# Entity Linking over Short Texts

Ricardo Ascenso Lavareda
rickylavareda@gmail.com
73985

Instituto Superior Técnico

**Extended Abstract**

Supervision
Professor Bruno Emanuel da Graça Martins
Professor Pável Pereira Calado

**Abstract.** Given an entity (e.g., a location, a brand, a football club, etc.) mentioned in a text, the task of entity linking is to associate the entity reference to the corresponding entry in an existing knowledge base, such as Wikipedia. This work focuses on applying entity linking to short texts (e.g., tweets), which involves additional challenges, namely the fact that these texts lack on string contextual evidence, or that they may use non-standard vocabulary. The work developed resulted in a prototype system, capable of recognizing and linking entities in short texts to the corresponding Wikipedia page. Throughout this document, some related work in the fields of machine learning and natural language processing will be presented, as well as the approach used to develop the proposed system.

## 1 Introduction

We live in an era of information growth. News articles, TV, social media, blogs or forums, the amount of daily produced information is enormous. In addition, this information covers almost everything, from the elections in the USA to the release of a new Apple product, without forgetting the number of goals scored by Messi and Ronaldo or the election of the new UN's Secretary-General. All these mentions can be associated entities with entities in a knowledge base.

Although there is already developed work in this area, most previous entity linking studies are mainly focused on long texts (e.g., news articles, documents, web pages, etc.) and not on short texts (e.g., tweets or TV subtitles), which leads to poor performances on this type of texts.

This paper presents a system to perform entity linking on tweets, by adapting previously used systems with some changes. As any system of this kind, it starts by performing named entity recognition, followed by named entity disambiguation.

Section 2 presents fundamental concepts and work related with the subject. Section 3 gives a general task definition, while Section 4 describes the approach used in this work, giving details on both subsystems. Section 5 describes the experiments made and Section 6 concludes the paper.

## 2 Fundamental Concepts and Related Work

To better understand this paper, this section presents some concepts that are fundamental to understand, as well as some related work on the subject.

### 2.1 Fundamental Concepts

First, it is important to know what *Named Entity Recognition* (NER) is. NER is a sub task of information extraction that aims to find and classify elements in text into a set of pre-defined categories (e.g., names of persons, organizations, locations).

There are many different ways to perform this task. Traditional NER methods on well-formatted documents heavily depend on a phrase's local linguistic features, such as capitalization, part-of-speech (POS) tags of previous words. One way to do it is by using *Conditional Random Fields* (CRFs).

The second most important concept to know is *Named Entity Disambiguation* (NED). NED is concern with associating mentions to named entities in a textual document to entries in a knowledge base.

The problem has 2 steps: detecting whether a proper name refers to a named entity included in a knowledge base (detection phase) and disambiguating between multiple named entities that can be denoted by the same proper name (disambiguation phase).

To perform NED, it's mandatory to have a repository of entities called *Knowledge Base* (KB). A KB is a repository used to store complex structured and unstructured information and usually contains a large number of entities, which are assigned to semantic types (e.g., a character from a movie, a singer form a music band), as well as an even bigger number of relational facts between entities. YAGO[1], Freebase[2], and DBpedia[3] are good examples of KBs.

### 2.2 Related Work

One example of general Entity Linking can be found at AIDA (Yosef et al. (2011)), which is a robust framework centered around collective disambiguation, exploiting the prominence of entities, similarity between the context of the mention and its candidates, and the coherence among candidate entities for all

[1] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

[2] https://www.freebase.com/

[3] http://wiki.dbpedia.org/

mentions. At Chisholm and Hachey (2015), the authors explored whether web links could replace a curated encyclopedia, obtaining entity prior, name, context, and coherence models from a corpus of web pages with links to Wikipedia. The authors found that combining both models produces the best-known disambiguation accuracy of 0.887 on standard newswire data. Plato (Lazic et al. (2015)) is a probabilistic model for entity resolution that includes a novel approach for handling noisy or uninformative features, and supplements labeled training data derived from Wikipedia with a very large unlabeled text corpus. The approach achieved the best results to-date on TAC KBP 2011[4].

Regarding Named Entity Recognition in Tweets, T-NER (Ritter et al. (2011)) begins with part-of-speech tagging, through chunking, to named-entity recognition. The system doubles F1 score compared with the Stanford NER system. It leverages the redundancy inherent in tweets to achieve this performance, using LabeledLDA to exploit Freebase dictionaries as a source of distant supervision. At Godin et al. (2014), the authors present a semi-supervised system that detects 10 types of named entities, by leveraging 400 million Twitter microposts to generate powerful word embeddings as input features and use a neural network to execute the classification. TwiNER (Li et al. (2012)) presents a 2-step unsupervised system for targeted Twitter stream. In the first step, it leverages on the global context obtained from Wikipedia and Web N-Gram corpus to partition tweets into valid segments using a dynamic programming algorithm. In the second step, it constructs a random walk model to exploit the gregarious property in the local context derived from the Twitter stream.

Finally, for Entity Linking in Tweets, at He et al. (2013), the authors propose a collective inference method that simultaneously resolves a set of mentions. It integrates three kinds of similarity (mention-entry, entry-entry and mention-mention) to enrich the context for entity linking and to address irregular mentions that are not covered by the entity-variation dictionary. Ar Yamada et al. (2015a), the authors do not use an external named entity recognition system. Instead, their system first generates the candidates by using approximate candidate generation. Then, it uses supervised machine-learning to remove irrelevant candidates and resolve them into the corresponding DBpedia URIs. At last, the authors constructed three supervised machine-learning models to detect NIL entity mentions and predict the types of the detected mentions.

## 3 Task Definition

Given a sequence of tweets, denoted $\boldsymbol{T} = (t_1, t_2 ..., t_n)$, the task is, for each tweet, to first identify a sequence of mentions denoted $\boldsymbol{M} = (m_1, m_2 ... m_i)$ and then output a sequence of entities denoted by $\boldsymbol{E} = (e_1, e_2 ..., e_i)$, where $e_i$ is the entity corresponding to $m_i$. Each entity refers to an item of a knowledge base. The KB used is based on Wikipedia. Each mention denotes a sequence of tokens

[4] http://tac.nist.gov/2011/

in a tweet than might possibly have an entity related in the KB.

The following example illustrates the task. Given the tweet *Huge #GameofThrones episode last night! Thank you @HBO for this series! PS: I love you Emilia Clarke* and the mentions *GameOfThrones*, *HBO* and *Emilia Clarke*, the expected output is the Wikipedia links of *Game of Thrones*, *HBO* and *Emilia Clarke*, respectively.

## 4 My Approach

As stated before, Entity Linking requires two main steps: recognizing named entities in text (in this case, tweets) and disambiguate them to link to the correct page in Wikipedia.

In the system developed, the tweets start by entering the main system. Once there, they start by entering the pre-processing step. Once there, operations are executed to increase context for disambiguation. The main assumption made was that if a user posts a tweet, he is possibly interested in that subject, so it's normal that he posts something related to the tweet in other time. This way, it's possible to have 2/3 tweets regarding one subject instead of just 1, which increases the chance of a correct linking between the named entity and the Wikipedia link corresponding. To do this, the Java library for Twitter API Twitter4J [5] was used. The strategy used was to extract the tweet ID and use the library to get the user who posted the tweet. After identifying the user, the same library is used to extract 5 more tweets from that user.

The next step is to clean up the tweets, to try to make them less noisy. To do so, some changes in the tweet's text are performed. This changes included replacement of the standard username (with the @ symbol), removal of some symbols (namely the hashtag # symbol), removal of URLs and removal of the *RT* sentence (used when a tweet mentions a retweet).

### 4.1 Named Entity Recognition

After the pre-processing described in the previous section applied to the tweets, the next step is, as stated, enter the Named Entity Recognition subsystem. NER is the first main step in this task. At this step, the candidate named entities in the tweets are identified.

I chose to adapt Stanford NER[6], a Java implementation of a Named Entity Recognizer. Stanford NER provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. That is, by training own models on labeled data, it's possible to use this code to build sequence models for NER or any other task.

I started by training my own NER model, specifically for tweets. I mixed two different datasets (described in Section 5.1) to use as training data. The dataset

[5] http://twitter4j.org/en/index.html
[6] http://nlp.stanford.edu/software/CRF-NER.shtml

|  | Ritter dataset |  |  | W-NUT dataset |  |  |
|---|---|---|---|---|---|---|
|  | P | R | F1 | P | R | F1 |
| Twitter | 0.9745 | 0.9259 | 0.9496 | 0.9712 | 0.9270 | 0.9486 |
| CONLL 4 Class | 0.3431 | 0.2507 | 0.2897 | 0.3183 | 0.2159 | 0.2573 |
| CONLL 4 Class Caseless | 0.3690 | 0.2507 | 0.2986 | 0.3482 | 0.1905 | 0.2462 |
| Twitter<br>CONLL 4 Class | 0.7693 | 0.9501 | 0.8502 | 0.7861 | 0.9450 | 0.8582 |
| Twitter<br>CONLL 4 Class Caseless | 0.7294 | 0.9444 | 0.8231 | 0.7739 | 0.9418 | 0.8496 |
| Twitter<br>CONLL 4 Class<br>CONLL 4 Class Caseless | 0.6533 | 0.9530 | 0.7752 | 0.6892 | 0.9481 | 0.7982 |

**Table 1.** Results for Named Entity Recognition

combination was divided into 4 categories: LOCATION, PERSON, ORGANIZATION and MISC. Two more models were used during this task: one trained on the CoNLL 2003 *eng.train* and the same model, but in a caseless version. Both this systems are also divided into the same 4 categories, which supports their choice. To train the model for Twitter (using tools provided by the software used), and since the datasets used were all in CONLL format (explained above), the features used to train CONLL 4 Class and CONLL 4 Class Caseless were used, in addition to others to try to improve the model. The new features used include, for example, a feature to include the lemma (canonical form, dictionary form, or citation form) of a word, a feature to record the NGram features that correspond to a String and reuse rather than recalculating if the String is seen again or the use of gazettes.

In addition to all those features, two more decisions were made for training the Twitter model: all usernames (tokens who started with @) where classified as PERSON and hashtags where not used in the training data. The pre-process step also removed some invalid chars in the tweets.

I also decided to combine the models, by using the class $NERClassifierCombiner$, available at Stanford NER, which allows for multiple CRFs to be used together. This class combines the models and outputs the named entities identified, based on the order in which the models are (i.e., the first model has higher power of decision, followed by the second and so on). Since the focus of this work is on tweets, I always decided to input the Twitter model in first place.

To test the models to be used (Twitter model, CONLL 4 class and CONLL 4 class caseless) and their possible combinations, Stanford NER tools where again used. To obtain the metric's results, the script *conlleval*[7] was used. This script evaluated the files and outputted the Precision, Recall and F1 scores. All systems where evaluated for both datasets used separately. The results are presented at Table 1.

[7] http://www.cnts.ua.ac.be/conll2000/chunking/output.html

By analyzing Table 1, we see that the Twitter model achieves the higher values of Precision and F1 for both datasets, while the combination of the three models achieves the higher Recall. Since the Twitter model is ”used” in all the best score achieved, I decided to use the combination of the three models to perform the entity linking task. Although Precision and F1 drop in the combination, Recall is very high.and both CONLL 4 Class and CONLL 4 Class Caseless are important to identify some entities (although they are not Twitter specific).

### 4.2 Entity Linking

To this subsystem, software previously available was again used. I chose to use AIDA[8], a framework and online tool for entity detection and disambiguation. AIDA was developed to perform entity linking on structured texts (e.g., news articles), which is proved to perform poorly in tweets. Since the source code is available at GitHub[9], it was possible to adapt the system to increase its performance on short texts (in this case, tweets).

AIDA first starts by identifying noun phrases that potentially denote named entities, using the combination of three models described in the previous section. Then, the system uses a graph-based approach. The graph is constructed with mentions and their candidate entities as nodes. The graph has two types of edges: mention-entity and entity-entity. The goal is to reduce this graph to a dense sub-graph where each mention node is connected to one and only one candidate entity node. Once the graph is constructed, they use a two-step greedy algorithm to compute the sub-graph. In the first step, this procedure identifies the entity node that has the lowest weighted degree. In the second step it removes that node and its incident edges from the graph unless it is the last remaining candidate entity for one of the mentions.

The authors also apply some features and measures for computing the edge weights. The mention-entity similarity is computed as a linear combination of two ingredients. The first is the prominence of an entity, which acts as a prior probability for each potential mapping. The second ingredient is based on the overlap between a mention’s context and a candidate’s entity context.

AIDA offers four different configurations to perform entity linking and all of them were used to test the performance of the system: PRIOR, LOCAL, GRAPH and GRAPH-KORE.

The first test performed was to check the values of the different measures (Precision, Recall and F1) without performing any changes in AIDA (except the models, which were already loaded to increase performance for tweets). Using Java, I developed an algorithm that ”breaks” the dataset and sends each tweet individually for AIDA. This step was necessary because since AIDA was developed for rich text, if all the tweets where sent at once, it would consider it as

[8] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/aida/
[9] https://github.com/yago-naga/aida

|  | Precision | Recall | F1 |
|---|---|---|---|
| GRAPH | 0.5541 | 0.5235 | 0.5383 |
| LOCAL | 0.5725 | 0.5620 | 0.5672 |
| PRIOR | 0.5739 | 0.5648 | 0.5693 |
| KORE | 0.5604 | 0.5357 | 0.5478 |

**Table 2.** Results produced by the original AIDA system, without introducing changes

a complete text, which would ruin the context of every tweet. The results obtained are presented at Table 2.

By looking at the table, we see that PRIOR is the configuration that achieves the higher results for all three measures. The results are not very high, which may have some explanations. The first one is the fact that usernames don't suffer changes for processing, i.e., the system tries to link a username (which is caught by the NER models, as explained before) in his standard version (including the @ symbol), which will hardly achieve success. The second one is similar to the usernames and is related to the hashtags, The system also tries to link the hashatgs with the # symbol, which also makes it hard to achieve success. Another motives might be related with some noise (specially abbreviations and some symbols) in the tweets, which might led the NER models to identify some entities which it shouldn't or to not identify others that should be caught.

Taking the results into account, the next step was to try to increase the performance. To do so, the pre-processing steps mentioned above where performed before executing the system. The Java algorithm was modified to send to the system not only the individual tweet, but also the five more tweets added in the pre-processing. In this case, since the tweets belong to the same user who tweeted the original tweet, AIDA is supposed to consider them as context for the original tweet, which is why they are all processed at once by AIDA. The results for this final experiment and their discussion is presented at Section 5.2.

## 5 Experiments

To use AIDA with YAGO2, I downloaded the repository provided on AIDA website as a Postgres dump and import it into my database server. The dump used was AIDA_entity_repository_2014-01-02v10, which was built from a Wikipedia dump from 2014. For recognizing named entities, I loaded the previous mentioned combination of three different NER models: a model trained specifically for Twitter, CONLL 4 Class and CONLL 4 Class Caseless.

### 5.1 Datasets

Different datasets where used accordingly to the subsystem. As said before, the combination of two different datasets was used to train the Named Entity Recognition subsystem. The first one was the dataset used by Ritter et al.

(2011). The combination of both train and eval sets ended up with 2400 tweets, with approximately 48K tokens. The second dataset was the one used at the ACL 2015 Workshop on Noisy User-generated Text (W-NUT) - NER Shared Task[10]. It contains approximately 35K tokens. As said before, both datasets were annotated with 4 types of categories: PERSON, LOCATION, ORGANIZATION and MISC.

For the Disambiguation of Entities Subsystem, I used the dataset from Making Sense of Microposts (#Microposts2015)[11]. I used the 2015 version instead of the 2016, because the most recent dump of Wikipedia available for AIDA is from 2014, which would result in many non know entities with the 2016 version. The dataset comprises tweets extracted from a collection of over 18 million tweets. They include event-annotated tweets provided by the Redites project, covering multiple noteworthy events from 2011 and 2013 (including the death of Amy Winhehouse, the London Riots, the Oslo bombing and the Westgate Shopping Mall shootout), and tweets extracted from the Twitter firehose in 2014.. Instead of the test set, I had to use the train set to evaluate the system, since almost every tweet from the test set was not available anymore, which would make impossible to increase context by getting more tweets from the user. The initial set had 3498, but ended up with 2644 after checking which ones were still available to be used the way I needed.

### 5.2 Results

Table 3 shows that all configurations improved their results. This proves that the changes performed were useful to increase the performance of AIDA. While PRIOR achieved the best performance on the first execution, at this one LOCAL achieved the higher results. Regarding each metric individually, Precision was the one with the lower increase values. This might be explained by the fact that Precision includes False Positives in its calculations (i.e., entities that were not supposed to be caught and linked), and although the changes in the pre-processing where meant to reduce noise (which had success), the truth is that it also made False Positives ”easier” to appear. On the other hand, Recall and Precision achieve good increase values (which for F1 is directly proportional to an increase of Recall). Since Recall uses False Negatives (i.e., entities that are supposed to be caught, but linked incorrectly), the improvements on their value supports the thesis that the changes performed increased the number of correct entities caught. I compared the best performance (LOCAL in this case) with some systems from Section 2.2, whose results can be seen at Table 4. It’s possible to see that my framework, with all the changes performed, outperforms the system by Yamada et al. (2015b) in all three metrics. In addition, my framework also outperformed the Recall values from Guo et al. (2013) and He et al. (2013), while also achieved a closer F1 value to the one at Guo et al. (2013).

[10] http://noisy-text.github.io/2015/ner-shared-task.html
[11] http://scc-research.lancaster.ac.uk/workshops/microposts2015/

|  | Precision | Recall | F1 |
|---|---|---|---|
| GRAPH | 0.5978 | 0.6575 | 0.6262 |
| LOCAL | 0.6117 | 0.6942 | 0.6503 |
| PRIOR | 0.6050 | 0.6757 | 0.6384 |
| KORE | 0.6035 | 0.6732 | 0.6365 |

**Table 3.** Results produced by AIDA, after introducing the proposed changes

|  | Precision | Recall | F1 |
|---|---|---|---|
| My Framework | 0.6117 | 0.6942 | 0.6503 |
| Yamada et al. (2015a) | 0.8570 | 0.8230 | 0.840 |
| Yamada et al. (2015b) | 0.5766 | 0.5522 | 0.5641 |
| Guo et al. (2013) | 0.7880 | 0.5990 | 0.6800 |
| He et al. (2013) | 0.7640 | 0.6750 | 0.7110 |

**Table 4.** Comparison between the best performance of my framework (after changes) and other systems

## 6 Conclusions

In this work, I presented an alternative use for AIDA, a framework and online tool for entity detection and disambiguation. Since both Stanford NER and AIDA were developed for rich text, I started by train a model specifically for Twitter. This model achieved very high results, and I decided to combine it with another two models: CONLL 4 Class and CONLL 4 Class Caseless. After loading this models to AIDA, two performances were made: one with standard configurations and another with changes performed.

Results improved from the first to the second execution, which allows me to conclude that the changes performed were effective. Another contribution is the Twitter model. It would be very hard to obtain these results if a model trained with Twitter data was not included, since tweets suffer from a lack of context and noisy data. Finally, since Recall improved in values higher than 0.1 in all four configurations of AIDA, it supports that my framework misses less linking of entities.

## References

Chisholm, A. and Hachey, B. (2015). Entity disambiguation with web links. In *Transactions of the Association for Computational Linguistics*, volume 3, pages 152–155.

Godin, F., Vandersmissen, B., Jalalvand, A., Neve, W. D., and de Walle, R. V. (2014). Alleviating manual feature engineering for part-of-speech tagging of twitter microposts using distributed word representations. *In Workshop on Modern Machine Learning and Natural Language Processing (NIPS 2014)*, pages 146–153.

Guo, S., Chang, B. M.-W., and Kiciman, E. (2013). To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1020–1030.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., and Wang, H. (2013). Learning entity representation for entity disambiguation. In *Annual Meeting of the Association for Computational Linguistics*, pages 30–34.

Lazic, N., Subramanya, A., Ringgaard, M., and Pereira, F. (2015). Plato: A Selective Context Model for Entity Resolution. In *Transactions of the Association for Computational Linguistics (TACL)*, volume Volume 3, pages 503–515.

Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A., and Lee, B.-S. (2012). TwiNER: Named Entity Recognition in Targeted Twitter Stream. In *Proceedings of the 35th international Association for Computational Linguistics SIGIR conference on Research and development in information retrieval*, pages 721–730.

Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011). Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534.

Yamada, I., Takeda, H., and Takefuji, Y. (2015a). An end-to-end entity linking approach for tweets. In *Proceedings of the the 5th Workshop on Making Sense of Microposts co-located with the 24th International World Wide Web Conference (WWW 2015), Florence, Italy, May 18th, 2015.*, pages 55–56.

Yamada, I., Takeda, H., and Takefuji, Y. (2015b). Enhancing Named Entity Recognition in Twitter Messages Using Entity Linking. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text*, pages 136–140.

Yosef, M. A., Hoffart, J., Bordino, I., Spaniol, M., and Weikum, G. (2011). AIDA: an online tool for accurate disambiguation of named entities in text and tables. In *Proceedings of the International Conference on Very Large Databases*, volume 4, pages 1450–1453.