An End-to-End Entity Linking Approach for Tweets

Ikuya Yamada1 2 3
ikuya@ousia.jp

Hideaki Takeda1
takeda@nii.ac.jp

Yoshiyasu Takefuji2
takefuji@sfc.keio.ac.jp

1 Studio Ousia Inc., 4489-105-221 Endo, Fujisawa, Kanagawa, Japan
2 Keio University, 5322 Endo, Fujisawa, Kanagawa, Japan
3 National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo, Japan

ABSTRACT
We present a novel approach for detecting, classifying, and linking entities from Twitter posts (tweets). The task is challenging because of the noisy, short, and informal nature of tweets. Consequently, the proposed approach introduces several methods that robustly facilitate successful realization of the task with enhanced performance in several measures.

Keywords
Entity linking; Wikification; Twitter; DBpedia; Wikipedia

1. INTRODUCTION
Microblogging services, such as Twitter, are rapidly becoming virtually ubiquitous. This is attributable to the fact that they are extremely valuable mechanisms that enable us to obtain live and raw information in real time. In this paper, we describe our approach to the #Microposts 2015 NEEL challenge [6], a competition for extracting and typifying entity mentions appearing in tweets, and linking those mentions to the corresponding URIs of the DBpedia 2014 dataset1, with non-existent mentions also being recognized as NIL mentions.

The main difficulty inherent in this task stems from the noisy, short, and informal nature of tweets. The performance of previous approaches suffered because they tended to focus on well-written, long texts such as news articles. Our system explicitly focuses on tweets and addresses the problem using a variety of methods working together.

Our proposed system addresses the task in an end-to-end manner. Unlike most of the previous approaches, the system does not use an external named entity recognition system (NER) to generate candidates of the entity mentions because the current NER typically performs badly for tweets [5]. Our system first generates the candidates by using approximate candidate generation that can detect misspelled and abbreviated mentions and acronyms. Then it uses supervised machine-learning to remove irrelevant candidates and resolve them into the corresponding DBpedia URIs.

1http://wiki.dbpedia.org/Downloads2014

Consequently, we constructed three supervised machine-learning models to detect NIL entity mentions and predict the types (e.g., PERSON and LOCATION) of the detected mentions.

2. THE PROPOSED SYSTEM
Our proposed system addresses the task using a procedure comprising the following five steps: 1) preprocessing, 2) mention candidate generation, 3) mention detection and disambiguation, 4) NIL mention detection, and 5) type prediction.

2.1 Preprocessing
We tokenize a tweet and assign part-of-speech tags to the resulting tokens using ARK Twitter Part-of-Speech Tagger [2] with our enhanced hashtag tokenization method. We also extract the timestamp of the tweet from the Tweet ID.

2.2 Mention Candidate Generation
In this step, the candidates of the entity mentions are generated from the tweet using the methods described below.

Mention-Entity Dictionary.
The system uses a mention-entity dictionary that maps mention surface (e.g., apple) to the possible referent entities (e.g., Apple Inc., Apple (food)). The possible mention surfaces of an entity are extracted from the corresponding Wikipedia page title, the page titles of the Wikipedia pages that redirect to the page of the entity, and anchor texts in Wikipedia articles that point to the page of the entity. We constructed this dictionary using the January 2015 dump of Wikipedia.

Candidate Generation Methods.
The system generates candidates using the mention-entity dictionary: it first takes all the n-grams (n < 10) from the tweet and performs queries to the dictionary using the text surface of each of these n-grams. The following four methods are used to retrieve candidates:

• Exact search retrieves mention candidates that have text surfaces exactly equal to the query text.
• Fuzzy match searches the mention candidates that have text surfaces within a certain distance of the query text measured by edit distance.
• Approximate token search obtains mention candidates whose text surfaces have a significant ratio of words in common with the query text.
• Acronym search retrieves mention candidates with possible acronyms2 that include the query text.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXXX-XX/XX/XX ...$15.00.
The system first generates possible mention candidates using the above methods, sorts these candidates according to the number of occurrences in which the mention appear as a link to the referent entity, and selects the top \( k \) candidates \((k = 100 \text{ for exact search and } k = 30 \text{ for other methods})\). Additionally, we experimentally set the maximum allowed edit distance of fuzzy match to two and the minimum ratio of approximate token search to 66% because these settings achieve the best scores in our experiments.

### 2.3 Mention Detection and Disambiguation

In this step, we first assign a score to mention candidates using a supervised machine-learning model. In this case, we used random forest as the machine-learning algorithm.

**Features.**

We started out using features similar to those proposed in previous works \([1, 3]\), and subsequently introduced several novel features to enhance performance. The features introduced include 1) contextual information using word embeddings to measure the contextual similarity between a tweet and an entity, 2) temporal popularity knowledge of an entity extracted from Wikipedia page view data, and 3) string similarity measures to measure the similarity between the title of the entity and the mention (e.g., edit distance).

**Overlap Resolution.**

Finally, the overlapped entity mentions are resolved. We start with the beginning of the tweet and iterate over the candidate entity mentions. Then, we detect the mention if the corresponding span of the mention has not already been detected and the score assigned to the mention is above the threshold. If multiple mentions are found, the mention with the highest score is selected.

### 2.4 NIL Mention Detection

We formulate the task of detecting NIL mentions from a tweet as a supervised classification task to assign a binary label to each of all possible n-grams \((n < 10)\). Random forest is again used as our machine-learning algorithm.

**Features.**

We extract several features from the output of the Stanford NER\(^3\) using two types of models: 1) a standard three-class model, and 2) a model that does not use capitalization as a feature. We also use the ratio of capitalized words as an indicator of the reliability of the capitalization in the tweet. Additionally, various other features are used, such as part-of-speech tags of the surrounding words and the length of the n-grams.

### 2.5 Type Prediction

We cast the task of detecting types of mentions as a multi-class supervised classification task. In the previous steps, we extracted two types of mentions: entity mentions and NIL mentions. Thus, we are able to build two separate classifiers to predict the entity types for each type of mention. We developed two machine-learning models using logistic regression and random forest and created the final model by building an ensemble model on top of these models in order to boost the performance.

**Features for Entity Mentions.**

The primary features used to detect types of entity mentions are the corresponding entity classes retrieved from DBpedia and Freebase (e.g., FictionalCharacter, SportsTeam).

<table>
<thead>
<tr>
<th>Name</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>strong_link_match</td>
<td>0.786</td>
<td>0.656</td>
<td>0.715</td>
</tr>
<tr>
<td>strong_typed_mention_match</td>
<td>0.656</td>
<td>0.630</td>
<td>0.642</td>
</tr>
<tr>
<td>mention_ceaf</td>
<td>0.857</td>
<td>0.823</td>
<td>0.840</td>
</tr>
</tbody>
</table>

Table 1: Summary of experimental results

We also use our 300 dimensional entity-embeddings constructed from Wikipedia and the predicted entity types of the Stanford NER.

**Features for NIL Mentions.**

In order to detect the types of NIL mentions, we use features extracted from word embeddings. Here, the GloVe Twitter 2B model \([4]\) is used as the word embeddings. We also use the predicted types of the Stanford NER and the part-of-speech tags.

### 3. EXPERIMENTAL RESULTS

In our experiments, we used the #Microposts 2015 dataset \([6]\) split into a training set and a test set. These sets contained 3,498 and 500 tweets respectively.

Table 1 shows a summary of our experimental results. We evaluated our system using the following three measures: strong_link_match to evaluate the performance of linking entities, strong_typed_mention_match to measure the performance of mention detection and entity typing, and mention_ceaf for calculating the performance of clustering detected mentions into entity mentions or NIL mentions.\(^4\) We successfully achieved accurate performance in all of the measures.

### 4. CONCLUSIONS

In this paper, we described our approach for detecting, classifying, and linking entity mentions in tweets. We introduced a novel machine-learning approach specifically targeted at tweets and successfully achieved enhanced performance on the #Microposts2015 dataset.

### References


\(^3\)http://nlp.stanford.edu/software/CRF-NER.shtml

\(^4\)For further details of these measures, please refer to https://github.com/wikilinks/neleval/wiki/Evaluation