NEED4Tweet: A Twitterbot for Tweets Named Entity Extraction and Disambiguation

Mena B. Habib
Database Chair
University of Twente
Enschede, The Netherlands
m.b.habib@ewi.utwente.nl

Maurice van Keulen
Database Chair
University of Twente
Enschede, The Netherlands
m.vankeulen@utwente.nl

Abstract

In this demo paper, we present NEED4Tweet, a Twitterbot for named entity extraction (NEE) and disambiguation (NED) for Tweets. The straightforward application of state-of-the-art extraction and disambiguation approaches on informal text widely used in Tweets, typically results in significantly degraded performance due to the lack of formal structure; the lack of sufficient context required; and the seldom entities involved. In this paper, we introduce a novel framework that copes with the introduced challenges. We rely on contextual and semantic features more than syntactic features which are less informative. We believe that disambiguation can help to improve the extraction process. This mimics the way humans understand language.

1 Introduction

Twitter is an important source for continuously and instantly updated information. It contains a large amount of unstructured information about users, locations, events, etc. Shortness and informality of Tweets are challenges for Natural Language Processing (NLP) tasks. Information Extraction (IE) is the NLP field of research that is concerned with obtaining structured information from unstructured text. IE systems attempt to interpret human language text in order to extract information about different types of events, entities, or relationships. Named entity extraction (NEE) is a subtask of IE that aims to locate phrases (mentions) in the text that represent names of persons, organizations, or locations regardless of their type. Named entity disambiguation (NED) is the task of determining which concrete person, place, event, etc. is referred to by a mention. Wikipedia articles are widely used as an entity's reference.

Challenges: NEE and NED in informal text are challenging. Here we summarize the challenges of NEE and NED for Tweets:

- The informal language widely used in Tweets makes the extraction process more difficult. Proper capitalization is a key feature that the state-of-the-art NEE approaches have relied on. However, this feature gets less attention from Twitter users when they write their Tweets.
- The limited length (140 characters) of Tweets forces the senders to provide dense information by using acronyms and informal language. This makes both the extraction and the disambiguation processes more complex.
- The limited coverage of a Knowledge Base (KB) is another challenge facing NED for tweets. According to (Lin et al., 2012), 5 million out of 15 million mentions on the Web cannot be linked to Wikipedia. This means that relying only on a KB for NED leads to around 33% loss in the disambiguated entities. This percentage is higher on Twitter because of its social nature where users also discuss information about seldom entities.
- The processes of NEE and NED involve degrees of uncertainty. For example, in the tweet "*history should show that bush jr should be in jail or at least never should have been president*", for some NEE systems, it may be uncertain whether the word '*jr*' should be part of the mention bush or not. This motivates us to fundamentally consider sets of possible alternatives in an early stage of the extraction and the disambiguation processes and do a later filtration instead of making hard decisions from the beginning.
- Named entity (NE) representation in KBs poses another NED challenge. The YAGO

KB (Suchanek et al., 2007) uses the Wikipedia anchor text as a possible mention representation for named entities. However, there may be more representations that do not appear in the Wikipedia anchor text, but are meant to refer to the entity because of a spelling mistake or because of a new abbreviation for the entity.

In this demo, we introduce NEED4Tweet, a Twitterbot for a combined system for NEE and NED in Tweets that uses their interdependency and mimics how humans exploit it in language understanding. The system is based on our work (Habib and van Keulen, 2015). We use a generic open world approach for NED in Tweets for any named entity even though it has no Wikipedia article. Mentions are disambiguated by assigning them to either a Wikipedia article or a home page. We handle the uncertainty involved in the extraction process by considering possible alternatives in an early stage then evaluate these alternatives later based on disambiguation outcomes. The proposed approach is shown to be robust against the coverage of KBs and the informality of the used language.

2 Related work

2.1 Named Entity Disambiguation

NED in Web documents is a topic that is well covered in literature. Recently, researchers have attempted NED for informal short text such as Tweets. Most of this research investigate the problem of entity-oriented disambiguation. Within this theme, (Spina et al., 2011), (Christoforaki et al., 2011), (Yerva et al., 2012) and (Delgado et al., 2012) focus on the task of filtering Tweets containing a given a mention of topic-centric entity, depending whether the Tweet is actually related to the entity or not. They develop a set of features (co-occurrence, Web-based features, collection-based features) to find keywords for positive and negative cases.

Similar to our problem discussed in Section 3.2, is the problem of entity home page finding, which was part of the TREC Web and entity tracks. One of the proposed approaches for this task was (Westerveld et al., 2002). The authors combine content information with other sources as diverse as inlinks, URLs and anchors to find an entry page. Although the TREC problem looks similar to ours, the Tweets' short informal nature makes it more tricky to find an entity reference page.

2.2 Named Entity Extraction

Many tools and services have been developed for the NEE task in web documents written in formal language. In spite of this, few research efforts studied NEE in Tweets. In (Ritter et al.,), the authors built an NLP pipeline to perform NEE. The pipeline involves part-of-speech tagging, shallow parsing, and a novel SVM classifier that predicts the informativeness of capitalization in a Tweet. It trains a Conditional Random Fields (CRF) model with all the aforementioned features for NEE. For classification, LabeledLDA is applied where entity types are used as classes. A bag-of-words-based profile is generated for each entity type, and the same is done with each extracted mention. Classification is done based on the comparison of the two.

The contextual relationship between the microposts is considered by (Jung, 2012). The paper proposes merging the microtexts by discovering contextual relationship between the microtexts. A group of microtexts contextually linked with each other is regarded as a microtext cluster. Once this microtext cluster is obtained, they expect that the performance of NEE can be better. The authors provide some suggestions for Contextual closure, Microtext cluster, Semantic closure, Temporal closure, and Social closure. Those closures are used by Maximum Entropy for the NER task.

Similarly, (Li et al., 2012) exploits the gregarious property in the local context derived from the Twitter stream in an unsupervised manner. The system first leverages 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 NE. Afterwards, a ranking approach tries to rank segments according to their probability of being an NE. The highly-ranked segments have a higher chance of being true NEs. Each segment is represented as a node in a graph, and using the Wikipedia and the context of Tweet (adjacent nodes (segments)), a score is assigned to that segment if it is an NE or not.

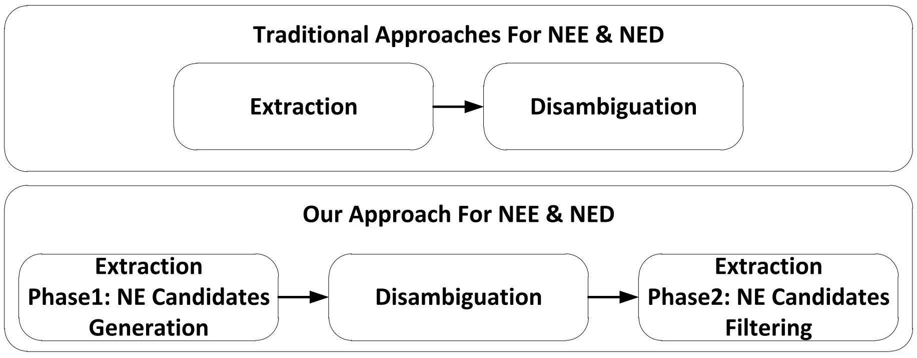

Traditional Approaches For NEE & NED
Extraction
Disambiguation
Our Approach For NEE & NED
Extraction Phase1: NE Candidates Generation
Disambiguation
Extraction Phase2: NE Candidates Filtering

Figure 1: Traditional approaches versus our approach for NEE and NED.

3 NEED4Tweet

Although the logical order for a traditional IE system is to complete the extraction process before commencing with the disambiguation process, we start with an initial extraction-like phase aiming for high recall (i.e. aiming to find as many reasonable mention candidates as possible). We then attempt disambiguation for all the extracted mentions. Finally we classify extracted mention candidates into true and false NE using features (clues) derived from the results of the disambiguation phase such as KB information and entity coherency. Figure 1 illustrates our general approach contrasted with the traditional process.

The potential of this order is that the disambiguation step gives extra clues (such as Entity-Tweet context similarity) about each NE candidate. This information can help in the decision whether the candidate is a true NE or not.

3.1 Mention Candidates Generation

This phase is aiming to find as many reasonable mention candidates as possible. For this task, we unionize the output of the following mention candidates generation methods:

- **Tweet Segmentation**: Tweet text is segmented using the segmentation algorithm described in (Li et al., 2012). Each segment is considered a mention candidate.
- **KB Lookup**: We scan all possible n-grams of the Tweet against the mentions-entities table of YAGO KB. N-grams that matches a YAGO mention are considered mention candidates.

3.2 Disambiguation

For NED, we use a generic open world NED approach where mentions are disambiguated by assigning them to either a Wikipedia article (Wikipedia entity) or a home page (non-Wikipedia entity) (Habib and van Keulen, 2013). The NED approach is composed of three modules; matcher, feature extractor, and SVM ranker.

- **Matcher**: This module is responsible for finding the possible candidate entities of a given mention. For this task, we use the mention-entity table of YAGO KB to get the possible entities for the given mention. Furthermore, we use the mention as an input query for the Google API. The top 18 Web pages retrieved by Google are also considered candidate entities for that mention.
- **Feature Extractor**: For each entity page candidate, we extract a set of context and URL features. Context features (such as language model and overlapping terms between tweet and document) measure the context similarity between mention context (the tweet text) and entity candidates' home pages. URL features (such as path length and mention-URL string similarity) measure the likelihood of the candidate URL being a representative of the entity home page. These features give indicators on how likely the candidate entity page could be a representative to the mention.
- **SVM Ranker**: After extracting the aforementioned set of features, SVM classifier is used to rank candidate entity pages of a mention. We consider the top ranked page to be

the entity of the input mention. In this demo, we use an SVM which is trained on the two NED datasets presented in (Habib and van Keulen, 2013).

3.3 Mention Candidates Filtering

After generating the mentions candidate list, we apply our disambiguate approach to disambiguate each mention candidate. After that, we use another SVM classifier to predict which mention candidates are true positives and which ones are not. For each mention candidate, we extract the following set of features :

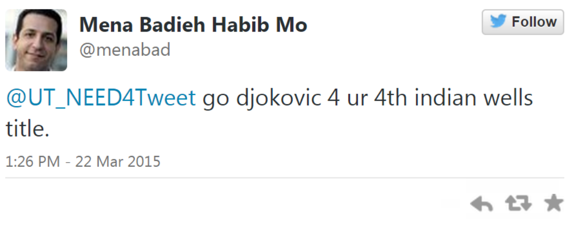
- **Shape Features**: If the mention candidate is initially or fully capitalized and if it contains digits.
- **Probabilistic Features**:
  - The joint and conditional probability of the mention candidate obtained from the Microsoft Web N-Gram service.
  - The stickiness of the segment as described in (Li et al., 2012).
  - The segment frequency over around 5 million tweets[1].
- **KB Features**:
  - Whether the segment appears in WordNet.
  - Whether the segment appears in the YAGO mention-entity look-up table.
- **Disambiguation Features**: All the features described in Section 3.2 derived from the entity page linked to the given mention candidate.

In this demo, we use an SVM which is trained on four different NEE datasets presented in (Ritter et al.,), (Basave et al., 2013), (Locke and Martin, 2009), and (Habib and van Keulen, 2012).

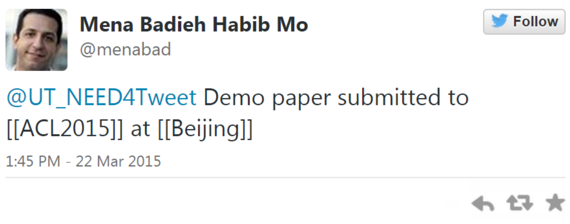
3.4 Final NE Set Generation

Beside the SVM, we also use a trained CRF model for NEE. We use the CRF model described in (Zhu et al., 2014) trained on the four collections mentioned in Section 3.3. To train the CRF, Tweet text is tokenized using a special tweet tokenizer (Gimpel et al., 2011) and the following features are extracted and used for training:

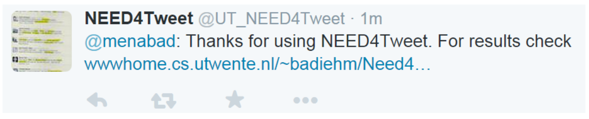
[1]`http://wis.ewi.tudelft.nl/umap2011/` + TREC 2011 Microblog track collection.


Mena Badieh Habib Mo
@menabad
@UT_NEED4Tweet go djokovic 4 ur 4th indian wells title.
1:26 PM - 22 Mar 2015

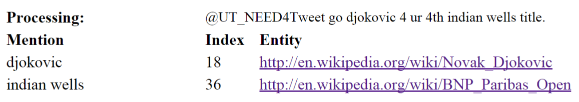
(a) Example 1: Tweet for testing both NEE and NED.


Mena Badieh Habib Mo
@menabad
@UT_NEED4Tweet Demo paper submitted to [[ACL2015]] at [[Beijing]]
1:45 PM - 22 Mar 2015

(b) Example 2: Tweet for testing NED only.

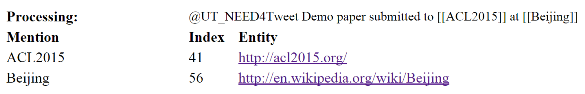

NEED4Tweet @UT_NEED4Tweet · 1m
@menabad: Thanks for using NEED4Tweet. For results check wwwhome.cs.utwente.nl/~badiehm/Need4…

(c) Tweet reply.


Processing: @UT_NEED4Tweet go djokovic 4 ur 4th indian wells title.

Mention	Index	Entity
djokovic	18	http://en.wikipedia.org/wiki/Novak_Djokovic
indian wells	36	http://en.wikipedia.org/wiki/BNP_Paribas_Open

(d) Results of example 1


Processing: @UT_NEED4Tweet Demo paper submitted to [[ACL2015]] at [[Beijing]]

Mention	Index	Entity
ACL2015	41	http://acl2015.org/
Beijing	56	http://en.wikipedia.org/wiki/Beijing

(e) Results of example 2

Figure 2: NEED4Tweet Twitterbot

- The Part of Speech (POS) tag of the token provided by a special POS tagger designed for tweets (Gimpel et al., 2011).
- Whether the token's initial is capitalized.
- Whether the token's characters are all capitalized.
- Whether the token has any capital letters.

We consider the best annotation set for the tweet given by the CRF model as true positives. To generate the final NE set, we take the union of the CRF annotation set (after being disambiguated) and the SVM results, after removing duplicate and overlapped extractions. To resolve the overlapped mentions, we select the mention that appears in Yago KB. If both mentions appear in Yago or both don't, we select the one with the longer length.

The idea behind this combination is that the SVM and the CRF work in a different way. The former is a distance based classifier that uses numeric features for classification which CRF can not handle, while the latter is a probabilistic model that can naturally consider state-to-state dependencies and feature-to-state dependencies. On the other hand, SVM does not consider such dependencies. The hybrid approach of both makes use of the strength of each. While the CRF makes better use of the traditional features like POS and Capitalization, the SVM makes better use of the disambiguation (coherency) features.

4 Twitterbot

A Twitterbot is a program used to produce automated posts on the Twitter microblogging service. We developed our system as a Twitterbot which receives the Tweet, processes it and sends a reply message contains a link to a page that shows the generated annotations. We use Twitter API[2] for both receiving the Tweets and sending the replies. To use NEED4Tweet Twitterbot, one should send a Tweet contains either the mention '@UT_NEED4Tweet' or the hashtag '#NEED4Tweet' as shown in Figures 2(a) and 2(b) respectively. Withing few seconds after sending the tweet, the sender will get a reply Tweet (see Figure 2(c)) that includes link to a simple HTML page contains the generated annotations (see Figures 2(d) and 2(e)). The page contains a list of the extracted mentions, their start offset in the Tweet, and their linked entities. It is also possible to test only the disambiguation component by manually coating the mentions required to be disambiguated using double square brackets ([[]])as shown in Figure 2(b).

5 Evaluation

5.1 Data sets

To validate our approach, we use three collections of tweets. The first two data sets are mainly designed for a NER task. We manually construct the NED ground truth by linking each NE to only one appropriate entity page. We give higher priority to Wikipedia pages. When no Wikipedia page exists for a mention, we link it to a non-Wikipedia home page or profile page.

The first data set (Locke collection) is the one used in (Locke and Martin, 2009). The second data set (Habib collection) is the one used in (Habib and van Keulen, 2012) which is relatively small in the number of tweets but rich in the number of NEs. It is composed mainly from tweeted news about sportsmen, celebrities, politics, etc.

The third data set (#Microposts collection) is provided by the #Microposts Named Entity Extraction & Linking (NEEL) Challenge (Cano Basave et al., 2014). The NEEL Challenge task required participants to build systems to extract entity mentions from a tweet and to link the extracted mentions to DBpedia. Note that this data set does not contain any non-Wikipedia entities. We have done the mapping from the YAGO KB to DBpedia by identifying the Wikipedia page as a common property for the identical entities.

(a) Locke collection

	Pre.	Rec.	F1
DBpedia Spotlight	0.1004	0.2669	0.1459
Stanford + AIDA	0.5005	0.2940	0.3704
NEED4Tweet	**0.5455**	**0.5640**	**0.5546**

(b) Habib collection

	Pre.	Rec.	F1
DBpedia Spotlight	0.3711	0.5333	0.4377
Stanford + AIDA	**0.7263**	0.5569	0.6304
NEED4Tweet	0.6861	**0.7157**	**0.7006**

(c) #Microposts collection

	Pre.	Rec.	F1
DBpedia Spotlight	0.1873	0.3349	0.2403
Stanford + AIDA	0.5092	0.2795	0.3609
NEED4Tweet	**0.5337**	**0.5343**	**0.5339**

Table 1: Combined evaluation of NEE and NED.

5.2 Experimental Results

In this experiment, we compare the performance of NEED4Tweet against two competitors: AIDA[3] and DBpedia Spotlight.[4] AIDA is a disambiguation system although it uses Stanford_NER for automatic NE extraction. We consider the combination of Stanford_NER and the AIDA disambiguation system as one competitor to our extraction and disambiguation system. DBpedia Spotlight (Mendes et al., 2011) is a tool for automatically annotating mentions of DBpedia resources in text. We used DBpedia Spotlight through its Annotate Web Service endpoint. We used the

[2]https://dev.twitter.com/

[3]https://d5gate.ag5.mpi-sb.mpg.de/webaida/

[4]https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki

NESpotter implementation for the extraction configuration. The results in Table 1 show the superiority of NEED4Tweet over DBpedia Spotlight and the combined Stanford and AIDA system. More experimental results and analysis can be found in (Habib and van Keulen, 2015).

6 Conclusion

In this demo paper, we present NEED4Tweet, a Twitterbot for NEE and NED in tweets. The system is composed of three phases. The first phase aims to generate NE candidates with an emphasis on achieving high recall. The second phase aims to disambiguate all the candidates generated in the first phase. For this task, we use a generic non-entity oriented disambiguation approach. Mentions are disambiguated by assigning them to either a Wikipedia article or a home page. Finally, the third phase is to filter the NE candidates using features derived from disambiguation and other shape and KB features. The proposed approach is shown to be robust against the coverage of KBs and the informality of the used language.

References

Amparo Elizabeth Cano Basave, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. 2013. Making sense of microposts (#msm2013) concept extraction challenge. In *Making Sense of Microposts (#MSM2013) Concept Extraction Challenge*, pages 1–15.

Amparo Elizabeth Cano Basave, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. 2014. Making sense of microposts (#microposts2014) named entity extraction & linking challenge. In *Proc. of (#Microposts2014) Workshop*, pages 54–60.

Maria Christoforaki, Ivie Erunse, and Cong Yu. 2011. Searching social updates for topic-centric entities. In *Proc. of exploreWeb 2011 Workshop*, pages 34–39.

A. D. Delgado, R. Mart'ınez, A. Pérez Garc'ıa-Plaza, and V. Fresno. 2012. Unsupervised Real-Time company name disambiguation in twitter. In *Proc. of RAMSS 2012 Workshop*, pages 25–28.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: annotation, features, and experiments. In *Proc. of ACL 2011*, HLT '11, pages 42–47.

Mena B. Habib and Maurice van Keulen. 2012. Unsupervised improvement of named entity extraction in short informal context using disambiguation clues. In *Proc. of SWAIE 2012 Workshop*, pages 1–10.

Mena B. Habib and M. van Keulen. 2013. A generic open world named entity disambiguation approach for tweets. In *Proc. of KDIR 2013*, pages 267–276.

Mena B. Habib and Maurice van Keulen. 2015. Twitterneed: A hybrid approach for named entity extraction and disambiguation for tweets. *To appear in the journal of Natural Language Engineering*.

Jason J. Jung. 2012. Online named entity recognition method for microtexts in social networking services: A case study of twitter. *Expert Syst. Appl.*, 39(9):8066–8070.

Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. 2012. Twiner: named entity recognition in targeted twitter stream. In *Proc. of SIGIR 2012*, pages 721–730.

Thomas Lin, Mausam, and Oren Etzioni. 2012. Entity linking at web scale. In *Proc. of AKBC-WEKEX 2012 Workshop*, pages 84–88.

Brian Locke and James Martin. 2009. Named entity recognition: Adapting to microblogging. Senior Thesis, University of Colorado.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: Shedding light on the web of documents. In *Proc. of I-Semantics 2011*, pages 1–8.

A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *Proc. of EMNLP 2011*, pages 1524–1534.

Damiano Spina, Enrique Amigó, and Julio Gonzalo. 2011. Filter keywords and majority class strategies for company name disambiguation in twitter. In *Proc. of CLEF 2011*, pages 50–61.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proc. of WWW 2007*, pages 697–706.

Thijs Westerveld, Wessel Kraaij, and Djoerd Hiemstra. 2002. Retrieving web pages using content, links, urls and anchors. In *Tenth Text REtrieval Conference, TREC 2001*, volume SP 500, pages 663–672.

Surender Reddy Yerva, Zoltán Miklós, and Karl Aberer. 2012. Entity-based classification of twitter messages. *IJCSA*, 9(1):88–115.

Zhemin Zhu, Djoerd Hiemstra, and Peter Apers. 2014. Linear co-occurrence rate networks (l-crns) for sequence labeling. In *Statistical language and speech processing*, volume 8791 of *Lecture notes in computer science*, pages 185–196.