr>
<tr>
<td>RWW</td>
<td><strong>0.985</strong></td>
<td>0.536</td>
<td>0.694</td>
</tr>
<tr>
<td>RWW+Wiki</td>
<td>0.929</td>
<td>0.646</td>
<td><strong>0.762</strong></td>
</tr>
</tbody>
</table>
local context of tweets, TwiNER shows a promising performance. Nevertheless, it is observed that the choice of $K$ value in various scenarios is planned for future work. Nevertheless, it is practically infeasible and unnecessary to listen 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 linguistic 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 planned for future work.

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 linguistic 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 planned for future work.

7. REFERENCES