

Named Entity Recognition and Disambiguation in Tweets

Master Thesis

Soumya Ranjan Patra

Supervisors:

dr. Mykola Pechenizkiy
Erik Tromp MSc

Committee Members:

dr. Mykola Pechenizkiy
prof. dr. Paul De Bra
dr. Alexander Serebrenik
Erik Tromp MSc

Eindhoven, August 2014

Abstract

Social media has grown exponentially over the past few years. Users are generating far more unstructured content than ever before. Successful companies are also very active in social media analysing these data for their marketing campaigns. But the informal and noisy nature of such data makes it quite difficult to extract meaningful information out of them.

In this thesis, we investigate the problem of named entity recognition and disambiguation in tweets. Given a tweet written in English, the task focuses on extracting all the named entities inside and assigning each entity with a reference link. It has potential applications in online product monitoring, sentiment analysis, electoral predictions and other social media analytics.

We present a five step approach for this problem consisting of tokenization, Part-of-speech tagging, normalization, mention extraction and disambiguation. First three steps are used as preprocessing steps for mention extraction and disambiguation which are the core components of our approach. For tokenization and part-of-speech tagging, we use the regular expression tokenizer and tagger developed by Owoputi et al. In the third step, we propose a normalization algorithm which uses Brown clusters and Microsoft web ngram language model to normalize the tweets. In the next step, our mention extractor extracts possible candidate entities with the help of POS tags and ngram matching against Freebase. Finally, using the intra-tweet context and relationship among entities, our disambiguation algorithm assigns appropriate reference links to these extracted entities.

To evaluate our approach, We carry out several experiments and benchmark our complete process against different state of the art extraction and disambiguation systems. We also evaluate the importance of individual steps by measuring the overall performance degradation by leaving that step out of the pipeline. Finally, we present a case study demonstrating how our approach can be applied practically in a real world scenario.

Acknowledgements

I would like to express my sincere gratitude to dr. Mykola Pechenizkiy for supervising and guiding me throughout my thesis. Particular thanks to my daily supervisor Mr. Erik Tromp for his constant support and motivation. His guidance and insightful comments have helped me greatly during this period. I would also like to thank the rest of my thesis committee: prof. dr. Paul De Bra and dr. Alexander Serebrenik, for their help in assessing my thesis.

Last but not least, I am thankful to my family and friends for their priceless encouragement and support.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Practical Applications	3
1.3	Problem Formulation	4
1.4	Our Approach and Main Results	5
1.5	Organization of The Thesis	6
2	Approach	7
2.1	Tokenization	7
2.1.1	Problem Formulation	8
2.1.2	Related Work	8
2.1.3	Owoputi et al. Regular Expression Tokenizer	8
2.2	Part-Of-Speech Tagging	8
2.2.1	Problem Formulation	9
2.2.2	Related Work	9
2.2.3	Owoputi et al. Tagger with Unsupervised Features	10
2.3	Normalization	10
2.3.1	Problem Formulation	11
2.3.2	Related Work	11
2.3.3	Our Approach	11
2.4	Mention Extraction	14
2.4.1	Problem Formulation	15
2.4.2	Related Work	15
2.4.3	Our Approach	16
2.5	Disambiguation	17
2.5.1	Problem Formulation	18
2.5.2	Related Work	18
2.5.3	Our Approach	19
3	Experimental Evaluation	28
3.1	Motivation	28
3.2	Datasets	28
3.3	Complete Process Evaluation	29

3.3.1	Entity Extraction Experiments	29
3.3.2	Entity Disambiguation Experiments	33
3.4	Evaluation of The Importance of Each Step	39
3.4.1	Without POS Tagging	39
3.4.2	Without Normalization	41
3.4.3	Without Disambiguation	41
3.5	Evaluation of Normalization	41
4	Case Study : Filtering Targeted Tweets with Disambiguation	44
5	Conclusion and Future Work	46
	Appendices	51
A	AIDA Disambiguation Errors	52
A.1	AIDA Disambiguation Error : Capitalization	52
A.2	AIDA Disambiguation Error : Grammatical Error	53
A.3	AIDA Disambiguation Error : In Complete Setting	54

List of Tables

1.1	sample tweets showing the challenges in performing information extraction tasks on tweets	2
3.1	Ritter et al. test dataset statistics	30
3.2	Comparison of performances with other approaches on Ritter et al.(2011) test dataset	32
3.3	Habib et al. dataset statistics	34
3.4	Disambiguation performances of our approach benchmarked against TAGME and different settings of AIDA	36
3.5	Showing the impact of removing individual steps on the overall performance (<i>T: Tokenization, P: POS tagging, N:Normalization, E: Mention Extraction, D: Disambiguation</i>)	39
3.6	Examples of some errors introduced due to the removal of POS tagging	41
3.7	Normalization Experiments	42
4.1	Showing the performance of our system in correctly filtering a targeted entity out of ambiguous entities	45
4.2	Some representative examples from the case study	45

List of Figures

1.1	A simple visualization of the expected goal of the overall system.	5
2.1	Pipeline for named entity recognition and disambiguation	7
2.2	An example showing the normalization process	13
2.3	Examples showing most similar entities from the entity vector model	22
2.4	Sorted list of candidate similarities for mention tuple (<i>Rick Ross, Drake</i>)	24
2.5	F-measure for different similarity thresholds on Ritter et al. sample dataset	25
2.6	Sorted list of candidate similarities for mentions ' <i>Catherine Hardwicke</i> ', ' <i>Maze</i> ', ' <i>James Dashner</i> '	25
2.7	Sorted list of candidate similarities for mention tuple (' <i>Richard Pryor</i> ', ' <i>Hamlet</i> ')	27
3.1	Plot showing the performance comparison of our approach with other baselines and Ritter et al.	33
3.2	Plot showing the performance comparison of our approach with different settings of AIDA	37
3.3	Plot Showing the impact of removing individual steps on the overall performance	40
3.4	Plot showing the effect of introducing spelling errors in our approach and AIDA complete system	43
A.1	Screen shots from AIDA online tool showing how small case affects disambiguation	52
A.2	Screen shots from AIDA online tool showing how a grammatical error affects disambiguation	53
A.3	Screen shots from AIDA online tool showing disambiguation error with the com- plete setting	54

Chapter 1

Introduction

Named entity recognition (NER) and named entity disambiguation (NED) are two important information extraction tasks in the domain of natural language processing. A *named entity* is basically a phrase inside a text that belongs to a predefined category such as person, location, organization, movie name, product name etc. NER is the task of identifying such named entities inside a given text. NED, on the other hand, focuses on identifying the actual identity of the extracted entity. For example, given an input text *Philips hired 500 employees this quarter but ASML only hired 200*, a recognizer component is expected to extract the phrases *Philips* and *ASML* as named entities. For the same example, a disambiguation component is expected to figure out that the extracted entity *Philips* is actually referring to the *Philips organization* and not some person named *Gerard Philips*.

In this thesis, we study the problem of named entity recognition and disambiguation in tweets. In section 1.1, We provide a brief motivation about why information extraction from social media is an interesting and a challenging problem. We talk about some of the practical applications of the system in section 1.2. In section 1.3, we formulate the problem for this thesis. The main results presented in this thesis are summarized in section 1.4. In section 1.5, we outline the organization of the thesis.

1.1 Motivation

The impact of social media can not be ignored today. Everyday, users are generating vast amount of unstructured information through various social networking websites such as Facebook, Twitter, Tumblr. According to a twitter blog¹, more than 500 million tweets were being generated each day in 2013. Analysing these giant sources of data can help us discover new insights about a wide range of topics. For example, collecting tweets targeted towards products of a company and

¹<https://blog.twitter.com/2013/new-tweets-per-second-record-and-how>

performing sentiment analysis on them can provide knowledge about online reputation of the company. Some other areas where these data can be utilized are market research, competitor analysis and predictive analytics.

NER and NED are quite well studied problems in natural language processing and have many real world applications. Artificial Intelligence systems such as question answering applications use disambiguation algorithms to disambiguate between ambiguous entities. Disambiguation is also a central component in semantic search engines which help users to retrieve fast and relevant results by considering the search query context. For instance, a query *michael invincible* in Google² outputs *Michael Jackson* in the top results whereas a query *michael F1* outputs *Michael Schumacher* in the top results. With the help of its disambiguation component, the search engine is able to accurately predict that in the context of *Invincible* (the studio album by Michael Jackson), *michael* is referring to *Michael Jackson* whereas in the context of *F1* race, *michael* is referring to *Michael Schumacher*.

Social media texts are highly informal and unstructured which makes information extraction extremely challenging. Table 1.1 shows some of the main challenges that we have to deal with while working on tweets. Capitalization is a very important feature for named entity extractors in formal texts. But, it can not be considered as a reliable feature in tweets because people often don't care about capitalization. Due to the 140 character limit, abbreviations, shortened texts, spelling and grammatical mistakes are also very common in tweets.

To extract useful knowledge from these texts, they need to be organized in a structured way first. NER and NED can help a great deal in this structuring process and make the noisy text suitable for knowledge extraction.

No.	Challenge	Example Tweet
1	missing capitalization	george #Clooney did a great job in <i>gravity</i> .
2	unreliable capitalization	#JLO WAITIN TO SEE JENNIFER 'S NEW CAR
3	abbreviation	HIMYM starts in 10 mins
4	shortened texts	@rk89 pls dnt 4get to brng ma ifun
5	spelling mistakes	sry bt you r wrng this time mr ptersan

Table 1.1: sample tweets showing the challenges in performing information extraction tasks on tweets

²<https://www.google.com> - Search engine

1.2 Practical Applications

In this section, we explain some of the scenarios where NER and NED in tweets can have a significant impact. With these examples, we want to stress the need of designing information extraction systems which are able to extract meaningful information out of noisy domains such as tweets.

Online Product Monitoring

It is often the case that products from different companies have ambiguous names. For example, a term *Monster* can refer to carrier website [monster.com](http://www.monster.com/)³, a product manufacturing company⁴ or an energy drink⁵. Similarly, *Nexus* is a product which can either refer to Google's smart phones⁶ or bicycle parts produced by an American company called Shimano⁷. Suppose Shimano wants to track tweets about a newly released *Nexus* bike part to know how people are reacting towards it in social media. They will get an overwhelming amount of tweets referring to Nexus smart phones which are quite popular these days. It would be quite a tedious task for them to filter out relevant tweets about their product. In such cases, they can use an automated disambiguation system which can filter out relevant tweets about Nexus bike parts by considering the context inside the tweet.

Election Polls

In today's world, Twitter is playing an active role in election predictions and surveys. According to a study in Indiana University⁸, there is a direct correlation between the number of times an electoral candidate is mentioned in tweets in the months before election and his success in the elections. Since, these kinds of analysis techniques do not require vast amount of funds, many electoral parties are using them. Let us consider an example of a politician Bill Gates⁹ who shares his name with the famous American business magnate Bill Gates¹⁰. Suppose the party wants to perform the above mentioned survey on the politician. To make a correct prediction about the success of the candidate, they must only consider the number of tweets about the politician. Manually doing this task can be tiresome. One can use disambiguation system such as ours to automate the task and help in accurate predictions.

³<http://www.monster.com/>

⁴[http://en.wikipedia.org/wiki/Monster_\(company\)](http://en.wikipedia.org/wiki/Monster_(company))

⁵<http://www.monsterenergy.com/>

⁶<http://www.google.com/nexus/>

⁷http://www.shimano.com/publish/content/global_cycle/en/us/index/products/0/nexus.html

⁸<http://www.theatlantic.com/technology/archive/2013/08/a-new-study-says-twitter-can-predict-us-elections/278612/>

⁹[http://en.wikipedia.org/wiki/Bill_Gates_\(politician\)](http://en.wikipedia.org/wiki/Bill_Gates_(politician))

¹⁰http://en.wikipedia.org/wiki/Bill_Gates

Sentiment Analysis

Sentiment analysis in tweets is another field which can benefit from disambiguation systems. These techniques extract subjective information from texts to find out peoples' opinion about other people, products, movies, books etc. Sentiment analysis can be applied to individual tweets. But, in most of the cases, set of tweets about the same topic are aggregated to calculate an overall sentiment score. This score gives us an idea of how positive, negative, or neutral people feel about that topic. To get a reliable sentiment score, we need to make sure that all the aggregated tweets are talking about the same topic and not about other ambiguous topics with same name. For example, if we want to analyse peoples' opinion about the actor *Adam Scott*¹¹, we need to make sure that our dataset only contains tweets about the actor and not about the famous golfer¹² with the same name. This task can be easily done by performing disambiguation to filter the relevant tweets.

1.3 Problem Formulation

NER and NED are very different tasks. While the former deals with finding the occurrence of a named entity, the later deals with finding the reference of the same entity. We formally define our problem as the follows.

Given a tweet written in English, does it contain any named entities and if so, which semantic entity it is referring to ?

Traditional NER approaches extract named entities out of texts and assign them to a category class such as person, organization or location. Often, these classes are predefined and have a fixed number of categories. These systems perform quite well for specialized information extraction tasks where the focus is only on extracting certain types of entities . However, in tweets, we come across entities which can belong to any of the large set of real world entity types. For example, most of the NER approaches do not predict classes such as movie names, song names, river names etc. In our approach, we focus on developing an entity extractor which is able to extract any generic type of entity from a given tweet.

To disambiguate an extracted entity, most of the current approaches use some kind of knowledge base. The disambiguation component finds the most appropriate KB page that best describes the entity in the referred context and assigns it's link to the entity. Majority of the known approaches assign Wikipedia links. We assign entities with Freebase links in our approach. The advantage of using Freebase links is that it is very easy to get the corresponding Wikipedia link with the help of a simple API query. Figure 1.1 shows an abstract visualization of the overall system and the expected output.

¹¹[http://en.wikipedia.org/wiki/Adam_Scott_\(actor\)](http://en.wikipedia.org/wiki/Adam_Scott_(actor))

¹²[http://en.wikipedia.org/wiki/Adam_Scott_\(golfer\)](http://en.wikipedia.org/wiki/Adam_Scott_(golfer))

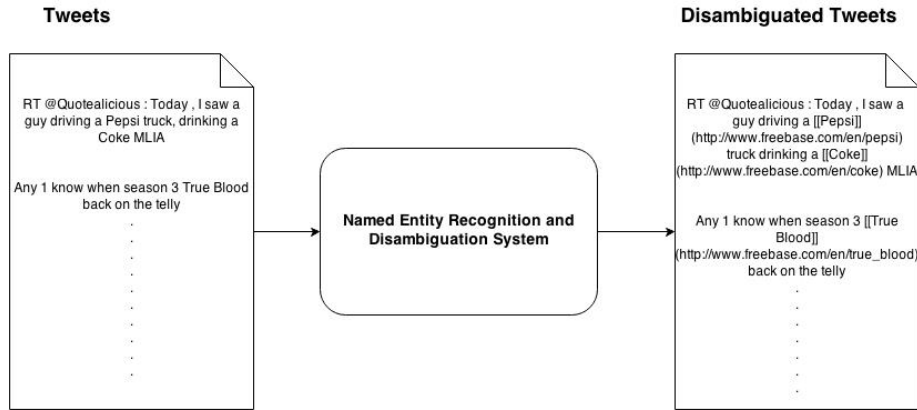


Figure 1.1: A simple visualization of the expected goal of the overall system.

1.4 Our Approach and Main Results

Our approach to extract and disambiguate named entities from tweets consists of 5 steps. We show this 5 step pipeline in fig. 2.1. First three steps are preprocessing steps but are important parts of the whole setting. In the first two steps, we tokenize the tweets and assign POS tags to the tokens. We use the state of the art tokenizer and POS tagger designed by Owoputi et al.(2013)[21] for these two steps. In the next step, we normalize the tweets with our normalization algorithm. Normalization is generally not included in traditional entity extraction and disambiguation systems. But, as we will show in our experiments, when dealing with noisy information such as tweets, this step can have a significant impact on the performance of the whole system. The final two steps are the core components of the system and perform entity extraction and disambiguation tasks respectively. All these steps are explained in detail in chapter 2.

We propose a generic system which is able to extract named entities from tweets and correctly assign reference links to them. To evaluate the performance of the system and its components, we perform various experiments in chapter 3. For the evaluation of our entity extractor, we compare our results with other baselines and competitors in section 3.3.1. We show through these experiments that our NER system performs better than other approaches. We also analyse these results and suggest how they can be further improved. Similarly, to evaluate the performance of our NED system, we benchmark against a competitor named AIDA system in section 3.3.2. We show that our approach outperforms the competitor in different settings. We again analyse these results to reflect upon why disambiguation approaches suffer on noisy data and what can be done to improve the disambiguation performance.

To justify the inclusion of individual steps in the complete system, we perform a different set of experiments in section 3.4, where we remove a step from the complete system and measure the degradation in performance. This gives us an idea about the importance of individual steps in the complete setting. These modified pipelines are benchmarked against our complete system on Ritter et al. test set results obtained in section 3.3.1. We show from these experiments that our NER performance drops significantly if we remove the POS tagging and disambiguation steps from the pipeline. We also show that using normalization in our approach helps us disambiguate entities even with spelling errors.

1.5 Organization of The Thesis

The remainder of the thesis is organized as follows. In chapter 2, we describe the problem in detail and discuss our approach to solve the problem. In chapter 3, we evaluate our overall approach as well as the choice of methods for each of the steps in the processing pipeline. In chapter 4, we employ our system in a real world scenario and talk about the practical applications of our work. Finally, in chapter 5, we present a conclusion and suggest how our system can be further enhanced for even better performance.

Chapter 2

Approach

As shown in fig. 2.1, our approach uses a five step pipeline to solve the entity extraction and disambiguation problem. First three steps i.e. *tokenization*, *POS tagging* and *normalization* are non-core yet key components of our approach. They serve as preprocessing steps. Last two components *mention extraction* and *disambiguation* are the core components of our approach. The pipeline takes a tweet written in English as input and returns the disambiguated tweet with named entities and their proper references. We make no assumptions on the presence of named entities inside a tweet before processing. For a tweet without any entity, this approach is expected to return an empty list.

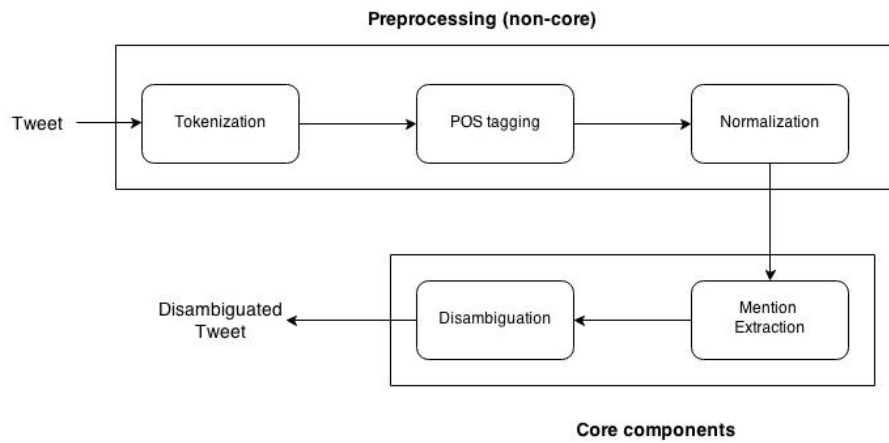


Figure 2.1: Pipeline for named entity recognition and disambiguation

2.1 Tokenization

The first step in our approach is tokenization. It is a necessary preprocessing step in all the natural language processing tasks. We describe the problem formulation in

section 2.1.1. Some of the related works are mentioned in section 2.1.2 and finally our approach to this sub problem is explained in section 2.1.3.

2.1.1 Problem Formulation

The main goal of tokenization in our approach is to split a given tweet into meaningful semantic units (tokens) which can be words, phrases or symbols. These tokens are later assigned POS tags in the next step. Because of their noisy nature, tokenization in informal texts is comparatively difficult than in formal texts. The presence of hashtags(#), user names(@), short urls and emoticons in tweets demand the need of sophisticated tokenization techniques.

2.1.2 Related Work

Text tokenization is a well researched topic and a number of tokenizers are available with reasonably good performance. Stanford tokenizer uses Jflex to generate a lexical analyser for tokenization and provides state of the art performance on formal texts. Penn Treebank Tokenizer uses a SED script using regular expressions. Most of the other tokenization approaches also use regular expressions.

2.1.3 Owoputi et al. Regular Expression Tokenizer

We use a Java-based tokenizer implemented by Owoputi et al.(2013)[21] in our approach. As described in section 2.1.2, there are many other alternative approaches which are capable of tokenizing text with satisfactory results. But the main reason to prefer this tokenizer over others is that it is the only known approach which implements Twitter specific regular expressions to match hashtags, usernames and large range of emoticons and also manages to achieve state of the art performance in tweets. Example 2.1.1 shows the output of the tokenizer for a sample tweet. We can see that the tokenizer is able to successfully split the tweet into meaningful tokens.

Example 2.1.1. Original input : *RT IMDb: As expected, 'Harry Potter and the Deathly Hallows' leads the Friday #boxoffice, taking in an est. \$61.2 million. http://imdb. ...*

Tokenized output : *RT IMDb : As expected , ' Harry Potter and the Deathly Hallows ' leads the Friday #boxoffice , taking in an est . \$61.2 million . http://imdb*

2.2 Part-Of-Speech Tagging

After tokenization, we perform part of speech tagging on the tweets. This step plays an important role in the entity extraction and disambiguation phases later.

Based on the tags, the tweet is segmented into partitions and these partitions are forwarded to the entity extractor as inputs. The POS tags are also used to increase the recall of the entity extractor by adding the tokens which are tagged as proper nouns. In the experiments, we also evaluate the importance of this step by removing it from the pipeline and then measuring the performance degradation.

We explain the problem formulation in section 2.2.1. Some of the related works are mentioned in section 2.2.2 and finally our approach to this sub problem is explained in section 2.2.3.

2.2.1 Problem Formulation

Given a tokenized tweet, part of speech tagging determines which grammatical category (proper noun, verb, adjective etc.) do the tokens belong to. It assigns POS tags to the tokens and these tags are quite frequently used as linguistic features in natural language processing applications. POS tagger is generally implemented as a supervised learning task where a model is trained on a tagged dataset. To assign a tag to a token, the model considers various features. Some of the features are capitalization, tags of surrounding words, the surrounding words themselves, whether the word contains digits etc. Based on these features, the model predicts the most appropriate grammatical tag for the token and assigns it with that tag.

The main challenge of a POS tagger is to use the context and assign correct tags to ambiguous tokens. For example, in a sentence *when is the next match?*, *match* is used as a noun whereas in *did the blood group match?*, *match* is a verb. The tagging becomes even more challenging in informal texts because of grammatical and spelling errors, OOV words and abbreviations. Therefore, extra care needs to be taken while choosing the features for the model. For example, capitalization is not an ideal feature for informal texts because people often ignore capitalization in short texts. Also, people tend to use a large number of adjectives in tweets which can make it difficult to consider the surrounding word tags as features. Therefore, while tagging tweets, it is not just enough to use the standard features. The tagger must also consider some other features to accurately predict the tags.

2.2.2 Related Work

Part of speech tagging is a well researched problem and there are a large number of taggers available which use different machine learning approaches. Stanford tagger[24] implemented a Maximum Entropy Model and was able to achieve 97% accuracy on the Penn Treebank dataset. But it could only achieve 80% accuracy on Ritter et al.(2011)[23] tweet corpus. Brants(2000)[2] proposed Hidden Markov Model to solve the problem. Ritter et al. used Conditional Random Fields to train a model with manually annotated tweets. Their model outperformed other baselines and performed tagging with an impressive 88% accuracy on tweets. Some

other supervised approaches such as SVM and Perceptron learning have also been applied successfully to the task and achieved good performances. There have also been proposals of using unsupervised learning methods which involve the use of contextual similarity and brown clustering.

2.2.3 Owoputi et al. Tagger with Unsupervised Features

We use the tagger developed by Owoputi et al.(2013)[21]. The main motivation to prefer this tagger over other approaches described in section 2.2.2 is that it is specifically designed to process tweets and achieves state of the art performance. On top of traditional supervised features, the model also uses unsupervised word clusters and some other lexical features. These word clusters are generated by applying Brown clustering algorithm¹ on unlabelled tweets. The model is implemented as a Conditional Random Field[16] which uses manually tagged tweets as training data. The feature set not only includes traditional tagging features such as surrounding word tags, presence of digits, but also uses twitter specific features such as presence of #, @ etc.

The use of distributional word similarity as a feature helps the model achieve good performance even with a small training set. Their model outperforms Stanford tagger on their dataset and also manages to achieve 90% accuracy on Ritter et al. (2011) tweet corpus. Example 2.2.1 shows the output of the tagger for a sample tokenized tweet. We can see that the tagger is successfully able to predict *Harry* and *Potter* as proper nouns(\wedge). The detailed list of tags as well as the list of used features can be found in Owoputi et al.(2013)[21] paper.

Example 2.2.1. Tokenized input : *RT IMDb : As expected , ' Harry Potter and the Deathly Hallows ' leads the Friday #boxoffice , taking in an est . \$61.2 million . http://imdb*

Tagged output : *RT/~ @IMDb/@ :/~ As/P expected/V ,/, /, Harry/\ Potter/\ and/& the/D Deathly/A Hallows/\ /, leads/V the/D Friday/\ #boxoffice/N ,/, taking/V in/P an/D est/N ./, \$61.2/\$ million/N ./, http://imdb/U ./, .../~*

2.3 Normalization

The third step in our pipeline is normalization of tweets. Most of the current NER approaches for tweets operate under the assumption of no spelling and grammatical errors. However, tweets are quite noisy and such errors are quite common. We wanted to know if performing normalization can improve the performance of entity extractors in tweets. Hence, we include an additional component to normalize the

¹http://en.wikipedia.org/wiki/Brown_clustering

tweets before they get to the extraction phase. Later, in our experiments in section 3.5, we show that normalization can indeed improve the performance of NER and NED tasks.

In section 2.3.1, we describe the problem formulation. Some of the related work are mentioned in section 2.3.2 and finally our approach to this sub problem is explained in section 2.3.3.

2.3.1 Problem Formulation

Given a tokenized and tagged tweet, the task of the normalization component is to locate the OOV words and transform them to their correct forms. This is quite a challenging sub-problem in itself. For instance, the word *tomorrow* is often abbreviated as *2mrw* in tweets. It is very difficult to correct such errors by simple spelling correctors. Therefore, a more sophisticated approach is needed for this task.

2.3.2 Related Work

Normalization is not so well researched in tweets. Aw et al.(2006)[1] proposed a phrase-based statistical model to normalize SMS texts. It handled abbreviations well, but there was no component for spelling correction. Hence it is not suitable for tweets. Kaufmann and Kalita (2010)[14] proposed a different approach and suggested to use statistical machine translation for normalization. The challenge with this approach is that it needs two large parallel tweet corpora to train the model, one with noisy tweets and another with their corresponding normalized forms. This again will need considerable manual effort to build the parallel corpora. Han et al. (2011)[12] generated a confusion set of possible candidates for each OOV word based on character edit distance and phonemic edit distance. Then they used a language model and dependency based features to select the best. Our work is partly inspired by their research.

2.3.3 Our Approach

Our approach to normalization is unsupervised and makes use of freely available online resources such as Enchant dictionary², Microsoft web n-gram³ and word clusters produced by Brown clustering⁴. Algorithm 1 explains our normalization process.

An example for the complete normalization process can be seen in Fig. 2.2. Let us consider a simple tweet *the expndibles is a gr8 movie*. For each token inside

²<http://pythonhosted.org/pyenchant/api/enchant.html>

³<http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx>

⁴http://www.ark.cs.cmu.edu/TweetNLP/cluster_viewer.html

Algorithm 1 Normalization

```
1: procedure NORMALIZE(tweet)
2:   Initialize an empty string normalized_tweet
3:   for all token in tweet do
4:     if token is not a proper noun then
5:       Check if token is an OOV word           ▷ Enchant dictionary check
6:       if token is not OOV then
7:         Append token to normalized_tweet
8:       else                                     ▷ token is OOV word
9:         Check if token is in any brown cluster
10:        if token in cluster then
11:          Retrieve candidates in cluster which are IV and
           $Levenshtein \leq 2$  or  $Metaphone \leq 1$  from token
12:          Append most frequent candidate to normalized_tweet
13:        else                                   ▷ token not in cluster
14:          Retrieve suggestions from dictionary ▷ Enchant suggestions
15:          for all suggestion in suggestions do
16:             $Score \leftarrow (\text{Prob}(\text{prev token} + \text{suggestion}) + \text{Prob}(\text{sug-}$ 
             $\text{gestion} + \text{next token}))/2$            ▷ probability from Microsoft web
            N-Gram
17:          end for
18:          Append suggestion with highest score to normalized_tweet
19:        end if
20:      end if
21:    end if
22:  end for
23:  return normalized_tweet
24: end procedure
```

the tweet, with the help of a dictionary, we check if it is an OOV word. In this example, there are two OOV words, *expndibles* and *gr8*.

Owoputi et al.(2013) applied Brown clustering on 56M tweets and generated 1000 word clusters where each cluster contains words used in similar contexts. Brown clustering algorithm operates under a simple intuition that similar words occur in similar contexts i.e. similar words have similar distributions of words to their left and right. We use these clusters in our normalization process to normalize the OOV words. First we check if the OOV word is present in the cluster or not. In our example, the word *gr8* is present in a cluster but *expndibles* is not. For the word *gr8*, we collect other words inside the same cluster which are in vocabulary and within 2 character edit distance or 1 phonetic edit distance from *gr8*. We used Levenshtein⁵ for character edit distance and Metaphone⁶ for phonetic edit distance. The choice and values of the edit distances are based on the experimental results⁷ obtained by Patra and Pechenizkiy (2014)[22] during the Web Engineering research seminar. Out of all the words that satisfy the above condition, we select the word with the highest frequency.⁸ In our example, we retrieve *great* as the normalized word.

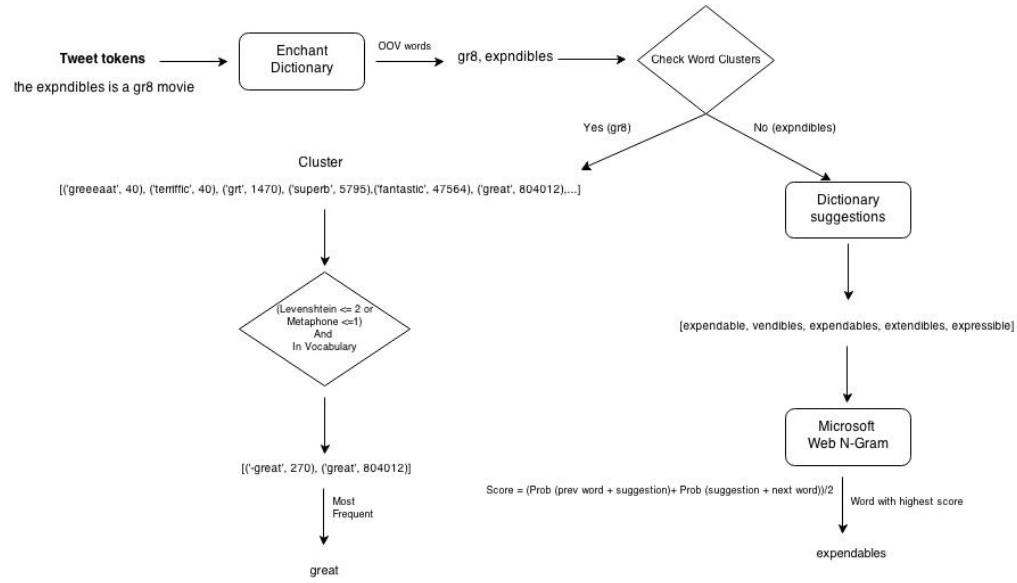


Figure 2.2: An example showing the normalization process

⁵http://en.wikipedia.org/wiki/Levenshtein_distance

⁶<http://en.wikipedia.org/wiki/Metaphone>

⁷https://github.com/srpatra88/Soumya_Thesis/blob/master/Normalization_Seminar/Normalization.pdf

⁸frequency refers to the number of times the word is mentioned in the tweet dataset in a similar context.

For the word *expndibles*, which is not in any cluster, we take help of the enchant dictionary which provides suggestions for misspelled words based on character edit distances. Out of all the suggestions, we check which one fits the most in the context. We measure the appropriateness of the suggestions with the help of Microsoft’s web n-gram service which provides conditional probabilities of words in a given context (sequence of words coming before and after the word). For each suggestion, a score is calculated with the formula given in algorithm 1. Finally, the word with the highest score is returned as the normalization output. In our example, the score for the suggestion *expendables* is calculated as below.

$$score = \frac{Prob("the expendables") + Prob("expendables is")}{2}$$

Example 2.3.1. Normalization input : *the expndibles is a gr8 movie*

Normalization output : *the expendables is a great movie*

Example 2.3.1 shows a sample input to the normalization component and the corresponding output. It is to be noted that we do not replace the normalized token with the actual token as there is still a good chance that the actual token may be part of a mention phrase. For instance, if a tweet contains a named entity *Crazy Steve* (Canadian wrestler) and if we replace normalized tokens with actual tokens, we will get the phrase *Crazy Steve* which is not a real entity. Therefore, if the actual token is part of any valid mention phrase, we do not consider normalization and leave the token as it is. However, in example 2.3.1, *expndibles* does not produce any valid mention phrase and hence we keep the normalized form of the token which is *expendables*.

We evaluate our normalization approach in section 3.5. In these experiments, we show that our normalization method is successfully able to disambiguate entities with spelling errors while other approaches fail to do so.

2.4 Mention Extraction

After the preprocessing steps, we now extract mentions (possible named entity candidates) from the tweet. We use pre-trained entity vectors with Freebase naming to extract the mentions out of the tweet. The source and the approach to generate these entity vectors is explained in detail in section 2.5. Freebase⁹ is a massive online knowledge base containing nearly 2.5 billion entities distributed over 44 million topics. The wide range of entities present in Freebase gives us the opportunity to use it in our mention extraction phase.

⁹<http://www.freebase.com/>

2.4.1 Problem Formulation

This is one of the core problem that we planned to solve in this thesis. Considering our problem statement in section 1.3, this sub-problem deals with the highlighted part below.

*Given a tweet written in English, **does it contain any named entities** and if so, which semantic entity it is referring to ?*

The goal of this task is to extract phrases out of the normalized tweet that could be possible named entities. For example, given a tweet *going to watch argo in texas*, this task is expected to extract *argo* (movie) and *texas* (USA state) as possible named entities.

2.4.2 Related Work

Traditional approaches on formal texts use supervised methods to solve the entity extraction problem. These approaches use linguistic features to first segment and then classify the named entities. Zhou and Su(2002)[25] proposed a Hidden Markov Model and a HMM-based chunk tagger to extract named entities. Their model was trained on features such capitalization, prefix type, suffix type etc. Chieu and Ng(2002)[4] used a Maximum Entropy Model trained on similar features. But as discussed before, these features are not so reliable in informal texts. Finkel et al.(2005)[9] developed Stanford NER classifier using Conditional Random Fields. Their model used capitalization, current word tag, surrounding word tags and some other features to classify entities. This classifier gives state of the art performance on formal texts. But the performance degrades significantly on tweets. Ritter et al.(2011)[23] also implemented a Conditional Random Fields model but used features that are more specific for tweets. They also used large lists of Freebase entities and their types to check if the segments are named entities. Although they achieved very good performance, they had to manually annotate a large set of tweets to train their model which takes a lot of human effort. In our experiments, we use their approach as a baseline and compare our results with theirs.

Minkov et al.(2005)[20] mention in their paper that adapting supervised approaches on named entity recognition to domains which are not as clean as newswire text can take considerable effort and knowledge of the domain. Therefore, some other approaches moved towards using freely available data on the web for the NER task. Fette(2010)[8] was able to achieve good performance by combining Google N-gram statistics with traditional methods for NER. Liu et al.(2011)[17] used Microsoft N-gram and Wikipedia for segmentation and NER. These approaches inspired us to use the Freebase entity vectors in our entity extraction and disambiguation process.

2.4.3 Our Approach

Our approach to entity extraction is quite simple but effective as suggested by the experimental studies in section 3.3.1. To find out if a phrase is a named entity or not, we match it against Freebase KB entities. We first segment the normalized tweet into partitions. The boundaries of the segments are selected by looking at the tags of the tokens. Tokens whose tags are either hashtags, user names, urls or sentence delimiters are selected as boundaries. Now, for each partition, we extract n-grams¹⁰ as potential candidates. We first collect all possible 4-grams from the partition and match against Freebase. If there is a match, we consider it as a mention and this 4-gram is not considered for further processing (i.e. we do not need to check for lower order n-grams for the tokens inside this 4-gram). The same approach is then continued for trigrams, bigrams and unigrams and all successful matches are collected as possible mentions. Due to performance considerations, we only check 4 types of n-grams and ignore higher order n-grams even though there are entities that may contain more than 4 tokens. Below, we formalize our approach in the form of an algorithm.

Mention Extraction Algorithm

```
1: procedure MATCH_NGRAMS(text, ngram_size, mentions)
2:   Initialize empty list matches
3:   ngrams  $\leftarrow$  Collect all n-grams from text with ngram_size
4:   for all ngram in ngrams do
5:     Match ngram with Freebase KB entities
6:     if match exists and ngram not in mentions then
7:       Append ngram to matches list
8:     end if
9:   end for
10:  return matches
11: end procedure
12:
13:
14: procedure EXTRACT_MENTIONS(tweet)
15:  segments  $\leftarrow$  split tweet at tokens with tags in (#, @, URL, Delimiters)
16:  Initialize empty list mentions
17:  for all segment in segments do
18:    four_grams  $\leftarrow$  Match_Ngrams(segment, 4, mentions)
19:    Append four_grams to mentions list
20:    three_grams  $\leftarrow$  Match_Ngrams(segment, 3, mentions)
21:    Append three_grams to mentions list
22:    two_grams  $\leftarrow$  Match_Ngrams(segment, 2, mentions)
```

¹⁰<http://en.wikipedia.org/wiki/N-grams>

```

23:     Append two_grams to mentions list
24:     one_grams ← Match_Ngrams(segment, l, mentions)
25:     Append one_grams to mentions list
26:   end for
27:   Append tokens tagged as proper nouns to mentions list
28:   return mentions
29: end procedure

```

The n-gram matching approach is not capable of capturing named entities which do not have Freebase entries. For example, n-gram matching will not be able to extract the mention *Joe* in a tweet *Joe is on leave today*. Therefore, we also extract tokens tagged as proper nouns as probable mentions. In our experiments in section 3.4.1, we compare the performance of the extraction process with and without including these proper nouns.

Example 2.4.1 shows a sample tweet with its segments and extracted mentions. First, the tweet is partitioned at boundaries having the above mentioned tags e.g. @IMDB, #boxoffice. Then, the combination of n-gram matching and proper noun extraction approaches extract the mentions *Harry Potter*, *Deathly Hallows* and *Friday*. In this case, *Friday* is a false positive. We realize that this approach also extracts a large number of false positives which are not actually named entities. We handle this problem in the disambiguation phase explained in section 2.5.

Example 2.4.1. Normalized input : *RT/~ @IMDb/@ :/~ As/P expected/V ./, /, Harry/∧ Potter/∧ and/& the/D Deathly/A Hallows/∧ /, leads/V the/D Friday/∧ #boxoffice/N ./, taking/V in/P an/D est/N ./, \$61.2/\$ million/N ./, http://imdb/U ./, .../∧*

Segments : '*RT*', '*As expected*', '*Harry Potter and the Deathly Hallows*', '*leads the Friday*', '*taking in an est*', '*\$61.2 million*'

Extracted mentions : '*Harry Potter*', '*Deathly Hallows*', '*Friday*'

2.5 Disambiguation

The last and the most important step in our approach is disambiguation. In this step, we remove the false positives from the extracted mentions and assign the final list of named entities with their respective reference links. For our disambiguation process, we consider the relationship between mentions as well as the context inside the tweet.

2.5.1 Problem Formulation

Considering our problem statement in section 1.3, this sub-problem deals with the highlighted part below.

Given a tweet written in English, does it contain any named entities and if so, which semantic entity it is referring to ?

Let us consider an example shown in 2.5.1. Given an input tweet and the extracted mentions, the disambiguation task is first expected to check if the mentions are actually named entities. If they are not, they are removed as false positives. Finally, the actual named entities are assigned with relevant reference links. In the example, *argo* is referring to the movie [http://en.wikipedia.org/wiki/Argo_\(2012_film\)](http://en.wikipedia.org/wiki/Argo_(2012_film)) and *texas* is referring to US state <http://en.wikipedia.org/wiki/Texas>. However, this task is not so easy. There could be a number of potential candidate entities for each mention. In the above example, *texas* could refer to *Texas Rangers (base ball team)*, *Texas Instruments (organization)* or *Texas tech university* all of which have wikipedia entries. The disambiguation component should be smart enough to choose the correct entity out of all these possible ambiguous entities.

Example 2.5.1. Tweet : *going to watch argo in texas*

Extracted mentions : '*argo*', '*texas*'

Expected output : *going to watch argo* ([http://en.wikipedia.org/wiki/Argo_\(2012_film\)](http://en.wikipedia.org/wiki/Argo_(2012_film)))
in texas (<http://en.wikipedia.org/wiki/Texas>)

2.5.2 Related Work

NED is a well researched topic in formal documents. Almost all the known approaches make use of some knowledge base to match the mentions to their appropriate entities. Some of the popular knowledge bases are Wikipedia, yago, DBpedia, Freebase. Kulkarni et al.(2009)[15] suggested to use the coherence between entities as a measure to disambiguate them. Hoffart et al.(2011)[13] used three different measures for disambiguation: the prior probability of the candidate entity for the mention, context similarity between mention and candidate and coherence between candidate entities. AIDA system¹¹ is an online tool for entity disambiguation built on Hoffart et al.(2011)[13] approach.

Among other approaches, Cucerzan(2007)[5] used heuristic rules and Wikipedia disambiguation markup to map mentions with the actual entities. Bunescu and

¹¹<https://gate.d5.mpi-inf.mpg.de/webaida/>

Pasca(2006)[3] used cosine similarity measures for disambiguation. For each candidate entity, they measured the similarity between the word vectors of the mention context and the candidate Wikipedia article. But their approach suffered on short and incomplete articles.

There has also been some work done for disambiguation in informal texts such as tweets. Davis et al.(2012)[6] proposed a three step supervised method that continuously monitors twitter streams and predicts if the mention in a tweet refers to a predefined entity. They first collected tweets containing particular mentions and applied manually developed filtering rules to separate positive examples where these mentions actually referred to the predefined set of entities. In the second step, they trained a model with these positive and negative examples to build a classifier. Finally, they used this classifier to process real time tweet streams and predict if the mention in the stream is actually referring to an entity. The disadvantage of this approach is that it needs significant manual effort to come up with filtering rules which can vary according to entities and mention contexts. Also, it is only suitable for specific domains and could not be applied to disambiguate tweets containing unknown entities.

Habib and van Keulen(2013)[11] proposed an unsupervised approach for disambiguation in tweets. They collect possible mentions of named entities by matching n-grams against Yago KB¹². Then they extract all possible candidate entities for each mention from the knowledge base. In the next step, they find all possible permutations of entities. Each permutation includes one candidate entity per mention. By performing agglomerative clustering, they obtain a set of clusters of related entities. Finally, the cluster with minimum size is chosen as the winning cluster and the related entities in this cluster are considered as final. In their approach, they consider two entities as related if there is a direct link between both the entities in Wikipedia. We believe that presence of link is not an efficient relationship measure between entities. Because related entities do not necessarily have links to each other. For instance, *PSV Eindhoven* and *FC Arsenal* are both football clubs and quite similar to one another: however, there are no links between these two entities in Wikipedia. Therefore, we use cosine similarity as a measure of relationship between entities. These similarities are derived from the Freebase entity vector model explained in the next section.

2.5.3 Our Approach

We propose an algorithm for disambiguation process which makes use of intra-tweet context (relationship between the mentions inside a tweet) and the surrounding context of the mentions. The algorithm is presented below.

¹²<https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/>

Disambiguation Algorithm

```

1: procedure COMPARE_MENTIONS(m1, m2)
2:   candidates_m1  $\leftarrow$  check_Freebase(m1) ▷ Freebase API
3:   candidates_m2  $\leftarrow$  check_Freebase(m2) ▷ get list of candidates
4:   max_similarity  $\leftarrow$  0.0
5:   for all c1 in candidates_m1 do
6:     for all c2 in candidates_m2 do
7:       score  $\leftarrow$  Freebase_model.similarity(c1,c2) ▷ cosine similarity score
       from entity vector model
8:       if score  $\geq$  max_similarity then
9:         best  $\leftarrow$  (m1,c1,m2,c2,score)
10:      end if
11:    end for
12:  end for
13:  return best
14: end procedure
15:
16:
17: procedure DISAMBIGUATE(tweet, mentions)
18:   Initialize empty list final_mentions
19:   for all mention in mentions do
20:     if proper noun in mention then
21:       Append mention to final_mentions
22:     else
23:       score1  $\leftarrow$  Prob (mention) ▷ probability from Microsoft web
       N-Gram
24:       score2  $\leftarrow$  (Prob (prev token + mention) + Prob (mention + next
       token))/2 ▷ conditional probability of mention with surrounding context
25:       if score1  $\geq$  score2 then
26:         Append mention to final_mentions
27:       end if
28:     end if
29:   end for
30:   Initialize empty dictionary final_mapping
31:   if length(final_mentions) = 1 then ▷ only one mention
32:     final_mapping(mention)  $\leftarrow$  best_candidate(check_Freebase(m1)) ▷
     Freebase API
33:   return final_mapping
34: end if
35:   Initialize empty list best_pair_list
36:   combos  $\leftarrow$  combinations(final_mentions,2)
37:   for all (m1,m2) in combos do
38:     best  $\leftarrow$  compare_mentions(m1,m2)

```

```

39:     Append best to best_pair_list
40:   end for
41:   Sort best_pair_list in the descending order of score
42:   for all best_pairs in best_pair_list do
43:     if  $score \geq 0.35$  then
44:        $final\_mapping(m1) \leftarrow c1$            ▷ assumes m1 not already in
final_mapping
45:        $final\_mapping(m2) \leftarrow c2$            ▷ assumes m2 not already in
final_mapping
46:       else if  $score \geq 0.20$  and m1 in final_mapping then
47:          $final\_mapping(m2) \leftarrow c2$ 
48:       else if  $score \geq 0.20$  and m2 in final_mapping then
49:          $final\_mapping(m1) \leftarrow c1$ 
50:       end if
51:     end for
52:     for all mention in final_mentions and not in final_mapping do
53:       if proper noun in mention then
54:          $final\_mapping(mention) \leftarrow \text{best\_candidate}(\text{check\_Freebase}(mention))$ 
55:       end if
56:     end for
57:   return final_mapping
58: end procedure

```

To measure the relationship among various entities, we use pre-trained entity vectors with Freebase naming. The model was generated with the help of a tool called *word2vec*¹³. The tool is implemented based on the continuous bag-of-words and continuous skip-gram architectures proposed by Mikolov et al.(2013)[18]. Given a large input corpus with billions of words, this tool can learn distributed vector representation of words at quick time. The resulting word vectors also provide a way to calculate similarities between words as a measure of cosine distances. For example, given a query *france*, we can get a list of words which are closest to *france* along with their cosine similarities. In the output, we get *spain* with similarity 0.678515, *belgium* with similarity 0.665923, *netherlands* with similarity 0.652428 and so on. From this example, we can see how accurately the model is able to capture semantic relationships between words, which in this example is the list of European countries. In their paper, Mikolov et al. claim that their approach outperforms other state of the art approaches and produces high quality word vectors. They further improve their model in [19] to generate distributed representations of phrase vectors.

The entity vector model that we use in our approach is trained on 100 billion words from various news articles and contains more than 1.4 million entity

¹³<https://code.google.com/p/word2vec/>

vectors. This model allows us to explore semantic relationships between various entities. Figure 2.4 shows some example entities along with other related entities from the model. We can see that some of the most similar entities for American singer */en/beyonce* are */en/alicia_keys* (also a singer), */en/jay-z* (her husband), */en/single_ladies* (her album). Apart from getting the most similar entities, we can also measure the similarity between any two entities in the model. To the best of our knowledge, these models have not been used for disambiguation tasks before.

<code>modell.most_similar('/en/france')</code>	<code>modell.most_similar('/en/beyonce')</code>
<pre>[('/en/paris', 0.6789076924324036), ('/en/nicolas_sarkozy', 0.6346868276596069), ('/en/belgium', 0.624652624130249), ('/en/germany', 0.6021987199783325), ('/en/marseille', 0.5810579657554626), ('/en/italy', 0.5734617114067078), ('/en/le_figaro', 0.570546567440033), ('/en/anglo-french_relations', 0.5698529481887817), ('/en/paris_region', 0.5652210116386414), ('/en/francois_fillon', 0.5643256902694702)]</pre>	<pre>[('/en/alicia_keys', 0.7990439534187317), ('/en/jay-z', 0.7822478413581848), ('/en/kanye_west', 0.7575857043266296), ('/en/usher', 0.7541432976722717), ('/en/mary_j_blige', 0.7531145811080933), ('/en/mariah_carey', 0.751306414604187), ('/en/lady_gaga', 0.750084638595581), ('/en/single_ladies', 0.7327812314033508), ('/en/ciara', 0.7310671806335449), ('/en/nicki_minaj', 0.729619026184082)]</pre>
<code>modell.most_similar('/en/microsoft')</code>	<code>modell.most_similar('/en/x-men')</code>
<pre>[('/en/steve_ballmer', 0.7532252669334412), ('/en/redmond', 0.6661102771759033), ('/en/windows_vienna', 0.6595468521118164), ('/en/google', 0.6496410369873047), ('/en/microsoft_windows', 0.6451455950737), ('/en/windows_live', 0.6442477107048035), ('/en/windows_vista', 0.6283221244812012), ('/en/windows_live_search', 0.6126171946525574), ('/en/microsoft_office_14', 0.6121448874473572), ('/en/windows_phone_7', 0.5974516868591309)]</pre>	<pre>[('/en/marvel_comics', 0.8247206211090088), ('/en/x-men', 0.8175938129425049), ('/en/fantastic_four', 0.7973655462265015), ('/en/daredevil', 0.7781320810317993), ('/en/hulk', 0.7408871650695801), ('/en/iron_man', 0.7363160252571106), ('/en/marvel_universe', 0.7320071458816528), ('/en/spider-man', 0.7236840128898621), ('/en/avengers', 0.7174135446548462), ('/en/captain_america', 0.7153397798538208)]</pre>

Figure 2.3: Examples showing most similar entities from the entity vector model

We explain our algorithm below with examples. In the first example, we explain the complete process in detail. For the following examples, we only explain the relevant steps taken for that particular tweet.

Example 1

Tweet : *When the last time you ran into Rick Ross and Drake twice in the same day at 2 diff video shoot locations . Today I did !*

Extracted mentions : *'the last time', 'Rick Ross', 'Drake'*

Disambiguated tweet : *When the last time you ran into Rick Ross (http://www.freebase.com/en/rick_ross_1977) and Drake (http://www.freebase.com/en/aubrey_graham) twice in the same day at 2 diff video shoot locations . Today I did !*

1. We consider that the mentions with proper nouns have high probability of being a named entity. So, they are first added to the final mentions list. In

this example, *Rick Ross* and *Drake* are added to the final list because they contain proper nouns.

2. We then test the mentions without proper nouns for false positives. In this case, the mention *the last time* is checked.
3. To remove false positives from the list of mentions, we use statistical language model from Microsoft web ngram services. SLM¹⁴ has been used before for word and phrase segmentation tasks. Gao et al.(2000)[10] used SLM for Chinese word segmentation. Our task is also kind of similar to the segmentation task. We consider a mention to be a false positive if it makes a more meaningful phrase when combined with surrounding tokens than when left alone by itself. This check is performed by the conditional probability score from the language model. In this case, the mention *the last time* is considered as a false positive and dropped.

$$Prob("the last time") < \frac{Prob("When the last time") + Prob("the last time you")}{2}$$

4. If the final list contains only 1 mention, we don't have much context to disambiguate. So, we check the Freebase API to find the best candidate based on its popularity entity ranking.
5. For tweets with more than 1 mention, we group all possible combinations of two. In this example, we get one tuple of 2 mentions (*Rick Ross*, *Drake*).
6. For each mention inside the tuple, we collect list of possible named entity candidates by querying Freebase API. Some possible candidates for mention *Rick Ross* are */en/rick_ross_1977* (American rapper) and */en/rick_ross* (a lecturer and consultant). Some candidates for *Drake* are */en/aubrey_graham* (Canadian rapper), */en/jessica_drake* (American actress) and */en/drake_university* (an university in Iowa, USA)

¹⁴Statistical Language Modelling

rick ross	similarity	drake
/en/rick_ross_1977	<----- 0.723967976296 ----->	/en/aubrey_graham
/en/rick_ross_1977	<----- 0.150620784802 ----->	/en/jessica_drake
/en/rick_ross_1977	<----- 0.148424026089 ----->	/en/drake_bell
/en/rick_ross	<----- 0.102069714267 ----->	/en/aubrey_graham
/en/rick_ross_1977	<----- 0.0921719056592 ----->	/en/nick_drake
/en/rick_ross_1977	<----- 0.0762504882361 ----->	/en/drake_josh
/en/rick_ross	<----- 0.0727031113613 ----->	/en/drake_bell
/en/rick_ross	<----- 0.0712550551445 ----->	/en/drake_equation
/en/rick_ross	<----- 0.0602228178405 ----->	/en/nick_drake
/en/rick_ross_1977	<----- 0.0570719395121 ----->	/en/tim_drake
/en/rick_ross_1977	<----- 0.0566679256497 ----->	/en/drake_hotel
/en/rick_ross_1977	<----- 0.0529266525251 ----->	/en/drake_equation
/en/rick_ross	<----- 0.0524698693689 ----->	/en/drake_josh
/en/rick_ross_1977	<----- 0.0446930869153 ----->	/en/francis_drake
/en/rick_ross	<----- 0.0423910164636 ----->	/en/drake_hotel
/en/rick_ross	<----- 0.0355052250272 ----->	/en/jessica_drake
/en/rick_ross	<----- 0.0310184634143 ----->	/en/tim_drake
/en/rick_ross	<----- 0.0288066414595 ----->	/en/edwin_drake
/en/rick_ross_1977	<----- 0.0287518607921 ----->	/en/david_drake
/en/rick_ross	<----- 0.0135697450604 ----->	/en/drake_passage
/en/rick_ross_1977	<----- 0.0124093290435 ----->	/en/vera_drake
/en/rick_ross	<----- 0.0106563613282 ----->	/en/drake_relays
/en/rick_ross_1977	<----- 0.0101147364224 ----->	/en/drake_relays
/en/rick_ross	<----- 0.00890155237177 ----->	/en/david_drake
/en/rick_ross_1977	<----- 0.00529202618851 ----->	/en/drake_bulldogs
/en/rick_ross	<----- 0.00340827700193 ----->	/en/vera_drake
/en/rick_ross_1977	<----- 0.00290638056411 ----->	/en/drake_passage
/en/rick_ross_1977	<----- 0.00194036986705 ----->	/en/drake_university

Figure 2.4: Sorted list of candidate similarities for mention tuple (*Rick Ross*, *Drake*)

- From the two lists obtained in the previous step, we again extract all possible combinations of entities where each tuple contains one entity from either of the lists.
- Now we measure the similarities between entities inside the tuples with the help of the Freebase entity vector model. Figure 2.4 shows a sorted list of similarity scores between various combinations of candidate entities for the mentions (*Rick Ross*, *Drake*). We extract the best score and the corresponding entities. In this case, it is (*/en/rick_ross_1977*, 0.723967976296, */en/aubrey-graham*). The score makes sense because both of the entities are American rappers.
- If the best score is more than a threshold value of 0.35, we consider that there is a strong connection between the corresponding candidates and assign the entities to their corresponding mentions as the final mapping. The choice of the threshold value was inspired by carefully experimenting with different thresholds on Mena collection dataset (statistics shown in table 3.3). Figure 2.5 displays the results of the experiments conducted with different threshold values. We achieve the best F-measure performance for threshold value 0.35.
- Since, this example had only two mentions, it does not need any further processing. We get the disambiguated version of the tweet as shown above.

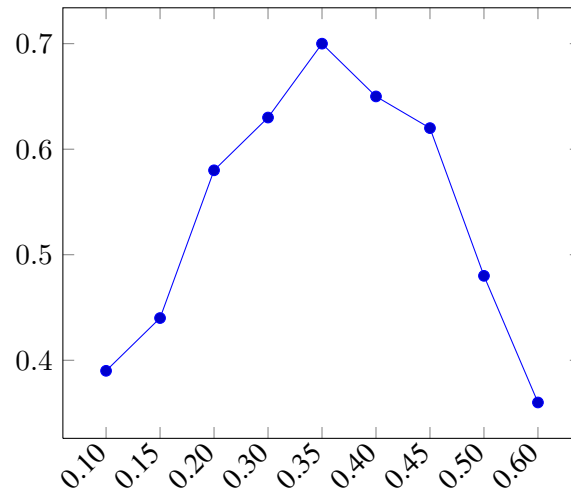


Figure 2.5: F-measure for different similarity thresholds on Ritter et al. sample dataset

Example 2

Tweet : *RT @IMDb : Catherine Hardwicke is in talks to direct ' Maze Runners ', a film adaptation of James Dashner's sci-fi trilogy . <http://imdb.to/> ...*

Extracted mentions : 'Catherine Hardwicke', 'Maze', 'James Dashner'

Disambiguated tweet : *RT @IMDb : Catherine Hardwicke (http://www.freebase.com/en/catherine_hardwicke) is in talks to direct ' Maze (http://www.freebase.com/en/the_maze_runner) Runners ', a film adaptation of James Dashner's (http://www.freebase.com/en/james_dashner) sci-fi trilogy . <http://imdb.to/> ...*

1. We proceed like the previous examples and generate all possible combinations of the mentions.
2. For each tuple, we generate the list of candidate entities and find the best two candidates corresponding to the highest similarity scores. Figure 2.6 shows the best candidates for each combination and their similarities found from the entity vector model.

```
Maze (/en/the_maze_runner) <----- 0.68020665481-----> James Dashner (/en/james_dashner)
Catherine Hardwicke (/en/catherine_hardwicke) <----- 0.328073202552-----> Maze (/en/the_maze_runner)
Catherine Hardwicke (/en/catherine_hardwicke) <----- 0.227105326887-----> James Dashner (/en/james_dashner)
```

Figure 2.6: Sorted list of candidate similarities for mentions 'Catherine Hardwicke', 'Maze', 'James Dashner'

3. Entity */en/the_maze_runner* for the mention *Maze* and entity */en/james_dashner*

for the mention *James Dashner* have the highest similarity 0.68020665481 which passes the threshold test. Hence, these two entities are assigned as final mappings for the corresponding mentions.

4. No other tuple has a similarity score more than 0.35. In this case, we make a final check to see if the remaining entities have any relationship with one of the final mapped entities. For this check, we reduce the threshold to 0.20. Again, the choice was inspired by experimenting with sample set of mentions with different values. We see that the entity */en/catherine_hardwicke* for mention *Catherine Hardwicke* is similar to */en/the_maze_runner* with a value 0.328073202552 which is more than 0.20. So we assign */en/catherine_hardwicke* as the final mapping for *Catherine Hardwicke*.

Example 3

Tweet : *RT @ebertchicago : Richard Pryor died in 2005 . " Next I'm doing Hamlet . " http://j.mp/hDKIpi!*

Extracted mentions : '*Richard Pryor*', '*Hamlet*'

Disambiguated tweet : *RT @ebertchicago : Richard Pryor (http://www.freebase.com/en/richard_pryor) died in 2005 . " Next I'm doing Hamlet (<http://www.freebase.com/en/hamlet>) . " http://j.mp/hDKIpi!*

1. Both the extracted mentions '*Richard Pryor*', '*Hamlet*' are kept as they contain proper nouns.
2. Figure 2.7 shows the sorted list of candidate similarities for mention tuple ('*Richard Pryor*', '*Hamlet*').
3. The best score is 0.256724771491 which is less than the threshold 0.35. So we disregard the similarity measures.
4. Now we consider the individual mentions. Mentions without any proper nouns are dropped as false positives. In this example, *Richard Pryor* and *Hamlet* both contain proper nouns. So, we check the Freebase API to find the best candidates based on popularity entity rankings for the mentions. Finally, we assign *Richard Pryor* with */en/richard_pryor* (American comedian) and *Hamlet* with */en/hamlet* (Shakespeare literature).

richard pryor	similarity	hamlet
/en/richard_pryor	<----- 0.256724771491 ----->	/en/hamlet_2
/en/richard_pryor_here_and_now	<----- 0.236153064963 ----->	/en/hamlet_2
/en/richard_pryor	<----- 0.21196385814 ----->	/en/hamlet_1996
/en/richard_pryor	<----- 0.209610704259 ----->	/en/hamlet_1969
/en/richard_pryor	<----- 0.209021183803 ----->	/en/hamlet_2000
/en/richard_pryor_here_and_now	<----- 0.200713603942 ----->	/en/hamlet_2000
/en/richard_pryor	<----- 0.198789336924 ----->	/en/hamlet_1948
/en/richard_pryor	<----- 0.198219829602 ----->	/en/hamlet_1964
/en/richard_pryor_here_and_now	<----- 0.190626839054 ----->	/en/hamlet_1948
/en/richard_pryor_here_and_now	<----- 0.187959518112 ----->	/en/hamlet_1996
/en/richard_pryor_here_and_now	<----- 0.186803429546 ----->	/en/hamlet_1969
/en/richard_pryor_here_and_now	<----- 0.183978766951 ----->	/en/hamlet_1964
/en/richard_pryor	<----- 0.174738037838 ----->	/en/hamlet
/en/richard_pryor_jr	<----- 0.153082444985 ----->	/en/hamlet_1969
/en/richard_pryor_here_and_now	<----- 0.148764960214 ----->	/en/hamlet
/en/richard_pryor_jr	<----- 0.147127265011 ----->	/en/hamlet_1996
/en/richard_pryor_jr	<----- 0.132136112604 ----->	/en/hamlet
/en/richard_pryor_jr	<----- 0.124752311179 ----->	/en/hamlet_1964
/en/richard_pryor	<----- 0.122718147081 ----->	/en/prince hamlet
/en/richard_pryor_jr	<----- 0.120095572612 ----->	/en/hamlet_2
/en/richard_pryor_jr	<----- 0.109961742334 ----->	/en/hamlet_2000
/en/richard_pryor_jr	<----- 0.107933226211 ----->	/en/prince hamlet

Figure 2.7: Sorted list of candidate similarities for mention tuple ('Richard Pryor', 'Hamlet')

Chapter 3

Experimental Evaluation

To measure the performance of our approach, we carry out experiments and compare our results with other baselines and competitors. Our modular implementation model also allows us to conduct experiments where we can leave out one step from the pipeline and measure its effectiveness by tracking the number of errors induced after removing that step. In all our experiments, we only experiment with tweet messages written in English. However, with very little changes, our approach could also be tested with other social media messages such as Facebook messages written in English. The approach is not suitable for experiments with messages written in other languages than English. We share our experimental data in Github¹.

3.1 Motivation

The main goal of the set of experiments is to justify our approach of using the 5 step pipeline for entity extraction and disambiguation. We show this by comparing the results of our system with other approaches with proven performances. Evaluation of the complete process is done using two different subsets of experiments, one for entity extraction performance and the other for entity disambiguation performance. We believe that our approach is not suitable for single step experiments as there are a lot of dependencies between individual steps. Therefore, we do not perform single step experiments to compare the performances of individual steps with other baselines. Instead, we justify the importance of each individual step by analysing the performance degradation of the system after removing that step from the pipeline.

3.2 Datasets

We consider three different datasets for our extraction and disambiguation experiments. For extraction experiments, we use Ritter et al.(2011) dataset. We choose

¹https://github.com/srpatra88/Soumya_Thesis

this dataset because it has a wide range of named entities and has been used in a number of NER researches. We could easily replicate their performance on the provided test dataset. For our disambiguation experiments, we use the test datasets used by Habib and van Keulen(2013). Since, their disambiguation system is not publicly available, we could not replicate their disambiguation performance on these datasets. The statistics of these two datasets are provided in table 3.3. We also explain why these two particular datasets were chosen for the experiments in section 3.3.2.

3.3 Complete Process Evaluation

With this experiment, we intend to evaluate the core components of our system. We evaluate the complete process of our approach by comparing our performance with the performance of known baselines. The evaluation process is divided into two sub-categories of experiments. In the first part, we measure the performance of our entity extraction component. Then we measure the performance of our entity disambiguation component.

3.3.1 Entity Extraction Experiments

The objective of this subset of experiments is to find out *how successfully our approach is able to extract the named entities after the complete process*. We do not consider the disambiguation performance for these experiments.

Dataset

We use the Twitter dataset provided by Ritter et al.(2011) with 2400 annotated tweets as our ground truth dataset. We also use their 1000 tweets dataset as our test set for these experiments. There is a lack of standard datasets for benchmarking in Twitter domain. This dataset was chosen for benchmarking because it has been used in most of the similar researches and also contains wide range of named entities ranging from movies, persons, organizations, album etc. There are 682 named entities present in the test dataset which means there are 0.68 named entities per tweet on average. The Ritter et al. baseline achieves an accuracy of 0.89. These statistics are shown in table 3.1.

Ritter et al. dataset	
Tweets	1000
Named Entities	682
Avg entity per tweet	0.68
Baseline Accuracy	0.89

Table 3.1: Ritter et al. test dataset statistics

Evaluation metrics

We use three metrics for our evaluation process which are precision, recall and F-measure. To calculate these measures, we first need to count the number of true positives (tp), false positives (fp) and false negatives(fn) for the test dataset.

True Positive : System extracts an entity and entity is in ground truth.

False Positive : System extracts an entity and entity is not in ground truth.

False Negative : System can not extract an entity which is in ground truth.

Let us consider the tweet shown in example 3.3.1. It shows a tweet and its actual entities (enclosed in brackets) from the ground truth dataset. Let us assume that the system predicts three entities *Alabama*, *PA* and *Newton*. In this case, *Alabama* is a *true positive* because the system is accurately able to predict it as an entity. *PA* is a *false positive*, because it is not an actual entity and is wrongly predicted by the system. *Auburn* is a *false negative* because it is an actual entity which is not predicted by the system. It is also to be noted that we consider partial matching and hence treat *Newton* as *true positive* even though the actual entity in ground truth is *Cam Newton*. An extracted entity is considered as a partial match if it is a substring of the ground truth entity and one of the boundaries (left or right) matches. Considering our overall goal (extraction as well as disambiguation), we believe full matching will be too restrictive and we might lose relevant information which can be useful for disambiguation context. Therefore, we think that partial matching is a fair measure for our task.

We do not assume that every tweet contains a named entity. Our system is expected to output an empty list for tweets without any entities. So if a tweet does not contain any entity in ground truth and our system extracts a named entity for that tweet, it will be counted as a false positive.

Example 3.3.1. Tweet (ground truth) : *RT @new_Parishotels : RT @espn : [[Alabama]] PA announcer fired for songs directed at [[Auburn]]'s [[Cam Newton]] -*

<http://es.pn/dUvP8X>

System output : *RT @new_Parishotels : RT @espn : [[Alabama]] [[PA]] announcer fired for songs directed at Auburn's Cam [[Newton]] - <http://es.pn/dUvP8X>*

For the test dataset, we calculate the number of true positives(tp), false positives(fp) and false negatives(fn). Then, we evaluate the above mentioned measures which are *precision*, *recall* and *F-measure*. We use *precision* measure to investigate what portion of our matched entities are ground truth. With the help of *recall*, we show what portion of the ground truth we cover. Finally, we take the harmonic mean of the *precision* and *recall* to calculate the *F-measure* which gives us an overall indication of the performance. Since, both *precision* and *recall* are equally important to us, *F-measure* is the most important metric in our evaluation experiments. The below mentioned formulas are used to calculate these measures.

$$Precision(P) = \frac{tp}{tp + fp}$$

$$Recall(R) = \frac{tp}{tp + fn}$$

$$F - measure(F) = \frac{2 * P * R}{P + R}$$

Experiments and Results

In table 3.2, we compare our performance with other baselines and a competitor. All the approaches were tested on the test dataset provided by Ritter et al.(2011). We consider Ritter et al. as our competitor approach because they also focus on named entity extraction from tweets. From the table, we can see that our approach performs better than Stanford NER² and MITIE³ which are state of the art information extraction tools for formal texts. This goes to show that when working on informal texts such as tweets, we can not reliably use features such as capitalization, surrounding word tags etc. We experimented with 2 different Stanford NER models, one of which uses capitalization among other features (Stanford NER) and the other one does n't (Stanford caseless). As expected, the caseless model performs slightly better than the other one.

²<http://nlp.stanford.edu/software/CRF-NER.shtml>

³<http://blog.dlib.net/2014/04/mitie-completely-free-and-state-of-art.html>

Approach	Precision	Recall	F-measure
Stanford NER	0.79	0.44	0.57
Stanford caseless	0.74	0.63	0.68
AlchemyAPI	0.82	0.51	0.63
Semantria	0.25	0.19	0.21
MITIE	0.58	0.56	0.57
Ritter et al.(2011)	0.97	0.82	0.89
Our approach	0.71	0.78	0.74

Table 3.2: Comparison of performances with other approaches on Ritter et al.(2011) test dataset

We also experiment with AlchemyAPI⁴ and Semantria⁵ who provide commercial natural language processing services. In both the cases, our approach could outperform them in extracting entities out of the test set. Our approach, however, does not provide results as good as Ritter et al. approach. While, their approach achieved a F-measure of 0.89, we achieve a score of 0.74. We manage to achieve a recall close to their approach which suggests that we cover almost as much ground truth as them. But our precision is not as impressive as theirs. This points out that their supervised model trained on various tweet specific features handles false positives better than ours. In examples 3.3.2 and 3.3.3, we show some of the typical instances where our approach extracted false positives such as '*BIG BUSINESS*' and '*Home-Going*' whereas Ritter et al. approach could successfully extract the ground truth entities. We believe our performance on precision can be improved by introducing a more advanced chunking component to filter out noun phrases before sending them for ngram matching. We plot the performance comparisons of our approach with other baselines and Ritter et al. approach. in fig. 3.1.

Example 3.3.2. Tweet : *ASPCA PETA HSUS .. Animal rescue or BIG BUSINESS ? http://tl.gd/63aeok CK OUT THIS SERIES OF LINKS BEFORE YOU DONATE*
Ground truth : '*PETA*'
Our output : '*BIG BUSINESS*', '*PETA*'
Ritter et al. output : '*PETA*'

Example 3.3.3. Tweet : *is at Home-Going Well - I take a Break-Online Saturday 4:30 pm- 6:00 pm ET- Good Day- Good Night-Good Week-End !!!*
Ground truth : *None*

⁴<http://www.alchemyapi.com/>

⁵<https://semantria.com/>

Our output : 'Home-Going'
Ritter et al. output : None

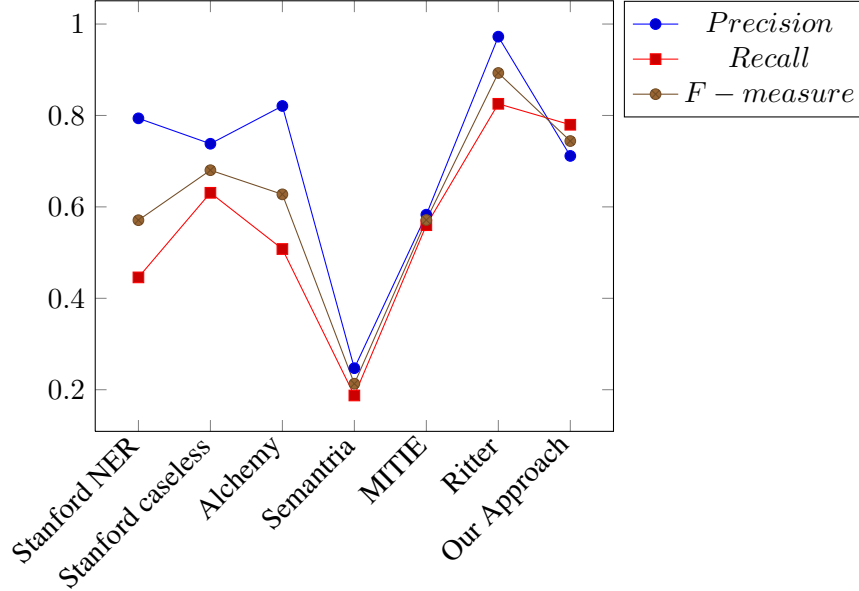


Figure 3.1: Plot showing the performance comparison of our approach with other baselines and Ritter et al.

3.3.2 Entity Disambiguation Experiments

The goal of this subset of experiments is to find out *how successfully our approach is able to disambiguate the extracted mentions and assign correct reference links to them*. We can achieve this goal by comparing our performance with some of the best known approaches in the same domain. Unfortunately, not much work has been done on disambiguating named entities in tweets. Also, most of the existing disambiguation systems are not publicly available for benchmarking. For these experiments, we benchmark our approach against two state of the art disambiguation systems AIDA⁶ and TAGME⁷. Hoffart et al.(2011)[13] proposed a disambiguation algorithm which they later implemented as an online disambiguation tool called AIDA. They benchmarked their approach against Kulkarni et al.(2009)[15] which was the best known work in NED at that time and also some other simpler approaches such as Cucerzan(2007)[5]. Hoffart et al. claim that AIDA managed to achieve better performance than the other approaches in different settings. Therefore, we use AIDA system as one of our competitors for NED experiments. The second benchmark TAGME was implemented by Ferragina and Scaiella(2010)[7].

⁶<https://gate.d5.mpi-inf.mpg.de/webaida/>

⁷<http://tagme.di.unipi.it/>

The system is designed to disambiguate long as well as short texts and hence is very interesting as a benchmark.

Datasets

For the NED experiments, we use two different annotated datasets provided by Habib and van Keulen(2013)[11] as ground truth. The first one (Mena collection) contains small set of tweets with more entities per tweet . The other one (Brian collection) contains large set of tweets with less entities per tweet. Table 3.3 shows the statistics of these datasets. From the table, we can see that Mena collection has 2.98 tweets on average per tweet. This collection allows us to evaluate the performance of our disambiguation approach on tweets that have enough context i.e. tweets where we can make use of the relationship among entities. On the other hand, Brian collection only has only 0.77 entities on average per tweet which permits us to test our approach on tweets without much context. With this second dataset, we can see how our approach performs when most of the tweets only contain one entity. Hence, these two datasets provide us the opportunity to evaluate our approach in two completely different scenarios.

	Mena collection	Brian collection
Tweets	162	1603
Mentions	510	1585
Wikipedia links	483(94%)	1233(78%)
non-Wikipedia links	19(4%)	274(17%)
no links	8(2%)	78(5%)
Avg entity per tweet	2.98	0.77

Table 3.3: Habib et al. dataset statistics

We can also see from the table that there are some entities in both the datasets which can either have no links at all or some non-Wikipedia link. As our approach can only assign Wikipedia links to mentions, we do not consider such entities during our evaluation. So, our performances on Mena and Brian collection are effectively tested on 483 and 1233 Wikipedia entities respectively.

Evaluation Metrics

The accuracy of finding the correct Wikipedia link for extracted mentions is used as the evaluation measure for our experiments. For example, if there are 100 en-

ties in the test dataset and the method is able to assign correct Wikipedia links to 80 of them, then the accuracy of the approach will be 80%. While matching Wikipedia links with the ground truth, we need to be careful about redirect pages⁸ because one entity might have more than one redirect pages. For example, article about *Inter Milan football club* can be reached by two different Wikipedia links http://en.wikipedia.org/wiki/Internazionale_Milano and http://en.wikipedia.org/wiki/Inter_Milan. For this reason, after our automated script matches the results with ground truth, we also perform a manual checking to correct errors introduced due to the redirect pages.

Experiments and Results

We benchmark our approach against TAGME and AIDA. The former system is implemented with the help of a heavy preprocessing step where a large index is built from 2.7M Wikipedia articles and a link-graph with 147M edges is generated. Disambiguation is performed with the help of a complex scoring function, details of which can be found in [7]. The latter system uses a combined objective function comprising of three different measures *prior*, *similarity*, and *coherence*. We explain these measures briefly, because a basic understanding of them is needed to understand the experimental setup clearly.

The *prior* measure considers the popularity of the entity for the corresponding mention. For example, let us consider a mention *Page* and suppose that it has only 2 candidate entities *Alan Page (American footballer)* and *Larry page (Google co-founder)*. If, out of all the occurrences of *Page* in Wikipedia, 81% refer to *Larry page* and rest 19% to *Alan Page*, then the prior values are 81 and 19 respectively. This is a very good measure for tweets because often tweets lack context and it can be used as the "best guess". In our approach, we rely on Freebase which provides us a similar kind of "best guess" when there is not enough context to disambiguate the mention.

Mention-entity similarity is the second measure that they use in their approach. From the mention side, they use all the tokens in the input text (except stop words) as context. As a preprocessing step, they extract most relevant and representative key phrases for all the Wikipedia articles and corresponding anchor texts using offline data mining techniques. Now, for each mention entity pair, they measure a similarity score based on the overlap between the mention and entity context. For example, let us consider an input text *Abbasi is the composer of Third Year in 1991*. *Abbasi* is one of the extracted mentions and *Rez Abbasi (American Composer)* is one of the candidates. In this example, the candidate *Rez Abbasi* will have a higher similarity score because its category http://en.wikipedia.org/wiki/Category:American_jazz_composers matches the token

⁸<http://en.wikipedia.org/wiki/Wikipedia:Redirect>

composer of the input text.

Third and the last measure used by AIDA is the coherence between entities. This is a rather simple measure and is calculated by the number of incoming links shared by the Wikipedia articles of the entities. For example, if an input text contains two mentions *Page* and *Brin*, then the entries *Larry Page* and *Sergey Brin* (founders of Google) will have a high coherence values for the mentions because both of them have many common incoming links to their articles.

To test our datasets in the above mentioned benchmarks, we developed Python scripts using web browser automaton⁹. With the help of these scripts, we use the AIDA and TAGME websites as APIs and automatically collect disambiguation results for the tweets. We use three different settings of AIDA system for our experiments. First setting uses only one measure *prior*. Second one (*prior+sim*) uses the combination of two measures *prior* and *similarity*. Final one (*complete*) uses all the three measures.

Accuracy		
Approach	Mena collection	Brian collection
TAGME	397 (82.19%)	818 (66.34%)
AIDA prior	306 (63.25%)	435 (35.28%)
AIDA prior+sim	307 (63.56%)	463 (37.55%)
AIDA complete	296 (61.28%)	398 (32.28%)
Our approach	361 (74.74%)	753 (61.07%)

Table 3.4: Disambiguation performances of our approach benchmarked against TAGME and different settings of AIDA

All the results for disambiguation experiments are shown in table 3.4. We also plot the results in fig 3.2. The numbers in the table represent the percentage of mentions correctly disambiguated out of the complete set of mentions in the test datasets. As shown in table 3.3, total number of mentions in Mena collection is 483 and in Brian collection is 1233. We only consider mentions with Wikipedia entities. It can be seen from table 3.4 that our approach outperforms AIDA system in all the settings. We achieve an accuracy of 74.74% on Mena collection and 61.07% on Brian collection. The decreased performance for Brian collection is because our approach can not find much context inside the tweet as there are less number of entities. This can be improved by introducing new features for disambiguation such

⁹<http://docs.seleniumhq.org/>

as topic modelling. TAGME system performs quite well with 82.19% on Mena collection and 66.34% on Brian collection.

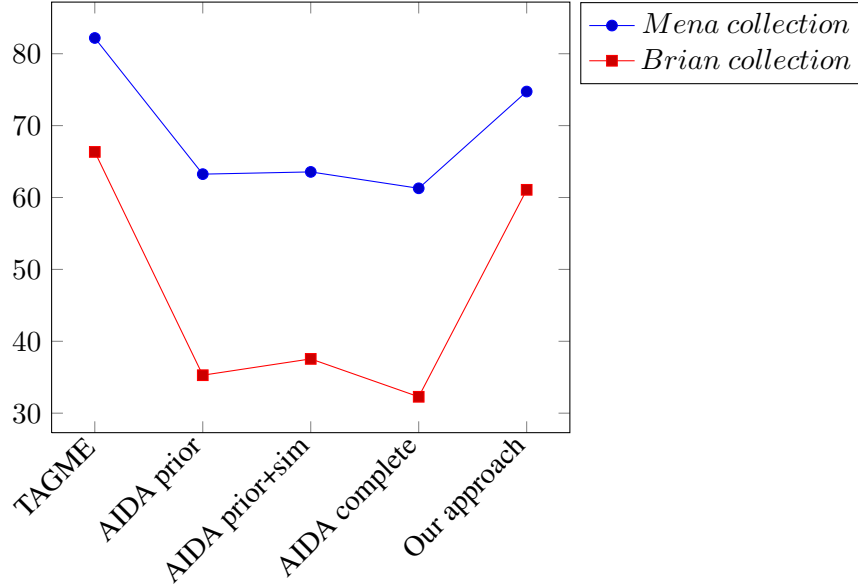


Figure 3.2: Plot showing the performance comparison of our approach with different settings of AIDA

AIDA relies on Stanford tagger to extract mentions out of the texts. We have shown from our experiments in section 3.3.1 that Stanford tagger is specifically designed for formal texts and hence is not suitable for tweets. For example, Stanford tagger primarily relies on capitalization rather than other features and is often unable to extract lower-case mentions. In example 3.3.4, we can see that AIDA is not able to extract the mention *barcelona* because of its lower case. It also wrongly disambiguates *Gerard* to *Gerard Kenny (Singer)* which actually should have been *FC Barcelona* footballer *Gerard Deulofeu*. This problem is shown in Appendix A.1. On the other hand, our approach and the TAGME system do not rely on capitalization and hence are able to disambiguate the mention *barcelona* correctly. Another observation here is that, only our approach is able to correctly disambiguate the mention *Gerard* using our the entity-entity relationship measure.

Example 3.3.4. Tweet : *Gerard scores for club barcelona yet again.*

Actual disambiguations : 'Gerard' : 'http://en.wikipedia.org/wiki/Gerard_Deulofeu', 'barcelona' : 'http://en.wikipedia.org/wiki/FC_Barcelona'

Our output : 'Gerard' : 'http://en.wikipedia.org/wiki/Gerard_Deulofeu', 'barcelona' : 'http://en.wikipedia.org/wiki/FC_Barcelona'

TAGME output : 'Gerard' : 'http://en.wikipedia.org/wiki/Gerard_Lopez', 'barcelona' : 'http://en.wikipedia.org/wiki/FC_Barcelona'

AIDA output : 'Gerard' : http://en.wikipedia.org/wiki/Gerard_Kenny'

It is not just the capitalization issue that affects the AIDA performance. Stanford NER also depends heavily on POS tag features to generate noun phrases for the entity extraction process. The grammatical errors in tweets often make these features unreliable. We show this in example 3.3.5. We can see that, even though there is no capitalization issue, AIDA is not able to extract *Spider Man 2* from the grammatically incorrect tweet. In Appendix A.2, we show how this error can be partially corrected by correcting the grammar of the tweet. Our approach is successfully able to extract such entities irrespective of the grammatical mistakes because we don't rely so heavily on POS tags. Additionally, approaches such as AIDA and TAGME are not designed to handle spelling errors which is one of the main problems in tweets. With our normalization component, we are able to handle such spelling errors to some extent.

Example 3.3.5. Tweet : *Spider Man 2 movie coming soon yipee*

Actual disambiguations : 'Spider Man 2' : 'http://en.wikipedia.org/wiki/Spider-Man_2'

Our output : 'Spider Man 2' : 'http://en.wikipedia.org/wiki/Spider-Man_2'

TAGME output : 'Spider Man 2' : 'http://en.wikipedia.org/wiki/Spider-Man_2'

AIDA output : None

We also see a significant decrease in accuracy by AIDA system for Brian collection compared to Mena collection. This suggests that, compared to other approaches, AIDA performance suffers more if the tweet contains less number of entities. Another interesting observation from the results is that AIDA *prior* and *prior+sim* settings perform better than the complete system. We show one of the reasons for this with an example. We can see from fig. 3.3.6 that the mention *England* is disambiguated correctly in the first two settings. However, in the complete setting, the disambiguation result is incorrect. We observed during the experiments that AIDA complete setting is not so good in disambiguating locations. We found a lot of instances where it could not disambiguate locations correctly. It always tries to gather some context from the rest of the text. Using this context, it wrongly disambiguates the location mention to some concept related to that location (in this case it uses the token *sports* to disambiguate *England* as *2007 English cricket season*). This also explains why AIDA complete system suffers for tweets containing less entities. We also present the screen shots for this example in Appendix A.3.

Example 3.3.6. Tweet : *Baseball is not a famous sports in England.*

Actual disambiguations : 'England' : '<http://en.wikipedia.org/wiki/England>'

AIDA Prior : 'England' : '<http://en.wikipedia.org/wiki/England>'

AIDA Prior+Sim : 'England' : '<http://en.wikipedia.org/wiki/England>'

AIDA Complete : 'England' : 'http://en.wikipedia.org/wiki/2007_English_cricket_season'

3.4 Evaluation of The Importance of Each Step

The goal of this subset of experiments is to *evaluate the importance of each step in the overall system*. With the help of these experiments, we intend to justify the reasoning behind the inclusion of each step in our approach. To conduct these experiments, we remove a step from the pipeline and measure the extraction performance of the rest of the system in terms of precision, recall and F-measure. We believe that measuring the errors introduced by removing a step from the pipeline can help us realize the importance of that particular step. These modified systems are benchmarked against our complete system on Ritter et al. test set results obtained in section 3.3.1. Our modular solution helps us achieve this goal with minimal changes to the overall system.

Overall results of these experiments are presented in table 3.5. We can see that F-measure drops by 55.40% when we do not include POS tagging. For normalization, we do not notice any significant performance degradation. We explain the reason for this in section 3.4.2. In the next setting, we remove the disambiguation step from the pipeline. As most of the false positives are dropped in the disambiguation phase of our approach, it was interesting to check the impact of dropping this step on the overall performance. We can see from the table that even though the recall increases by 8.86%, precision drops by 42.25% due to more number of false positives.

Approach	Precision	Recall	F-measure
Complete(T+P+N+E+D)	0.71	0.78	0.74
T+N+E+D	0.37 (↓ 47.89%)	0.30 (↓ 61.54%)	0.33 (↓ 55.40%)
T+P+E+D	0.70 (↓ 1.41%)	0.74 (↓ 5.13%)	0.72 (↓ 2.70%)
T+P+N+E	0.41 (↓ 42.25%)	0.86 (↑ 8.86%)	0.56 (↓ 24.32%)

Table 3.5: Showing the impact of removing individual steps on the overall performance (T: Tokenization, P: POS tagging, N: Normalization, E: Mention Extraction, D: Disambiguation)

3.4.1 Without POS Tagging

POS tagging is an important preprocessing step in our approach. To evaluate its impact on the overall process, we remove it from the pipeline and measure the

performance degradation of the overall system. So, after *Tokenization*, we directly perform *Normalization* on the tweets. In the complete system, the later steps depend heavily on POS tags. Hence, those steps had to be carefully modified for this particular setting. From table 3.2, it is evident that there is a significant degradation in the overall performance with this setting. Precision drops by 47.89% to 0.37 and recall drops by 61.54% to 0.30. F-measure also decreases by 55.50% compared to the complete system performance.

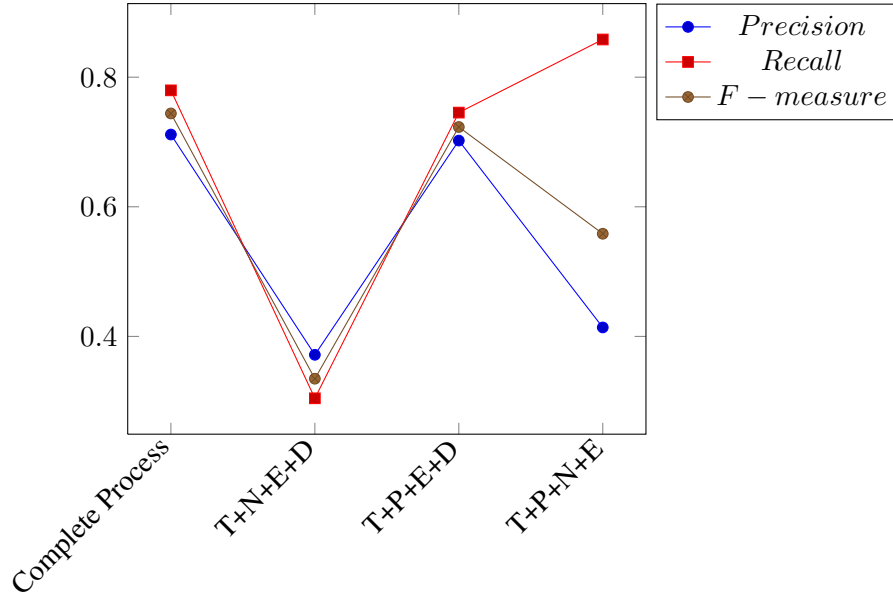


Figure 3.3: Plot Showing the impact of removing individual steps on the overall performance

We explain the reason for this degradation in performance with the help of some examples in table 3.6. In the examples, we can see that the system is now not able to discover mentions such as *yakuza* and *Lilia* which could easily be extracted in the complete system with the check of proper nouns. Also, in the complete process, we use POS tags to segment the tweets to partitions. Without the segmentation, the surrounding context of the mentions are not as much reliable to drop false positives. Due to this, large number of false positives are predicted as possible entities which degrades the precision performance.

Tweet	Missing Mention
@Jessica_Chobot did you see the yakuza vs zombies smh but cool at the same time	yakuza
@vincentsmojo You think I secretly love dresses lol Only when Lilia wears them ;)	Lilia

Table 3.6: Examples of some errors introduced due to the removal of POS tagging

3.4.2 Without Normalization

Like the previous experiment, we remove normalization from the pipeline to evaluate its importance in the overall system. In this setting, we do not normalize the tweets after POS tagging. Instead, we directly perform the step to extract the mentions out of the tweet. From table 3.5, we can see that removing normalization step from the complete process does not have a significant impact on the overall performance. F-measure drops by a mere 2.70% from 0.74 to 0.72. After carefully investigating the Ritter et al. dataset, we found that majority of the entities are already in their correct forms and do not contain any errors. Therefore, we conduct a separate experiment in section 3.5 to evaluate the importance of our normalization approach. In this experiment, we cripple the test dataset by introducing intentional spelling errors and test how successfully our system is able to handle such errors with the help of normalization. We also benchmark this experiment against AIDA disambiguation system.

3.4.3 Without Disambiguation

For this experiment, we leave out the disambiguation step from the complete process. All the mentions extracted from the previous step (mention extraction) are considered as the final set of entities. From the results shown in table 3.2, we can see that the precision drops by 42.35% from 0.71 to 0.41 which is intuitive because there are more false positives now after removing the disambiguation step. We cover a little more ground truth which can be seen from the 8.86% improvement on the recall. But, the overall performance, which we measure by F-measure, decreases from 0.74 to 0.56. Hence, with the help of this setting, we show that our disambiguation step contributes to achieve 32.14% better performance.

3.5 Evaluation of Normalization

The goal of these experiments is to justify the use of normalization step in our disambiguation system. To achieve this goal, we manually introduce spelling errors in a sample tweet dataset. Then we process this crippled dataset through our pipeline as well as through AIDA disambiguation system (complete setting). The

motivation for choosing AIDA as benchmark is explained in section 3.3.2. Like many of the other extraction and disambiguation systems, AIDA is also not able to handle spelling mistakes in entities. Using these experiments, we will show that a normalization component can be extremely useful in noisy text domains such as tweets.

For these experiments, we randomly choose 83 tweets (236 Wikipedia entities) from Mena collection as test data. First, we evaluate the disambiguation performances of both the systems (i.e. our approach and AIDA) on the clean test dataset. Then we start introducing errors. For the first experiment, we introduce a spelling error for every fifth named entity inside the tweet dataset. So, for this experiment, nearly 20% of the total named entities i.e. 51 entities contain spelling errors. For the second experiment, we double the number of errors to 99 i.e. 40% of the total number of entities contain spelling errors. For each error, we use 1 character edit distance which might be added, swapped or removed from the entity. For example, an entity *Egypt* can become *Egpt* by removing a character *y*. We can add an extra character *e* and make it *Eegypt*. We can also swap *y* with *i* and make it *Egipt*. All the edit characters were chosen at random and without any particular order.

	Our Approach	AIDA
No. of tweets	83	83
No. of entities	236	236
No error	178(75.4%)	137(58.0%)
20% error	167(70.7%)	116(49.1%)
40% error	157(66.5%)	94(39.8%)

Table 3.7: Normalization Experiments

Table 3.7 shows the results obtained in these experiments. As described in section 3.3.2, the accuracy of finding the correct Wikipedia link for the extracted mentions is used as the evaluation measure. The ground truth dataset contains 247 entities in total out of which 236 are Wikipedia entities. As we only consider Wikipedia entities for our experiments, all our accuracy measures are also calculated from these 236 Wikipedia entities. We can see from the table that our approach achieves 75.4% accuracy on the clean and error free test data as compared to 58.0% in AIDA. As we start introducing errors, there is a high degradation in performance of AIDA system. By introducing 20% error in the test data, accuracy of AIDA drops to 49.1%. When we double the errors to 40%, accuracy further drops down to 39.8%. On the other hand, our approach handles the spelling errors quite well and the performance degradation is not as significant as AIDA.

To visualize the above mentioned observations more clearly, we also provide a plot in fig. 3.4. From the figure, we can notice that even with 40% errors, our system performs better than AIDA system with no errors. This emphasizes the importance of having a normalization component in the entity extraction and disambiguation applications, especially in noisy domains such as tweets. Using these experiments, we have shown that our normalization component does a fair job in extracting and disambiguating misspelled named entities.

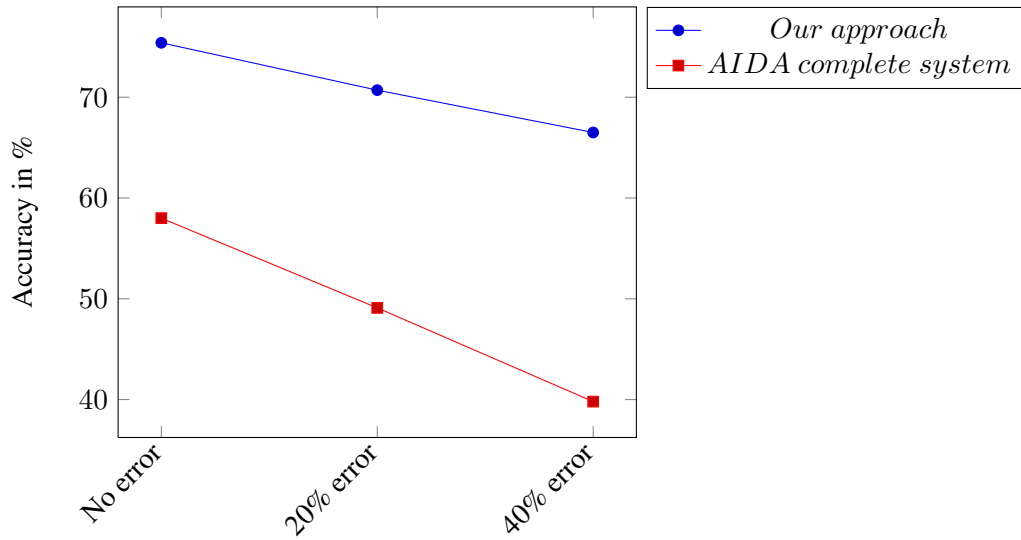


Figure 3.4: Plot showing the effect of introducing spelling errors in our approach and AIDA complete system

Chapter 4

Case Study : Filtering Targeted Tweets with Disambiguation

In this case study, we show an example of how our system can be applied in a real world scenario. For this study, we collect random tweets tracking an ambiguous term. We select one of the candidate entities as the target and see if our system can filter the tweets about the selected entity. These filtered tweets can later be used for different analysis techniques. Let us consider an example. Suppose we want to perform sentiment analysis to know what people are saying about *Jennifer Aniston*. We track the Twitter API with the term *jennifer* and collect some tweets. Now, we process these collected tweets through our pipeline with a condition specifying that we only want tweets about *Jennifer Aniston*. Finally, with the accuracy measure, we evaluate how effectively our system is able to perform this task. For the evaluation, we also consider the accuracy of correct rejection along with the accuracy of correct prediction. Because, in this task, it is equally important to reject entities that are not referring to the targeted entity.

In table 4.1, we show the results of the case study. We consider 4 different ambiguous terms referring to different people. Using Twitter API, we collect 100 tweets for each case. The column *Entity* displays the targeted entities which we are trying to extract. For each entity, we also show the number of tweets which are actually referring to that entity. This was done manually by looking at the context in which the term is mentioned inside the tweet. The column *predicted tweets* represents the prediction of our system. For example, after processing 100 tweets on the term *robert*, our system predicts that 19 of them are referring to the entity *Robert Pattinson* and outputs these tweets with their disambiguations. From the table, we can see that in most of the cases our system is able to accurately extract all the tweets talking about the targeted entity. Only for the entity *Cristiano Ronaldo*, we get a comparatively lower accuracy of 79%. Our system also performs quite well in rejecting the tweets that are not talking about the targeted entity. In three out of the 4 cases, we get a rejection accuracy of 100% i.e. all the tweets

which are rejected are indeed not referring to the targeted entity.

Term	Entity	Actual tweets	Predicted tweets	Correctly pre-dicted	Correctly re-jected
jennifer	Jennifer Aniston	24	24	100%	100%
chris	Chris Brown	61	67	100%	85%
robert	Robert Pat-tinson	21	19	90%	100%
ronaldo	Cristiano Ronaldo	58	46	79%	100%

Table 4.1: Showing the performance of our system in correctly filtering a targeted entity out of ambiguous entities

We highlight some of the representative examples from the case study in table 4.2. We can see that the system is able to accurately predict the entity to be *Jennifer Aniston* irrespective of the spelling error in *Anniston*. Using our disambiguation approach, we are also able to correctly predict that the tweet containing mention *CR7* is referring to *Cristiano Ronaldo*. Even though we have only given examples of people in this case study, this approach can also be used to filter tweets in other domains such as products, movies, location, organizations.

Tweet	Predicted Entity
Jennifer as rachel was my favourite	Jennifer Aniston
RT @WomanCrushPixs : Jennifer Anniston is perfection	Jennifer Aniston
@bbaptism i want chris 2 get chubby again its unfair	Chris Brown
RT @totalfilm : We're excited to announce a Google Hangout with The Rover stars Robert Pattinson & Guy Pearce : http://t.co/kFt0e2O5fg	Robert Pattinson
CR7 was honored to receive the award Goal 50 (best player of the season) awarded by Goal magazine http://t.co/9cBtHi3kqV #Ronaldo	Cristiano Ronaldo

Table 4.2: Some representative examples from the case study

Chapter 5

Conclusion and Future Work

In this thesis, we have studied the problem of named entity recognition and disambiguation which have always been considered as challenging tasks. Performing this research in the noisy domain of tweets makes it even more difficult. As a solution to the problem, we have proposed a generic system which is able to extract and disambiguate entities belonging to wide range of categories. With the help of our experiments, we have also shown that our approach outperforms most of the other baselines and competitors in both entity extraction and disambiguation. The proposed system is able to disambiguate entities irrespective of spelling and grammatical errors. This is made possible by our normalization algorithm. Through our normalization experiments, we stress the need of normalization in tweets and show how spelling errors can cripple the performance significantly if we don't have a normalization component. Finally, we have also presented a case study indicating how our system can be applied to filter targeted tweets for various analysis purposes.

Missing capitalization, abbreviation, shortened texts, grammatical and spelling errors are some of the most challenging problems in tweets. In our NER experiments, we have explained how these problems can even degrade the performance of state of the art NER systems. We have also shown how our approach is able to tackle these problems quite well.

For our disambiguation component, we make use of an entity vector model generated with the help of powerful unsupervised learning techniques. With the similarity scores provided by the model, we are able to capture the relationships between various entities inside a tweet. In our approach, we use such relationships to disambiguate the entities. In future, our model can be extended by introducing a topic detection component. The role of this component would be to predict the underlying topic of a tweet. This feature can be combined with the existing entity relationship model to get better results. For example, if we can detect the topic of a tweet to be about jobs, then we can easily predict that a term *monster* is referring

to the carrier website *monster.com* rather than a cable manufacturer *Monster cables*. Apart from this, the system could also gather more context by exploring the mentioned url contents, other tweets shared by the tweeter etc.

The main practical deliverable of the work done in this thesis was to implement, study, investigate and experiment with different approaches that can extract knowledge out of the giant information source Twitter. With the help of experiments and the case study, we have shown that our system does a decent job towards fulfilling this goal. We also believe that there is still space for improvement and the proposed system can be further enhanced by considering the above mentioned improvements.

Bibliography

- [1] AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL*, 2006.
- [2] Thorsten Brants. Tnt - a statistical part-of-speech tagger. In *Proceeding of 6th Applied Natural Language Processing Conference*, 2000.
- [3] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 9–16, 2006.
- [4] Hai Leong Chieu and Hwee Tou Ng. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, 2002.
- [5] Silviu-Petru Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, page 708716, 2007.
- [6] Alexandre Davis, Walter Santos, Adriano Veloso, Wagner Meira Jr., Alberto Laender, and Altigran Soares da Silva. Named entity disambiguation in streaming data. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 815–824, 2012.
- [7] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010.
- [8] Ian Fette. Combining n-gram based statistics with traditional methods for named entity recognition. 2010.
- [9] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.
- [10] Jianfeng Gao, Hai-Feng Wang, Mingjing Li, and Kai-Fu Lee. A unified approach to statistical language modeling for chinese. In *Proceedings of 2000*

IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000.

- [11] Mena B. Habib and Maurice van Keulen. A generic open world named entity disambiguation approach for tweets. In *Proceedings of the 5th International Conference on Knowledge Discovery and Information Retrieval*, 2013.
- [12] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makn sens a #twitter. In *ACL*, 2011.
- [13] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordin, Hagen Furstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, Edinburgh, Scotland*, pages 782–792, 2011.
- [14] Max Kaufmann and Kalita. Syntactic normalization of twitter messages. In *International conference on natural language processing*, 2010.
- [15] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.
- [16] John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.
- [17] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. Recognizing named entities in tweets. In *ACL*, 2011.
- [18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, 2013.
- [19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.
- [20] Einat Minkov, Richard C. Wang, and William W. Cohen. Extracting personal names from email: applying named entity recognition to informal text. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 2005.
- [21] Olutobi Owoputi, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*, 2013.

- [22] Soumya Ranjan Patra and Mykola Pechenizkiy. Normalization of tweets. In *Web Engineering Research Seminar*, 2014.
- [23] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: An experimental study. In *EMNLP*, 2011.
- [24] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP*, 2000.
- [25] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002.

Appendices

Appendix A

AIDA Disambiguation Errors

A.1 AIDA Disambiguation Error : Capitalization

Gerard scores for club barcelona yet again.

Disambiguate

Input Type:TEXT Overall runtime:0 sec(s)

Gerard Gerard Kenny scores for club barcelona yet again.

yago
select knowledge

Gerard Kenny

✗

Gerard scores for club Barcelona yet again.


Disambiguate

Input Type:TEXT Overall runtime:0 sec(s)

Gerard Gerard Deulofeu scores for club Barcelona FC Barcelona yet again.

yago
select knowledge

FC Barcelona



Gerard Deulofeu

✓

Figure A.1: Screen shots from AIDA online tool showing how small case affects disambiguation

A.2 AIDA Disambiguation Error : Grammatical Error

Spider Man 2 movie coming soon yipee

Disambiguate

Input Type:TEXT Overall runtime:0 sec(s)

Spider Man 2 movie coming soon yipee

Run InformationGraphRemoval Steps

null

The Spider Man 2 movie is coming soon yipee

Disambiguate

Input Type:TEXT Overall runtime:0 sec(s)

The **Spider Man** **[Spider-Man]** 2 movie is coming soon yipee


 Spider-Man

Figure A.2: Screen shots from AIDA online tool showing how a grammatical error affects disambiguation

A.3 AIDA Disambiguation Error : In Complete Setting

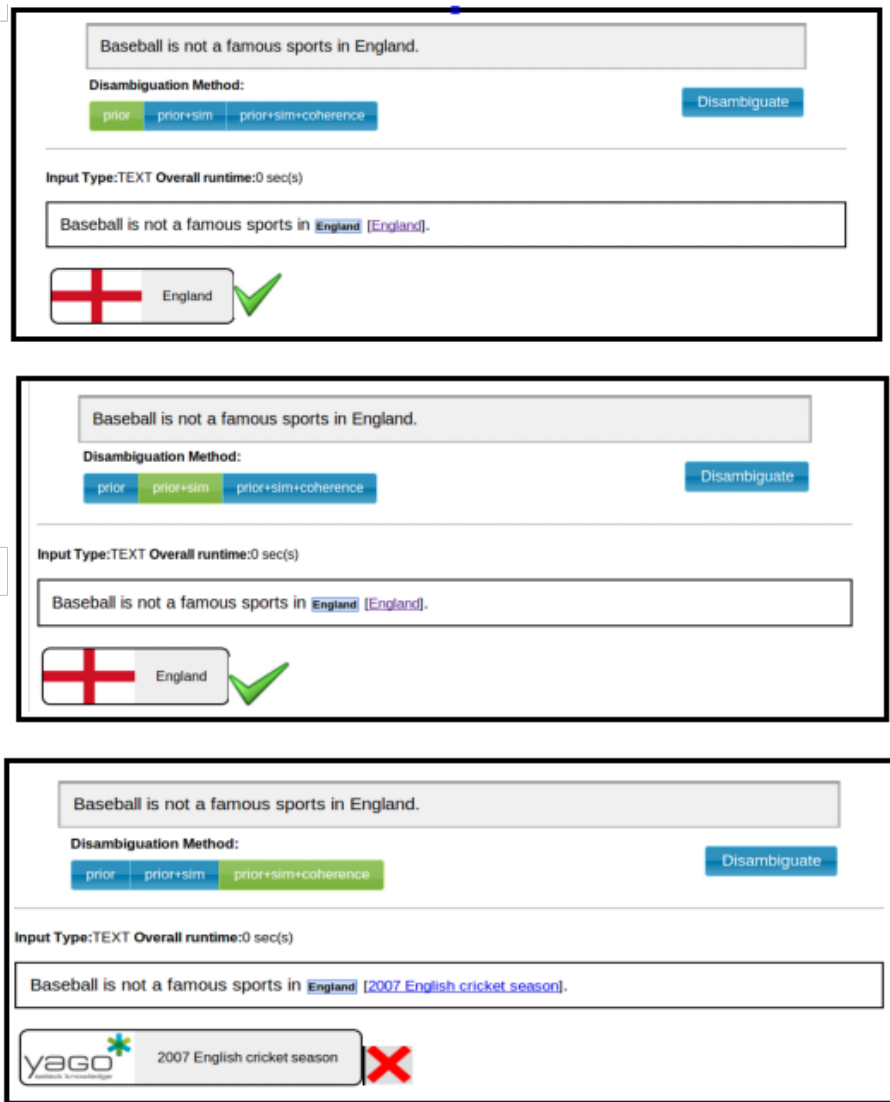


Figure A.3: Screen shots from AIDA online tool showing disambiguation error with the complete setting