

E2E: An End-to-End Entity Linking System for Short and Noisy Text

Ming-Wei Chang Bo-June Hsu Hao Ma Ricky Loynd Kuansan Wang
Microsoft Research
Redmond, WA
{minchang|paulhsu|haoma|riloyn|kuansanw}@microsoft.com

ABSTRACT

We present E2E, an end-to-end entity linking system that is designed for short and noisy text found in microblogs and text messages. Mining and extracting entities from short text is an essential step for many content analysis applications. By jointly optimizing entity recognition and disambiguation as a single task, our system can process short and noisy text robustly.

Keywords

Information Extraction, Social Media, Entity Linking

1. INTRODUCTION

In this paper, we describe our entity linking system called E2E for the #Microposts2014 NEEL Challenge [1]. Our system focuses on the task of extracting and linking entities from short and noisy text given entity databases like Wikipedia or Freebase. An entity linking system usually needs to perform two key functions: mention recognition and entity disambiguation. In mention recognition the system identifies each mention (surface form) of an entity in the text. In entity disambiguation, the system maps mentions to canonical entities. E2E has been carefully designed to treat entity recognition and disambiguation as a single task.

2. THE ARCHITECTURE OF E2E

When a short message is received, E2E processes the message in four stages: Text Normalization, Candidate Generation and Joint Recognition-and-Disambiguation, and Overlap Resolution.

Text Normalization.

In this stage, a short message is normalized and tokenized. For tweets, the retweet symbols and some other special symbols are removed. Punctuation symbols are represented as separate tokens in general.

Copyright © 2014 held by author(s)/owner(s); copying permitted only for private and academic purposes.
Published as part of the #Microposts2014 Workshop proceedings, available online as CEUR Vol-1141 (<http://ceur-ws.org/Vol-1141>)

#Microposts2014, April 7th, 2014, Seoul, Korea.

Candidate Generation.

The next step is to generate a list of surface form candidates that could potentially link to entities. E2E uses a lexicon to generate the candidate surface forms. A lexicon is a dictionary that maps a surface form to its possible entity set. For example, the word “giants” could refer to “New York Giants”, or “San Francisco Giants”, etc. Our lexicon is mainly composed by extracting information from Wikipedia and Freebase. The dictionary is constructed to support fuzzy mention matching based on edit distance. Note that we over-generate candidates at this stage, and no filtering is performed.

Joint Recognition and Disambiguation.

This stage is the key component of the E2E framework. Given a message, the goal here is to figure out the entity assignment of each candidate mention generated from previous stages. Note that a candidate mention may be rejected altogether (mapped to the null entity).

Our model is based on a supervised learning method. Given a message m and a candidate mention a , the entity assignment is generated from the ranking of all possible entities in the entity set $\mathcal{E}(a)$.

$$\arg \max_{e \in \{\mathcal{E}(a) \cap \emptyset\}} f(\Phi(m, a, e)), \quad (1)$$

where f is the function of the model, and Φ is a feature function over the input m , the mention a and the candidate output e . Note that it is very likely E2E rejects a candidate and does not link it to an entity (link a to \emptyset). The joint approach that recognizes and disambiguates entity mentions together is crucial for E2E to properly link surface forms to the corresponding entities.

Overlap Resolution.

At this point, many of the linked mentions will overlap each other. Dynamic programming resolves these conflicts by choosing the best-scoring set of non-overlapping mention-entity mappings. The experimental results show that resolving overlap improve the models performance consistently in different settings.

3. SYSTEM IMPLEMENTATION

Our database is constructed from both Wikipedia and Freebase. The whole system is implemented in C#.

Entity linking systems often require a large amount of memory due to the size of the structured/unstructured data for many entities. High memory consumption restricts the scale of an entity linking system, limiting the number of allowed entities that can be handled. Long loading times also

reduce the efficiency of conducting experiments. In E2E, we adopt the completion trie data structure proposed in [4] instead of a hash map dictionary. The completion trie greatly reduces the memory footprint and loading time of E2E.

We have tested two learning methods when developing E2E: a structured support vector machine algorithm [2] and a fast implementation of the MART gradient boosting algorithm [3]. The structural SVM model is a linear model that takes into account all of the candidates together in the same tweet. MART learns an ensemble of decision/regression trees with scalar values at the leaves, but treats each candidate separately. The submitted results are generated using MART due to its superior performance on our development set.

Features.

Three groups of features were used in our system. The textual features are the features regarding the textual properties of the surface form and its context. For example, one feature indicates if the current surface form and the surrounding words are capitalized or not. We also use features generated from the output of the in-house named entity recognition system that is specially designed to be robust on non-capitalized words. The entity graph features capture the semantic cohesiveness between the entity-entity and entity-mention pairs. This group of features was mainly calculated using the entity database and its structured data. Finally, the statistical features indicates the word usage and entity popularity using the information collected from the web.

Among the three group features, the statistical feature group is the most important one. We describe some of the most important features in the following. Let a denote the surface form of a candidate, and e denote the an entity. One important feature is the link probability feature $P_l(a)$, which indicates the probability that a phrase is used as an anchor in Wikipedia. For each phrase a , we also collect statistics about the probability that a phrase is capitalized in Wikipedia. We refer to this feature as the capitalization rate feature, $P_c(a)$.

We also compute features that captures the relationships between an anchor a and an entity e . The probability $P(e|a)$ captures the likelihood of an anchor linked to an Wikipedia entity. We have downloaded Wikipedia page view counts, representing page view information from 2012.¹ According to the popularity information, we add another probability feature that captures the relative popularity of the pages that could be linked from the anchor a . More precisely, $P_v(e|a) = v(e_i) / (\sum_{\{e \in \mathcal{E}(a) \cap \emptyset\}} v(e))$, where $v(e)$ represents the view count for the page e .

4. RESULTS

In our experiments, we split the training set into two sets that contains 1534 and 800 tweets, respectively. The 800-tweet data is used as our development set. Our analysis shows that robust mention detection is often the source of errors in the current the entity linking systems. In order to achieve better F1 score, we change the prediction function to

$$\arg \max_{e \in \{\mathcal{E}(a) \cap \emptyset\}} f(\Phi(m, a, e)) - s[e = \emptyset], \quad (2)$$

¹<http://dammit.lt/wikistats>

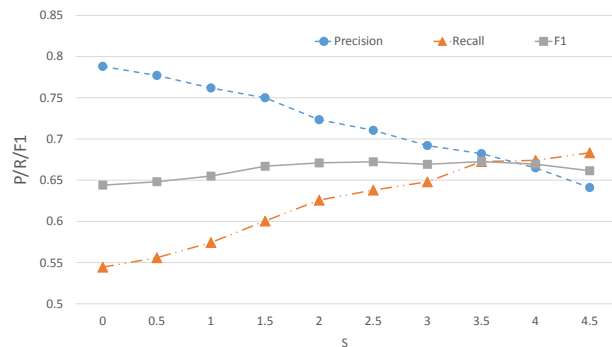


Figure 1: Results of E2E on the development set.

where $[\cdot]$ is an indicator function. When s increases, the system will produce more entities. From the results in Figure 1, we found that tuning s does impact results significantly. After learning parameters and desired value of s are chosen, we then retrain the E2E using the full training data, and generate final results with $s = 0, 2.5$ and 3.5 , respectively.

Error Analysis.

We analyze at our results on the development set with $s = 3.5$. In the development set, there are 1304 mentions, and E2E generates total number of 18746 candidates in the candidate generation stage. Our error analysis shows that E2E misses 340 entity mentions and predict extra 284 mentions. Among the errors, E2E has troubles on the “number” entities (e.g. `1_(number)`). Further investigation reveals that the tokenization choice of E2E plays a big part of these errors, given that most punctuations are being treated as separate tokens. Interestingly, E2E only makes 44 cases where it correctly recognizes the mentions but link to wrong entities. Most errors occur when E2E fail to recognize mentions correctly.

5. CONCLUSIONS

In this paper, we presented E2E, a system that performs joint entity recognition and disambiguation on short and noisy text. We found that the substance of a successful entity linking system consists of successfully combining all of the components.

Due to the time limitation, the submitted system still has plenty of room to improve. For example, one important direction is to explore the relationships between different tweets to improve entity linking results. Developing a robust mention detection algorithm is an important research direction as well.

6. REFERENCES

- [1] A. E. Cano Basave, G. Rizzo, A. Varga, M. Rowe, M. Stankovic, and A.-S. Dadzie. Making Sense of Microposts (#Microposts2014) Named Entity Extraction & Linking Challenge. In *Proc., #Microposts2014*, pages 54–60, 2014.
- [2] M.-W. Chang and W.-T. Yih. Dual coordinate descent algorithms for efficient large margin structured prediction. *TACL*, 2013.
- [3] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1999.
- [4] B.-J. P. Hsu and G. Ottaviano. Space-efficient data structures for top-k completion. In *WWW*, 2013.