Exploring the Power of Supervised Learning Methods for Company Name Disambiguation in Microblog Posts

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Nafiye Polat

in partial fulfillment for the
degree of Master of Science

in

Electronics and Computer Engineering

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Electronics and Computer Engineering.

APPROVED BY:

Assist. Prof. Ali Çakmak .....
(Thesis Advisor)

Assoc. Prof. Vural Aksakallı .....

Assist. Prof. Barış Arslan .....

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL: 29 August 2014

SEAL/SIGNATURE:

Declaration of Authorship

I, Nafiye Polat, declare that this thesis titled,''Exploring the Power of Supervised Learning Methods for Company Name Disambiguation in Microblog Posts' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: N. Polat

Date: 29 August 2014

Exploring the Power of Supervised Learning Methods for Company Name Disambiguation in Microblog Posts

Nafiye Polat

Abstract

Entity disambiguation is the task of identifying the real world entity that was referred to/mentioned in a context. Ambiguous references to entities may occur due to variations of how an entity is referenced (BT, British Telecom) or inherent ambiguities of the names used for entities (Orange Telecom vs. fruit orange), and misspellings (Best Buy vs. BestBuy). Ambiguities in company names however come with a price, when it comes to finding information about the company on the Web. Recently, tracking social media for brand management has become a very important part of the process in marketing, public relations, and product marketing. Therefore, resolving references to real world objects has become an important part of social media analytics systems. In this thesis, we study different machine learning algorithms for entity disambiguation in micro-blogging posts. We show that with the carefully selected set of features, supervised learning techniques would improve the disambiguation quality significantly.

Keywords: Information Retrieval, Text Mining, Machine Learning, Natural Language Processing, Data Mining, Online Reputation Management, Twitter, Social Media, Name Ambiguity

Tweet Metinlerinde Şirket İsimleri Belirsizlik Problemini Öğretici ile Öğrenme Yöntemlerinin Gücü ile Çözme

Nafiye Polat

Öz

Varlığın belirsizliğini giderme, varlığın içerik içerisinde asıl kastettiği varlığı bulma işlevidir. Varlığın belirsizlik problemi çeşitli nedenlerden dolayı meydana gelebilir. Örneğin, bu problem varlığın referans verilme çeşitliliğinden kaynaklanabilir. Ya da varlık için kullanılan kelimelerin belirsizliğinden kaynaklanabilir. Son olarak, bu belirsizlik hatalı yazımlardan kaynaklanabilir. Şirket isimlerindeki belirsizlikler, şirket hakkında Web üzerinde bilgi araması yapılması söz konusu olduğunda önemli olabilir. Son zamanlarda, marka yönetimi için sosyal medyanın takip edilmesi pazarlama, yerel ilişkiler ve ürün pazarlamasında şirket hakkında adımların atılması noktasında önemli olmaktadır. Bu çalışmada Tweet metinleri üzerinde belirsizliği giderme problemi için farklı makine dili öğrenimi algoritmaları uyguladık. Çalışmamızda dikkatli seçilen özellik setleri ile, öğretici ile öğrenme tekniklerinin belirsizlik probleminin çözümünde katkı sağladığını gösterdik.

Anahtar Sözcükler: Bilgi Geri Alımı, Metin Madenciliği, Veri Madenciliği, Makine Dili Öğrenimi, Varlığın Belirsizlik Problemi

I dedicate this work and give special thanks to my mother who is my best friend, and she has always supported me throughout my thesis process.

Acknowledgments

I would like to express the deepest appreciation to my thesis co-advisor, Doctor Rabia Turan. Without her guidance and persistent help, this thesis would not have been possible. Also, I would like to thank my thesis advisor, Ali Çakmak for his guidance, discussion and feedback through my thesis process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Ambiguities in company names are very common. A company name may refer to anything. This is meaningful; because, companies intentionally prefer ambiguous brand names in order to improve their marketing and branding strategy. For instance Apple and Blackberry are both company names as well as fruit names. Similarly, Chanel is a person name and a cosmetics company name. Avon may refer to another cosmetics company or a town in Ohio. This situation leads to new challenges, when it comes to finding information about a company on the Web. Companies track Twitter and other microblogging/blogging sites for brand awareness. Twitter [1] has rapidly gained worldwide popularity, with 500 million users since 2012, generating over 340 million tweets daily and keeping over 1.6 billion search queries per day [2]. Users can share short messages (tweets) on any subject. Usually, users share their good and bad experiences with company and products so that the people in their networks are aware of the brand and its shortcomings or excellence. Thus, analysing such messages can help exploring the important social circumstances. Hence, identifying the tweets referring to products and companies is an important tool to manage brand awareness on social media.

Online Reputation Management, Social Media Monitoring, and Opinion Mining from social media are some research areas that focus on users' views on social media on different types of entities. These tasks are challenging as the company and product names are often ambiguous. For example, the company Apple Inc. shares its name with the fruit apple which again could have a number of different meanings depending on the context, for example, information about the story of Adam, Eve and the serpent. If the content analysis that companies use are not able to disambiguate between the content related to the company versus the content related to the namesakes, the data they are

tracking will be erroneous and irrelevant. Therefore, analysts spend a huge amount of time to identify to filter irrelevant social media content.

In this study, we focus on finding relevant tweets to a company in the context of the WePS-3 data set [3], where we are given a set of companies, and for each company, a set of tweets which may or may not be related to the company. The tweets contain the company name, as a keyword. Firstly, it is beneficial to explore the research challenges to better understanding the problem.

1.2 Research Challenges

Limited Content: Tweet messages are very short, that have 140 characters at maximum, so they contain very little information. From this aspect, Twitter differs from the traditional user media. For example, Facebook users do not have any space scarcity problem. They share their feelings and opinions with the rich text format. Therefore, analysing such content to obtain company relevant information is easier in comparison to Twitter data. Analysing tweet data for tweet classification problem is a very difficult task. For instance:

SmebdY cMe keeP mE coMPanYY !! CaNt sLeePP !!!

I just became the mayor of dunkin donuts in Crystal City on @foursquare!http://4sq.com/cksPC1

Assume that these tweets are considered for whether they are related to CME Inc. and Dunkin Donats Inc., it is very challenging to classify the first tweet as 'false'and the second tweet as 'true'. When we remove company keywords from tweet content (In this study, we remove company keyword names from tweet text in order to prevent bias), the remaining parts are mostly meaningless and insufficient to solve tweet classification problem.

Due to the content limitation, it is almost impossible to solve company disambiguation problem using only tweet content. This forces us to use external resources as company profiles implicitly.

Misspellings and weak content: Tweet messages use a specific language, often with incorrect grammar and specific abbreviations that make analysing the tweet content difficult. In other words, since tweet content is informal, Twitter users generate their

content without doing spell checking that leads to erroneous content. Also, like other social websites, Twitter is used for the purpose of sharing opinions. Since users want to share their ideas instantly, they use some abbreviations and symbols that are very difficult to interpret for Tweet analyser. Below example show some misspellings of the given tweet:

@gabeyskanker your tellin mehh i dnt evn noe what this hype is about im only gunna cme on dis agen wen fb is totallyy DEAD :| just like bebo

Moreover, tweets are being generated in conversational mode, the quality of the content information is far away from revealing satisfying information about tweet relevancy or irrelevancy regarding a given company. Below examples show the weak content of tweets. Both of the sample tweets are relevant with to the mentioned companies. However, constructing such a classifier is a challenging task in order to identify those tweets as related.

I told you! (@ Dunkin' Donuts) http://4sq.com/6gy4zT

@BigBuilder @Lennar You're welcome. Stay well.

As we mention in the next sections, as a result of the weak and mistaken content, we could not obtain convenient results when we use more user based categorical features. Also, we face with the same situation when we use company review profiles.

Inconsistency between Training and Testing Data:

In our data set [3], we have trial, training, and testing companies each have non-overlapping organization names. In other words, we have different organizations in trial, training, and testing categories. Therefore, the characteristics of data in each group is different from others.

In [4], the relatedness factor is explained as one of the company classification technique. The relatedness factor is defined as:

$$relatedness = \left(\frac{\text{Number of tweets in Set} \in Company}{\text{Number of tweets in the Set}} \right) \tag{1.1}$$

The relatedness factor is significant due to demonstrate the characteristic properties of our training and testing data. When we apply (1.1) for our training (including Trial

Data set) and testing data, the obtained results are represented in Table 1.1 where T represents Tweets, and R Value represents relatedness value.

TABLE 1.1: Relatedness Values for Training and Testing Data

Data Set	**# of True T**	**# of False T**	**# of Unknown T**	**R Value**
Training	9884	13248	523	0.414
Testing	7717	13076	2299	0.371

As you see from Table 1.1, the relevancy of training tweets is greater than testing tweets. That means, our training data mostly includes relevant keywords about the corresponding company in comparison to testing data set. Therefore, the extracted features from training data might not be compatible with the test data. In the Experiments section, we show that sophisticated capable classifiers which consider on multiple features do not provide good results as we expect. We conclude that one of the reason might be different characteristics of the training and testing data. This also may cause overfitting problem, since the learned patterns may not have been suitable for testing data.

1.3 Our Approach

In order to overcome the problem that tweets contain little information, we use external resources to enrich the information for an organization. More specifically, we generate *several profiles* for each company which contains richer information. For each company, we construct twelve different profiles automatically. The first seven profiles have essentially sets of keywords, which are related to the company in some way. On the other hand, the remaining profiles explicitly contains unrelated keywords. The names of the related profiles are 'Company Home Page Profile','Company Wikipedia Page Profile', 'Company Review Page Profile','Company Wikipedia Page Kullback-Leibler Profile', 'Company Wikipedia Page Noun Phrase Profile', 'Company Wikipedia Page Term Frequency Profile','Company Wikipedia Page Latent Semantic Indexing Profile'. On the other hand, 'Company Wikipedia Disambiguation Page Profile', 'Company Wikipedia Disambiguation Page Kullback-Leibler Profile', 'Company Wikipedia Disambiguation Page Noun Phrase Profile', 'Company Wikipedia Disambiguation Page Term Frequency Profile', and 'Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile'contains company irrelevant keywords. In order to analyse the influence of keyword set size, we construct those profile vectors as including 100, 250 and 500 different sets of keyword.

Then, we perform feature extraction to obtain both numerical and categorical features. Our numerical features include several similarity measures and the number of different meanings of the company name feature which shows the intensity of the company name ambiguity. As similarity features, we compute Wikipedia cosine similarity, Wikipedia Disambiguation cosine similarity, Review cosine similarity, Wikipedia Kullback-Leibler Divergence cosine similarity, Wikipedia Term Frequency cosine similarity, Wikipedia Latent Semantic Indexing cosine similarity, and Kullback-Leibler [5] Divergence asymmetric similarity. For cosine similarity features [6], we compute similarity between each tweet vector of a certain organization and the corresponding organization profile vector. Also, in order to compute Kullback-Leibler Divergence distance similarity, we use asymmetric distance approach between each of the organization tweet and that organization. As categorical features, we extract some user based features from tweet itself such as unigram, url. Lastly, we generate our feature vector by combining all of above features.

Then, we employ several supervised classifiers for previously unseen companies, by training the features of the classifier. We build those classifiers both on several combinations of the extracted features and all features. With all features, we get the best accuracy using Majority Voting classifier[7] that includes LADTree[8], BFTree [9] and Multilayer Perceptron[10] as classifiers. Then we perform feature extraction using Attribute Selected Classifier[11] that chooses most important features for classification task. Using only those selected features, we conduct some additional experiments.

Then we employ Threshold Algorithm and Simple Approach Algorithm as alternative methods. In Threshold Learning approach, the similarity threshold between a tweet and a company profile is learned from historical data. This learned threshold value is then used in classification of unseen tweets as related or unrelated. Surprisingly, the algorithm produces better results, although the approach is simpler. We do Threshold experiment for Company Wikipedia Page Profile, Company Wikipedia and Wikipedia Disambiguation Page Profile (Two Profile Approach), Company Review Page Profile, Company Wikipedia Page Noun Phrase Profile, Company Wikipedia Page Kullback-Leibler Profile, Company Wikipedia Page Term Frequency Profile, and Company Wikipedia Page Latent Semantic Indexing Profile with different sets of keyword. For Threshold Approach, we obtain the best result using Company Wikipedia Page Profile that includes 100 keyword set. Also, this result outperform other classifiers that we tested. The Threshold approach improves the accuracy by approximately 11% over our baseline algorithm. (We initially employ Baseline Algorithm using weighted bag of keywords from Twitter. The approach produces 59,7% accuracy.) Since, we get our data set from WePS-3 competition data set that was held in 2010, we have a chance to compare our results with other

competitors' results. Among all participants, we get the third best result in the WePS-3 Evaluation System. As compared our systems with WePS-3 participant system, our system's performance (71%) is less than the system of Yerva (83%) and Yoshida (75%).

Also, Simple Approach Algorithm produces more accurate results as similar to Threshold Experiment did. In this experiment we follow simple approach. For each company, top 100 key words that have the highest tf-idf values are obtained. Then, we make an assumption that if a certain tweet includes one of the corresponding company profile vector term, the tweet would be related with the company; otherwise it would be unrelated. As we get the best accuracy with Company Wikipedia Page Profile that includes top 100 keywords for Threshold Learning approach, we employ Simple Approach Algorithm only using Company Wikipedia Page Profile.

Lastly, we employ Entity Ranking algorithm [12] that uses two language techniques named Entity Mention Language and Review Language respectively. As Entity Mention Language, we use two profiles, one is company related profile and the other is company unrelated profile. For each company, we generate company page profile, noun phrase profile, term frequency profile, Kullback-Leibler profile and Latent Semantic Indexing profile using both company Wikipedia page and company Wikipedia disambiguation pages with different size of keyword sets. As Review Language Model, we use all review corpus obtained from company training and testing review profiles. We obtain the best performance with Company Latent Semantic Indexing profile. The experimental results show the significant improvement of accuracy (9.5%) over our baseline approach.

1.4 Comparison with Other Systems

Our task mainly falls under Information Retrieval whose goal is to match unstructured short text (query) against unstructured document text. However, our problem is not suitable to directly employ standard Information Retrieval models such as tf.idf. The standard tf.idf scheme assumes that the query is short and the document is long. In our problem, we have a company profile vector (long) including different sets of keyword and a tweet message (short) for that company. If we consider each tweet as a query and a profile vector as a document, we need to rank and find the best query for a given document. However, we are interested in finding relevant tweets for a given company rather than finding the best tweet. We use tf.idf technique for the purpose of assigning a weight for each word, which represents the importance of that word in the company profile. Also, while assigning weight to the tweet profile keywords, we consider each tweet as a document, and each word in the tweet document as a query, then we compute the weight of the word for each tweet document.

Our problem is also different from entity matching problem that aims to find the compatibility between two structured objects. Our problem is harder than entity matching problem, because the entity matching problem seeks an attribute correspondence between two structured objects. However, there is no any attribute correspondence between a tweet and a company profile. For example, there is no company attribute such as the location of a company or founder of a company that can be always extracted from all tweet content or company profile content. In contrast, as mention in above examples in the Introduction section, tweet content is quite unstructured and far away from having well-defined attributes.

Moreover, our problem is different from Information Extraction tasks whose goal is to extract structured data such as organization name, people name from unstructured text. Our tweet text is unstructured; however, as we mention above paragraph, there is no any valid structured information all tweets in the data set. The same situation is also valid for company review page. In review texts, there might be no predefined structural information about a certain organization. For company Wikipedia page, since Wikipedia content is more formal, we can say to be had a kind of informational structure, but it is clear that, this might not be true for company review page. For example, company review pages do not have to contain any information about organization location. Therefore, our task is more challenging in comparison to Information Extraction tasks, since if our tweet content or both tweet and company profile content contained such structural information, the correspondence between tweet text and company text content might have been detected easily. That would lead to label such those tweets as 'true'easily.

1.5 Our Contributions

We use a highly rich company profile set with different sets of keyword to see their influence on our classification task. In addition to company home pages and company Wikipedia pages that are commonly used as external resources, we construct Company Wikipedia Disambiguation Page Profile, Company Review Page Profile. In addition, we generate noun phrase profile, Kullback-Leibler profile, term frequency profile, and Latent Semantic Indexing profile using both Wikipedia company page, and Wikipedia company disambiguation pages, and, we use these profiles in different ways to improve the accuracy of our classification task. At a high level, we aim to obtain relevant and irrelevant keywords about a certain company.

We construct Company Wikipedia Page Noun Phrase Profile and Company Wikipedia Disambiguation Page Noun Phrase Profile that include only noun and noun phrases

to see whether removing verbs from company profile make a good contribution to our classification task or not. In this way, we aim to bring nouns into the forefront.

We generate Company Wikipedia Page Kullback-Leibler Profile and Company Wikipedia Disambiguation Page Kullback-Leibler Profile to assign weights to words in order to select the most important keyword set for that text. We make a bit change to the Kullback-Leibler distance approach and computed word weights in the company web page text.

As Company Wikipedia Page Latent Semantic Indexing Profile and Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile, we use Latent Semantic Indexing unsupervised algorithm approach that uses Singular Value Decomposition method from Linear Algebra approach to find the most related sets of keyword in a document corpus. We generate Company Latent Semantic Indexing profiles using Wikipedia company page and Wikipedia disambiguation pages. Latent Semantic Indexing algorithm succeeds to find most relevant keywords for our classification task.

In order to compare the similarity between related and unrelated company profiles with a tweet, we use Company Wikipedia Page Profile, Company Wikipedia Disambiguation Page Profile, Company Review Profile, Company Wikipedia Page Noun Phrase Profile, Company Wikipedia Page Kullback-Leibler Profile, Company Wikipedia Page Term Frequency Profile, and Company Wikipedia PageLatent Semantic Indexing Profile and extract our cosine similarity features. In addition, we use Kullback-Leibler asymmetric distance to measure company profile-tweet similarity and we extract Kullback-Leibler distance as our other similarity feature.

In addition to the above numerical features, we extract commonly known tweet features like is capital or unigram. Alternatively, we consider whether a name of the organization in a given tweet has organization prepositions such as ‘at ’, ‘for ’, and ‘of ’in front of it. If so, this might be a strong indicator about tweet relevancy with a given company.

We do Attribute Selection to identify the most important features that have a big influence on our classification task. As different from other tweet-company classification approaches, the optimal set of classifiers that produce the best accuracy for our classification task are selected and combined by a fusion method based on Majority Voting. This shows better performance than using individual classifiers.

We also employ Threshold approach and Simple Approach algorithm on WePS-3 data set. The obtained results are surprisingly over our expectations. Threshold-based method provides approximately 71% accuracy performance, and Simple Approach algorithm achieves 70% accuracy with cheaper computational cost. When we compare our system with WePS-3 participant system, our system's performance is lower than only two

participants (Yerva (83%) and Yoshida (75%)), but higher than remaining participants. (see Table 4.20)

As distinct from other tweet-company classification methods, we use a similar algorithm (i.e. 'Entity Ranking') that is performed by [12] for their entity matching problem. In Entity Ranking Algorithm, two language models called 'Entity Mention Language'and 'Review Language'are used. The intuition behind this model is that for a given tweet, each word in the tweet is chosen with α probability from Entity Mention Language and (1- α) probability from Review Language. Entity Mention Language uses two distinct profiles: one is related with a given entity and the other is unrelated. Since we have two profiles, the algorithm computes the probability values of Entity Mention Language and Review Language. Then, the algorithm assigns the tweet as 'true'or 'false'depending on the returned profile values. More specifically, if the computed probability of the company unrelated profile is lower than the company related profile, the tweet is labelled as 'true', otherwise it is labelled as 'false'(i.e., it is not related to the company).

For Entity Ranking algorithm, we construct our related and unrelated company profiles as Wikipedia page profile, term frequency profile, Kullback-Leibler profile, noun phrase profile, and Latent Semantic Indexing profile from both Wikipedia company page and Wikipedia disambiguation pages with different keyword sets. Latent Semantic Indexing profile outperforms other profiles. Entity Ranking algorithm also outperforms over our baseline approach by approximately 9.5%. Statistical t-test also shows that the improvement over baseline is significant.

The rest of the study is organized as follows. Section 2 summarizes related work. Section 3 gives a more precise problem definition, presents our technique and gives more details on the classification techniques we used. Section 4 gives details on the experimental evaluation of our methods. Finally Section 5 concludes the thesis.

Chapter 2

Related Work

2.1 Company Disambiguation

- For this task, the research of Yerva et al. [13] shows the best performance in the competition. In this paper, authors focus on determining whether given tweets are related or unrelated with a certain company. The company profile including related or unrelated key words about company is created.

  For the tweet classification task, the content of the tweet is compared to the content of the profile. Yerva and his colleagues use a rich variety of company profiles for this task. These are:

  - **Homepage:** The WePS-3 Web site has URL of the company home pages. They crawl all the relevant links up to a depth level=2 from starting page in order to generate a profile that captures most of the keywords about company. In the profile the stemmed version of the keywords are stored.
  - **Metadata Profile:** HTML standards can provide a few meta tags, and these meta tags include some key words that are relevant to a given company.
  - **Category Profile:** Information about the category of the company provides relevant information about its entity. These kinds of keywords may not be in company home page, and they can be obtained via use of Wordnet.
  - **GoogleSet Profile:** Google Set is a good source in order to obtain a common knowledge about a company. This also allows to obtain relevant words about similar and competitive companies.
  - **User Feedback Positive:** In case of companies where sample ground truth is available, they infer the keywords from the tweets (in the training set) belonging to the company and construct User Feedback Positive Profile.

  - **Negative Feedback Profile:** This profile includes keywords that do not belong to the certain company. Gathering such words is a very challenging task. The Wikipedia Disambiguation Pages may help for some of the entities. This information may also be obtained via tweets that do not belong to this company in the training set.

After extracting entity profile features, tweet specific features and some heuristic features, Naive Bayes classifier is trained on these features. For each company in the training set, the conditional distribution of the related and unrelated classes are computed. Then, for an unseen tweet t, using the feature extraction function, the feature values are obtained. Whether the tweet is related to a certain company or not is determined based on the following: if the posterior probability of related class is higher than the posterior probability of unrelated class, the tweet is assumed to belong to company, and vice versa. With this approach, they get 0.83 accuracy, and demonstrate the best performance in the competition. Similarly, we use a highly rich company profile set with different sets of keyword. For each company, we construct twelve different profiles automatically, i.e., the first seven of the profiles are related to the company, and the remaining profiles contain unrelated keywords. For our problem, we use Wikipedia company page and Wikipedia company disambiguation pages to construct company profile vectors. In order to create company profiles, we use different following approaches. The one is extracting all words from Wikipedia company page and Wikipedia company disambiguation pages and assigning weight to them via tf.idf term weighting scheme, and the second is extracting noun and noun phrases (noun phrase approach) from those pages and giving weight by tf.idf approach. Our other approaches are Kullback-Leibler word weighting scheme, term frequency weighting scheme, and Latent Semantic Indexing weighting scheme, which are used in order to extract company related and unrelated keywords. Like User Feedback Negative Profile, we use company Wikipedia disambiguation pages to include unrelated keywords about a corresponding company. Moreover, we pick company review pages and parse them to obtain company related keywords. This profile can be assumed as User Feedback Profile; since company review pages may obtain both positive and negative comments about a corresponding company. Similarly, we extract features from entity profiles and tweet itself, then we employ several supervised learning algorithms on those features.

- In another study, Yerva [14] and his colleagues construct user profiles by integrating content from two different social networks Twitter and StackOverflow. They demonstrate that the content published on user's social networks may help for entity disambiguation problem considerably. When they compare their system with

the standard classifier that classifies a tweet only based on tweet keywords and company profile keywords, their system's classification accuracy increase noticeably.

While generating a user profile, they use different techniques. These are Term Popularity, Tf-Idf, Semantic Concepts and Categories, Topic Modeling (The top topics related to the user generated content extracted using Latent Dirichlet Algorithm). User's Twitter content is enriched with those profiles. Based on the Conditional probability values of P(C|M_i) and P($C^-|M_i$) where C represents company entity profile and M_i represents tweet enhanced content by user profiles (either Twitter or StackOverflow), they determine whether a tweet belongs to the given company or not. Since most of the tweets have not been posted by the users in WePS-3 dataset, they do not use WePS-3 data set. From 5 million tweets that they gather, they choose a tweet that include one of the six company words: apple, oracle, apache, subway, seat, orange. For each of those 6 keywords, they manually annotate a total of 100 tweets as 'true'or 'false'. This manual annotation acts as a ground truth for their problem. Their improvement table shows that for given companies and tweets, they reach 0.74 accuracy using user's Twitter profile and 0.77 accuracy using user's StackOverFlow profile (The accuracy performance of the baseline algorithm is 0.53). While constructing our external profiles, we use similar techniques that are employed in this paper. In order to create Wikipedia Company Page Term Frequency Profile, we use term popularity approach to give weight for words that are on the Wikipedia pages. Moreover, for Company Wikipedia Page Profile and Company Wikipedia disambiguation Page Profile, we use tf.idf term weighting scheme. Similarly, in order to find top most related topics from document corpus, we use Latent Semantic approach that produces good results like Latent Dirichlet approach.

- In this study, Yerva et al. [15] define a relatedness factor which is the percentage of tweets that belong to a given company. It helps them to understand the many limitations of the basic profile-based classifier. To overcome that, he and his colleagues inspect the messages from the Twitter stream, which contain the company name as a keyword. For each company, by inspecting the Twitter stream, they study the word frequency distributions. They observe that if they have a knowledge about all or top k of words, and if they know that whether these words contribute as positive or negative evidence, this would help them to classify many more tweets accurately. For this purpose, they use an active stream learning profile. According to the algorithm, from the inspected tweets, which overlap with the basic profile can accurately be classified. All words co-occuring with profile keywords in these tweets can be added to the profile. For tweets that do not overlap with the basic

profile are classified according to the relatedness factor. As a result of the relatedness classification, they have two sets of tweets either belonging or not belonging to a given company. For these sets , keywords above certain threshold are added to the profile as negative evidence keywords. They use WePS-3 data set for the above approach.The experimental studies demonstrate that the new classifier that identifies many more keywords by inspecting the twitter streams improves the accuracy significantly. Similarly, we use relatedness factor to show the number of non-overlapping profile-tweet for our Entity Ranking Algorithm. Based on the result of this factor, we come to a conclusion that since our company related profiles and company unrelated profiles have limited keyword set, our considerable amount of company tweets do not overlap with a corresponding company profile.

- In a related study, Yerva [4] Yerva and his colleagues look into the reasons for why some of the companies underperformed with their previous approaches. They observe that companies which do not demonstrate considerable success have mid-range relatedness factor. They generated a so called'perfect profile'by using the words inferred from the entire test set. By comparing the current profile to the perfect profile they observe that errors could occur in three different ways. The first is 'missing words error'i.e. the current profile may not contain words appearing in the perfect profile. The second is 'words weights error'i.e. the differences in word weights in the current and perfect profile can result in an error. The last is 'wrongly placed words error'referring to words that are marked as a positive evidence for classification can act a negative evidence. In order to reduce those errors, they use some error reduction techniques. Statistical analysis show that these techniques can increase the accuracy of companies that have a bad performance earlier. Like Yerva and his colleagues, we analyse companies whose performance are under our expectation. Similarly, we observe that missing words error and wrongly placed words error might have lowered our accuracy results. As a future work, we plan to use several approaches that might prevent to such those errors, which are explained in the Conclusion chapter.

- In WePS-3 competition, the second most accurate system ITC-UT team (Yoshida et al. 2010) reaching an accuracy of 0.75. They employ two steps. In the first step, they categorize each organization name in the training data into 3 or 4 classes based on the ratio of the accurately tweets and incorrectly tweets. For this categorization, six binary features are used e.g Is the query identical to the entity name? or Does Wikipedia have disambiguation page for the query? In the second step, for each category, tweet categorization is done based on the simple heuristic rules like whether entity name consists of two or more words. Their intuition is that company names in the data set are usually either organization-like-names (McDonald's),

or general word-like names (Pioneer), and organization-like-names lead to higher percentage of tweets related to a given company than general-word-like names do. We use similar binary features like does a company tweet includes company name as capital or not, and similar heuristic features that are used in this paper. Also, similarly, we use number of alternative meanings feature that shows the number of other meanings of a company which is obtained from Wikipedia disambiguation page of that company. More clearly, this information gives a precise information about company name ambiguity. In other words, if the number of other meanings of a corresponding company is high, the company has a high ambiguity and it seems general word-like names. As we mention in Experiments and Evaluation section, this feature makes a good contribution to the classification task.

- In [16], company tweets in WePS-3 are clustered as true or false according to the term expansion methodology. This methodology aims to enrich term representation of tweets. They use four different techniques: (1) Self-term expansion methodology (replacing terms of a tweet with a set-of co-related terms), (2) Term Expansion Methodology (in addition to the first approach, Wikipedia information is used for enriching process), (3) Term Expansion Methodology with Positive examples that uses the second enriching methodology with additional exclusive positive samples, and (4) is Full Term Expansion Methodology. For this purpose, they consider two types of company names, the first is generic (company names that tend to be very ambiguous), and the second is specific (company names that tend to be less ambiguous). Based on the experiments, they conclude that Full Term Expansion Methodology perform well on clustering tweets that belong to generic company names, and Term Expansion Methodology perform well on clustering tweets that belong to specific company names. In order to solve company name ambiguity, they employ clustering technique, which is different from our classification task. However, similarly, we use external source as Wikipedia company Web pages in order to generate company profile vector. Differently, while construction profile vector, we do not compute co-related relationship between terms i.e., pairwise mutual information as they computed in our task.

- In [17], the main approach is based on the idea of representing the information of a company in the form of a unique profile. This profile consists of a bag of stemmed words with their associated weights and which are obtained using a representation based on a fuzzy combination of criteria. Then, they employ the tweet disambiguation method by computing a comparison function between the company profile and tweet content. Lastly, an unsupervised threshold is used for categorizing each tweet as related or unrelated to the company. They test their application with the WePS-3 Online Reputation Management corpus, and they get 0.69 accuracy.

Different from us, they use only a single profile i.e., company home page, and they use a different approach in order to construct company profile (In our task, we use tf.idf term weighting scheme, Kullback-Leibler term weighting scheme, term frequency term weighting scheme, and Latent Semantic Indexing term weighting scheme to give weight to Web page words). They use a set of heuristic rules to define the importance of a term in Web page. The examples of those rules are, word frequency counts in titles, emphasized text segments, in the beginning and end of the document, and in the whole document.

- In [18], they use WePS-3 Online Reputation Management data set in order to solve the disambiguation problem. Both supervised (Maximum Entropy Classifier) and semi-supervised method (Label Propagation) are used, and they have considerable accuracy of 0.75.

- In [19], authors focus on a bootstrapping method to classify the tweets by collecting external company website information. Co-occurring words in each tweet are used as features. To compute the relevance of each word to a given company, they compute the pointwise mutual information between the word and the target's label i.e. 'related'or 'unrelated'.

- The research of [20] is based on some heuristics that use the named entities and external sources such as Wikipedia, DBPedia and the company home page for certain company names.

2.2 Entity Disambiguation

- The disambiguation problem also appears, when users are looking for Web pages of a specific individual person using the individual's name as a query. For instance, when a user query consists of a person name, search engine returns a ranked list of Web pages that correspond not only to the individual's interest but also to questioned person's namesakes. If a user is interested in Web pages of a particular person, he has to manually disambiguate the returned Web pages. If a user's interest is in the Web pages related to the someone other than the famous person, the user may have to scan through pages of search results in order to find relevant results which seems very challenging. In [21], authors focus on disambiguation problem for people search on the Web. In order to solve the problem, the direct solutions based on extracting features from Web pages such as n-grams, named entities, hyperlinks are developed. Since, direct features may not be sufficient in order to come up with correct clustering, indirect similarity computation is done.

Indirect similarity of Webpage pairs is computed using Web co-occurence statistics, which are collected using a set of queries which were submitted to a search engine such as Yahoo. These indirect features are used by a skyline based classifier, that learns how to convert the indirect similarity-based features to a 'merge'or 'do not merge'decision. Direct and indirect similarities are then combined to form overall similarity that is used to label the graph G = (V,E) where V represents the set of Webpages to be clustered and E represents the set of edges, and an edge is created per each distinct pair of Webpages. The clustering step clusters the Web pages based on the collected similarities. The labelled graph G is partitioned into its clusters using the correlation clustering. After correlation clustering is applied, the result frequently consists of a few large clusters and several singleton 1-Web page clusters. There can be true or false 1-Web page clusters. Therefore, the second step in the clustering process aims to refine false singleton clusters. The approach is as follows, firstly the similarities between this one page Web page cluster and the remaining Web pages are computed. If their similarity exceeds a certain threshold that is estimated per queried name by exploiting the number of clusters, the Web pages are merged with another cluster.

Each resulting cluster is processed to summarize the content of the cluster, then clusters are ranked to decide the order while they are presented to the user. Lastly; in each cluster, Web pages are ranked. Therefore, the results are presented to the user in the form of clusters corresponding to namesakes. Our problem is different from person disambiguation problem. For our task, the set of organization names in the training and test corpora are different. The model could not be trained for a certain organization.

- In [22] , the author disambiguation problem is studied. The problem arises when entities in a database contain references to other entities. References can be ambiguous due to differences in the descriptions of the same entity and errors in data entries. More clearly, the disambiguation problem often arises when multiple tables are merged to create a single table. The authors also emphasize that recent surveys show researchers who are working on data mining projects spend more time for data cleaning and data preprocessing in the case of merging information from heterogeneous sources in a single database. To this end, besides feature based similarity methods that analyse similarity of entity groups, quality of disambiguation can be significantly improved by additional semantic information. Relationship-Based Reference Disambiguation approach is used in this paper. It exploits not only features but also relationships among entities for the purpose of disambiguation. Relationship-Based Reference Disambiguation approach views the database

as a graph of entities that are linked to each other via relationships. After identifying a set of entities (choices) by using feature-based method for a reference to be disambiguated, then graph theoretic techniques are used to discover and analyse relationships between the reference entity and a set of candidate entities. The generic approach, firstly connections between entities which the reference appears in and matching the candidate entities are discovered. Then, the strength of the discovered connections is measured in order to rank the matching candidates and choose the top one.

- In [23], they develop an algorithm that aims to solve disambiguation problem in e-mails using graphs. When an informal nickname is used or when the mentioned person does not appear in e−mail header, it is difficult to find out which person is referred. Therefore, resolving the reference to a person name is an important task for entity name extraction. For this task, the dataset includes those names that are in the header, but can not be matched to the text because they are referred to using initials. Nicknames refer to name mentions including common nicknames, or American names that are adopted by persons with foreign language names. Based on a name mention in an e-mail message m, it is formulated a query distribution V_q, and then a ranked list of person nodes is retrieved. In this task, a base-line method is used, and graph walk method provides more accurate results than the baseline.

2.3 Entity Matching

- In [24], the SHINE approach is proposed. This approach is based on linking the named entities in Web text with heterogeneous information networks that consist of multi-type interconnected objects. Their approach is the first probabilistic model for this purpose. The probabilistic approach uses entity popularity model (meaning the popularity of an entity) and entity object model that refers to the probability of the multi-type object distribution in Web text from the heterogeneous information network. Since multi-type objects have different types of connections that form to a set of meta-paths (For example, in a bibliographic dataset, objects of multiple types, such as papers, authors, publication venues and title terms have relations of multiple types such as write, publish and contain). Meta-path is defined as including a sequence of relations between different object-types. In order to learn the weights for each meta-path, the Expectation-Maximization algorithm is used. Statistical analysis shows that the SHINE approach has better efficiency than the baselines.

- [25] focuses on the probability of a given attribute value reappearing over time. The idea is that an entity might change its attribute value as depending on its past value. It uses the temporal information of entity records in the form of time stamps. In the training phase, how frequently values evolve in each attribute of the record set is learnt. Then, they calculate the likelihood that the record attribute changes its value that have never seen before. This method is called the mutation model. Their approach uses the mutation function that computes the probability of entity's mutant record for given attribute at a certain time.

- In [12], a method they develop to match unstructured text reviews to a structured list of objects. Review language model that they use gives them a principled method that, given a review, finds the object which is most likely to be the topic of the review. In this paper, they explore the scenario of matching reviews to objects using only their textual context. They propose a general method to match objects to reviews. When a review is written about an object, each word in the review is drawn either from a description of the object, or generic review language that is independent of the object. The experiments and their extensive analysis show that their language model-based method significantly outperforms traditional tf.idf based methods. We use their entity matching problem to determine whether a given tweet is relevant or irrelevant with a corresponding company. In our adaptation, we use two language models: one is entity mention language and the other is review language model. Similarly, our review language model includes all review data for training and testing companies. Also, our entity mention language consists of two profiles: one of the profiles consists of Wikipedia company keywords, and the other profile consists of Wikipedia disambiguation company keywords (We are not given any information in this article about Entity mention language that they use). We use the same algorithm in this article, and it computes the value for both of two profiles based on the probability values of entity mention language and review language model. Then, the algorithm assigns the tweet as 'true', or 'false'depending on the returned greatness of the profile values. Also, we show that our tweet classification results based on this algorithm outperforms our baseline approach, too.

- [26] studies the object matching problem in tweets. They formulate a simple user model for generating a tweet about an object. The model depends on the tendency of a user to tweet about an object, the object's popularity and the distance between the user and object's geographic location. They compute the probability of a user tweets about an object as following: the product of the user's interest about an object, the popularity of the object, and the function of the distance between a user

and the object locations. Statistical analysis show that their geography-enabled model provides improved performance over geography-less models.

- In [27], they focus on matching reviews to given objects using a translation model. This model is based on generating a word for the object from object's attributes. Here attributes include name of the object, city of the object and cuisine of the object (The experiments are done both restaurant reviews from Yelp and movie reviews from IMDb). They use generative model and learn the parameters of the generative model using Expectation-Maximization algorithm. Their experimental analysis shows that their model gives better results than other object-review matching problem approaches.

2.4 Social Media Analysis

- It is reasonable to make an assumption that public mood including i.e., anxiety can drive stock market values as much as news, over the past years. In [28], authors explain how Twitter mood influences the stock market. For this purpose, measurements of collective mood states derived from large-scale Twitter feeds are correlated to the Dow Jones Industrial Average over time. In this paper, two tools are used in order to measure the variations in the public mood from tweets submitted to the Twitter service during a certain time. The first tool is Opinion Finder that measures the tweet in a given day text as negative or positive of public mood. The other is GPOMS that generates a six dimensional daily time series of public respectively **mood**, **calm**, **alert**, **sure**, **vital**, **kind**, and **happy**. The experimental results show that there is a strong correlation between the certain public mood dimensions and the stock market prediction.

- In [29], the terms in microblog posts are linked to Wikipedia pages in order to use Wikipedia's link structure to estimate semantic similarity. Their method is the following: useful feature terms are extracted using Wikipedia. As feature terms, only the terms that are used in Wikipedia at least once as an anchor text for a link are used. Combining those features, the distance between microblog posts are measured. Then, based on the semantic similarity measurements, an unsupervised topic detection method is used to cluster microblog documents. Lastly, for each topical cluster, the topic is labelled with a selected term.

- In [30], the Hydra approach is proposed to solve the problem of automatically linking user accounts belonging to the same user across different social platforms. For this problem, they use 5 social network service that are originated from China and two globally social networks, Facebook and Twitter. They propose a model, called

'Heterogeneous Behaviour Modeling'in order to measure the similarity between two users using several aspects. The first one is user attributes which are either textual attributes like name, gender, age, or visual attributes like face images used in the user profile. The second is user's topic interests, the third is the language style of a user such as personalized wording and emoticon adaptation that might be beneficial to distinguish different users. The last one is information about user's location and multimedia sharings like image, video or music on the Web. Based on the above behaviour modeling, they propose to learn a linkage function via a multi-objective optimization system. This system is based on the decision model on pairwise similarity and users' social structure consistency information. Statistical analysis show that their HYDRA approach outperforms the standard algorithms in predicting the user identity across different platforms.

- In [31], the authors explain that in daily life, tweets are ranked in chronological order regardless of their potential interestingness. Therefore, more personalized ranking scheme is needed to filter the overwhelmed information. In this study, they focus on, how to learn a predictive model to rank the tweets in order to determine what tweet's are likely to attract one's attention, according to their probability of being re-tweeted. With this approach, users can find interesting tweets in a short time. For this work, they generate a graph consisting of 3 types of nodes, users, publishers, and tweets. To incorporate all sources of information like users profile, tweet quality, interaction history, nodes and edges are represented as feature vectors. All these feature vectors are mapped to node weights and edge weights. According to the graph model, feature aware factorization model is designed, which can re-rank the tweets and fully explore all the information in the graph for prediction. Different from the previous studies, this work is focuses on local factors at individual level.

- In [32], a natural alternative for advertising keyword recommendation for short-text web pages is to recommend relevant key words not present in the target web page by leveraging the content of Wikipedia is proposed. Given a target web page, they propose to use a content biased Page Rank on the Wikipedia graph to rank the related entities. More clearly, advertising keywords are extracted from web page, then these keywords are used in order to find the relevant ads. The short-text web pages contain little information which makes ranking difficult for a recommendation system. Therefore, in order to overcome this problem, the existing advertising keywords will be enriched with those keywords that are relevant to the target Web page even if they do not appear on the target web page. Advertising keywords are analysed to find a relationship between existing ones and new obtained keywords, which are called 'leveraged keywords'that are semantically relevant with each other.

This paper's approach uses Wikipedia pages as the dictionary of the recommended keywords. Structurally, Wikipedia can be viewed as a directed graph with vertices and edges corresponding to its entities. Page Ranking algorithm is used to infer related keywords. Also, two kinds of biased are used, which are called content biased and advertisement biased making it possible to extract advertising keywords that are both relevant with a target page and a valuable for advertising.

- The main idea behind [33] is that words co-occuring in text similarly refer to concepts that close together in the Dbpedia graph. The approach is graph-based topic labeling by using Dbpedia. Dbpedia subgraph of topic labels is extracted, and then network centrality measures are adapted so that represent a good label for a topic. The most important improvements are better corpus coverage and much higher ability to represent broader labels. In this field, one of the best known multi-domain knowledge is Dbpedia which extracts structural information from Wikipedia. This paper proposes to extract topic labels in text documents by linking the inherent topics of a text to concepts found in Dbpedia, and mining the resulting topic graphs. The goal is not only finding a good label but also integrating a topic with related concepts. An important aspect of this work is relating a topic label with a URL which identifies a concept. This opens a way to knowledge exploration in Dbpedia. Using graph centrality measures, concepts that are most likely to represent the topic are identified.

- In [34], the authors make use of Twitter messages for the task of sentiment analysis in order to classify tweets as positive, negative, or neutral sentiments. A sentiment classifier is used on a tweet corpus. Companies want to become aware of positive or negative comments with regard to themselves through a social media. Our work can be a preprocessing step of sentiment analysis task. After the relevant tweets are identified, those can serve for sentiment task.

Chapter 3

Problem Statement and Our Approach

3.1 Problem Statement

Given a set of tweet t and an organization name o, our goal is to determine whether t is related to o or not. An input tweet is composed of: the tweet identifier, the entity (organization) name, the query used to retrieve the tweet, the author identifier and the tweet content. For each organization in the dataset, we are given the organization name and its homepage URL. The output per tweet is 'True'or 'False'tag corresponding to related or non-related to the given organization. Compared with the conventional classification process, this task has some challenges as presented in the first chapter in detail. Briefly, the main challenge is that tweet and organization name contain little information, i.e., contextual information is very limited. To overcome this problem, we create several profiles for a company, each of which is either related or unrelated to the company.

3.1.1 Company Profile Representation

We represent each company as a collection of several profiles, formally:

$$C_k = (P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}) \tag{3.1}$$

$$P_1 = (\text{Company Wikipedia Page Profile}) \tag{3.2}$$

$$P_2 = \text{(Company Wikipedia Disambiguation Page Profile)} \tag{3.3}$$

$$P_3 = \text{(Company Home Page Profile)} \tag{3.4}$$

$$P_4 = \text{(Company Review Page Profile)} \tag{3.5}$$

$$P_5 = \text{(Company Wikipedia Page Kullback-Leibler Profile)} \tag{3.6}$$

$$P_6 = \text{(Company Wikipedia Disambiguation Page Kullback-Leibler Profile)} \tag{3.7}$$

$$P_7 = \text{(Company Wikipedia Page Noun Phrase Profile)} \tag{3.8}$$

$$P_8 = \text{(Company Wikipedia Disambiguation Page Noun Phrase Profile.)} \tag{3.9}$$

$$P_9 = \text{(Company Wikipedia Page Term Frequency Profile)} \tag{3.10}$$

$$P_{10} = \text{(Company Wikipedia Disambiguation Page Term Frequency Profile.)} \tag{3.11}$$

$$P_{11} = \text{(Company Wikipedia Page Latent Semantic Indexing Profile)} \tag{3.12}$$

$$P_{12} = \text{(Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile)} \tag{3.13}$$

Each profile is a set of weighted keywords. For better classification results, a company profile should have a good overlap with its tweets. We are not given tweet messages in

advance, so we create such profiles from alternative external sources, independent of the tweet messages. A company profile should not be too general, because it would result in many false positives during the classification, and also not too narrow; otherwise, we could miss potential relevant tweets.

3.1.2 Tweet Representation

Each tweet is represented as bag of words i.e., the occurrence of the word is used as a feature for classifier. Initially these tweets are pre-processed by removing html tags and stop words. Then, stemming and tf.idf computation are performed for each tweet. Lastly, the tweet profile vector is generated as tuples of words and their weights. The vector for i'th tweet inluding n words can be represented as:

$$V_i = \text{set}(\text{word}_1, \text{weight}_1), ..., (\text{word}_n, \text{weight}_n)$$

3.2 Our Approach

Using the provided training data, we train a classifier with generic features. The features should not be too general which may lead to biased the preference to tag tweets as 'True', or too narrow which may lead to biased the preference to tag tweets as 'False'.

The features should be generated automatically, with no manual labelling. We propose to build and use a set of supervised classifiers on the extracted features. The first step is to select the features, which will maximize the accuracy of classification. In 3.1, you can see our disambiguation approach visually.

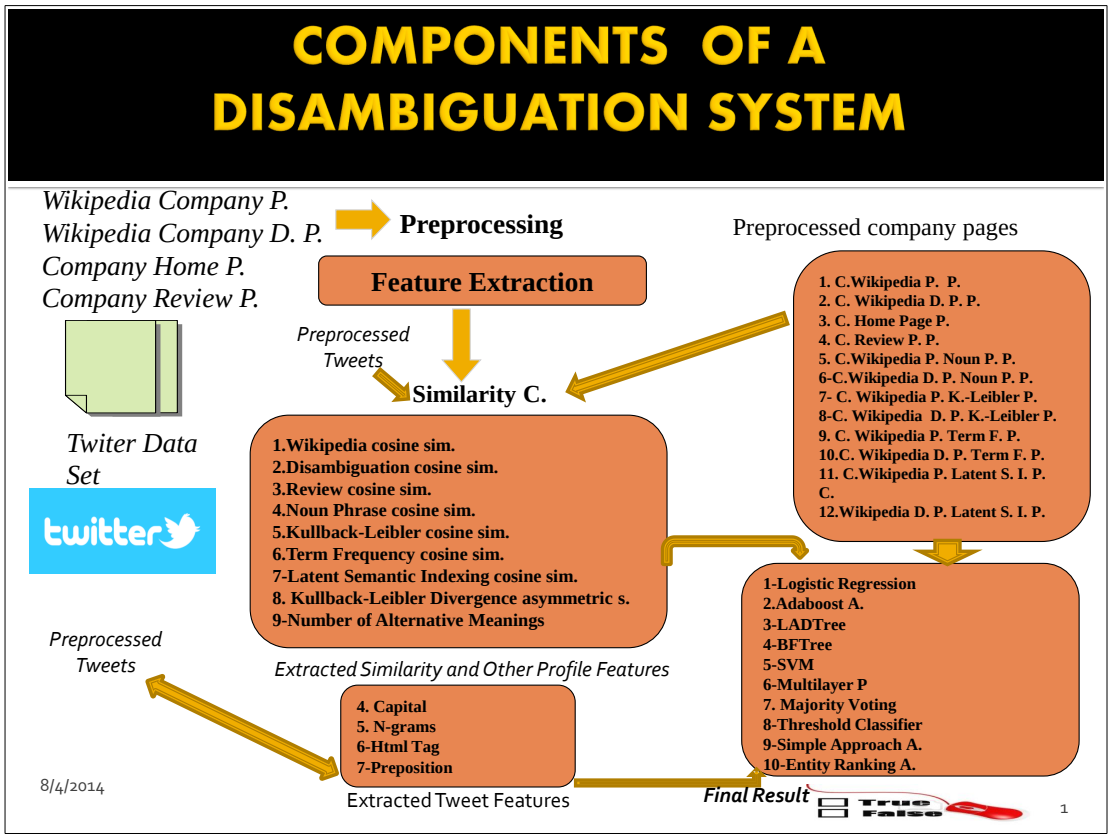

FIGURE 3.1: Components of a Disambiguation System.

3.2.1 Data Preprocessing

We first explain several data-preprocessing techniques that are widely used for our classification task. Data-preprocessing methods transform raw data into an 'understandable format'. In this way, irrelevant, redundant, noisy and unreliable data are removed, which makes data content cleaner. Data pre-processing includes data cleaning, data normalization, data transformation, feature extraction, feature selection, etc. The output of data pre-processing is the final cleaned-up training set.

3.2.1.1 Removing Stopwords

While generating company profile, it is important to remove all stop words like 'a, an, the, ...' in order to get the clear content. For this purpose, beside using Python's NLTK library [35], we generate a custom list of ignorable words that leads to much more reasonable results. You can see our additional stopword list in Appendix section. Moreover, to prevent the bias, we also ignore company name keywords from both Tweet vector and company profile vector.

3.2.1.2 Stemming

In Information Retrieval, stemming is the process of converting the derived words to their stems and roots. This data transformation process provides the capability of grouping the keywords from the same root. For instance the stem for computer and computation is comput. In this thesis, both the Porter Stemmer and Lancaster Stemmer is tested. However, since the Porter Stemmer is more popular and commonly used in many applications, we determine to use it.

3.2.1.3 Lemmatization

Lemmatisation is the algorithmic process of determining the dictionary form of a given word in the context. For example, In English, the verb 'to walk' may appear as 'walk', 'walked', 'walks', 'walking'. The dictionary form of 'walk' is called the lemma for this word. The other example is that the word 'better' has 'good' as its lemma. For our problem, the words are lemmatized using NLTK's WordNetLemmatizer [36]. Lemmatization can be seen as a form of data transformation and preprocessing step.

3.2.1.4 Data Normalization

Data Normalization is the process of transforming all variables in the data into specific range of values. By normalization, the similarities of two documents can be compared even, if one of them is small, and the other is large.

3.2.1.5 Generating Vector Space Model

The representation of the documents as vectors in a common vector space is known as the 'Vector space model'[**?**]. This refers to a vector representation of a document d includes the words in d with their weights. The Vector Space Model is commonly used in different Information Retrieval tasks, such as document classification, clustering, retrieval, etc.. The weight vector for document d is:

$$[V_d] = [w_{1,d}, w_{2,d}, \cdots, w_{n,d}]^t$$

represented as a bag-of-words model.The weight of the each term in a document is computed commonly via term frequency-inverse document frequency (tf.idf) model:

$$w_{t,d} = \left(tf_{t,d} \cdot \log \frac{|D|}{|d^{'} \in D | t \in d^{'}|} \right) \tag{3.14}$$

where $tf_{t,d}$ denotes term frequency of term t in document d. $tf_{t,d}$ value is computed as term frequency of term t in document d is divided by the document length i.e., the number of words in the document. The other component of the weight computation $\log \frac{|D|}{|d' \in D | t \in d'|}$ is inverse document frequency where $|D|$ is the total number of documents in the document set, and $|d^{'} \in D | t \in d^{'}|$ is the number of documents containing the term t.

3.2.2 Feature Extraction

The feature extraction includes organization features (i.e. company) and tweet features. In this step, we pay more attention to the organization information, which is enriched with external resources. We use Wikipedia company web page, company home page and company review page, which include the keywords related to the company and the Wikipedia disambiguation pages which include the keywords unrelated to the company to get features representing an organization.

3.2.2.1 Numerical Features

Company Wikipedia Page Profile

We use the Wikipedia page of a company with the goal of getting higher quality and rich data about the company and its products and services. The most important words on these pages are mostly relevant to the specifics of the company, and can easily be used to identify the product/company. We first download the Wikipedia pages and the pages linked from these Wikipedia pages up to depth=2. These pages are then parsed and the text in the pages are used to derive the Company Wikipedia Page Profile for the company.

In order to generate this profile, then we compute the weight of each term in these parsed text files using the tf.idf term weighting scheme. Most important 100,250 and 500 keywords are then selected as to represent the company profile as a vector. For example, for Apple Inc., the first five keywords with their associated weights are:

$$V_{profileWiki} = [('ipad', 0.532), ('mac', 0.484), ('iphone', 0.422), ('store', 0.366), ('steve', 0.322)]$$

Company Wikipedia Disambiguation Page Profile

We generate an alternative profile for each of the company using the Wikipedia pages of the other meanings of the company name. Wikipedia provides a disambiguation page, when an entry might mean more than one entity in the real world. For example, Apple disambiguation web page includes all other meanings of 'apple'. Thus, we use these disambiguation pages to find the other meanings of the company name and each namesake's. The purpose of creating these Wikipedia disambiguation page profiles is to find out the most important terms that would help distinguish between the company we are interested in and all other real world entities with the same name.

To create the profile, we again perform stemming, lemmatization as we do in Company Wikipedia Page Profile creation. Terms are weighted using tf.idf and ranked. Top K (100,250,500) keywords are then selected as the Company Wikipedia Disambiguation Page Profile.

$$V_{profileWikiDisambiguation} = [('river', 0.187), ('valley', 0.185), ('band', 0.183), ('card', 0.172), ('store', 0.169)]$$

Company Review Page Profile

Tweets usually mention companies in an informal context. They do not include complete and clearly identifiable company names. Thus, with the assumption that if we can capture the informal context/language in a more content rich platform we might be able to get better company profiles, we create the review pages-based company profiles. We expect that the company review profile keywords will have better overlap with the tweets mentioning the company.

Since, our data set includes companies providing services different categories, we have to pick company review pages from several web sites that suits to the company function. For instance, for the companies in the Food/Restaurant categories (e.g., Friday's and McDonald's), we use restaurant review sites such as Yelp [37] and Pissed Consumer [38]. These sites contain information about the businesses in addition to the customer reviews. Amazon [39] and CNET [40] are used to collect reviews about the technology companies

such as Alcatel and Apache, while AirlineQuality.Com [**?**] is used to gather comments on airlines. For hotels and reservation companies Tripadvisor [41] is exploited. For the companies that are in other industries such as automotive and education, we select a representative set of sites that has enough consumer reviews on those companies.

The data collection process involve crawling the mentioned review sites. Instead of crawling the whole site, we only crawl the initial pages about the company and the pages linked from those pages (crawling depth = 2). Same preprocessing steps are applied to this crawled data. Then the top K (100, 250, and 500) terms ranked according to their tf.idf are selected as Company Review Page Profile. For instance, we show the first five keywords about Apple Inc. below.

$$V_{profileReview} = [('inch', 0.999), ('iphon', 0.583), ('iwatch', 0.429), \\ ('macbook', 0.403), ('speaker', 0.363)]$$

To improve the classification accuracy, we perform some *additional filtering method* by fixing the typos in the review text, which are prevalent.

Firstly, we apply on to do fuzzy matching process. We initially detect misspelled words in review data using Python's Enchant library [42]. Enchant library includes dictionaries that have different language tags like Spanish or English. These dictionaries are used to check the spelling of words in the text and to get suggestions for misspelled words. Using Enchant's default English tag dictionary, we obtain misspelled words. Then, in order to do string matching for misspelled words, we use Levenshtein distance metric [43] for measuring the difference between two strings. The Levenshtein distance between two words refers to the minimum number of single-character edits, i.e., insertions, deletions, or substitutions require to change one word into the other. Technically, for a misspelled word, we obtain all the possible strings that have minimum Levensthein Distance to a given erroneous word. However, in the case of returning more than one word that have the minimum distance, there is a second job that we have to choose more meaningful word that is most related with a given concept manually, so this method is useless.

Next, we apply another method that succeeds to eliminate misspelled words considerably. We enrich the keyword set of the Enchant's default dictionary by generating additional entries. In order to prevent incorrect filtering, we create very large dictionary containing both company-based words and general public language words. For that purpose, we use a big text file which consists of about a million words. The file is a concatenation of several public domain books from Project Gutenberg and lists of most frequent words

from Wiktionary and the British National Corpus. This file is obtained from Peter Norvig's Web site [44]. As this file involves general language words at a high level, the created dictionary only using the words in this file may not be sufficient for filtering process. Therefore, we crawl 2 level of each company's home page, then we extract all keywords and add into the dictionary that we have created. Then, Enchant's default dictionary and the generated dictionary discard misspelled words automatically. Before filtering, some of the keywords taken from Apple Inc. profile vector is:

['tbdtax', 'seller', 'instal', 'techtrack', 'msrp', 'tbdmsrp', 'offens', 'softwar', 'iphon', 'prohibit', 'unavail', 'imac', 'wireless','inch', 'shop', 'macbook', 'cool', 'devic', 'speaker', 'attempt', 'messageif', 'capsul', 'fullest']

As seen from the keyword set, most of the words seem very meaningless or irrelevant that would not make any contribution to classifying process. The given example representing 5 keywords of Apple Inc. above involves the second method that we mention. As you from the example, the second technique makes the profile content more clear and understandable.

Company Wikipedia Page Kullback-Leibler Profile

Kullback-Leibler divergence measures the relative entropy between two probability mass functions namely P and Q. When sum of P and Q is 1, and for any i, if $P(i) > 0$ and $Q(i) > 0$, we define their Kullback Leibler Divergence based on this formula:

$$KL(P||Q) = \sum_{x \in X}^{n} P(x) \cdot \log \frac{P(x)}{Q(x)}$$

In the document scenario, we consider a document d as discrete distribution of $|d|$ random variables, where $|d|$ is number of words in the document. Let d_1 and d_2 be two documents that we want to calculate their Kullback Leibler distance. The divergence between two distributions of words is:

$$D_{KL}(\vec{t_{d_1}}^{\,t}, \vec{t_{d_2}}^{\,t}) = \sum_{t=1}^{m} w_{t,d_1} \cdot \log \frac{w_{t,d_1}}{w_{t,d_2}}$$

where w_{t,d_1} represents term frequency of term t in document d_1 and w_{t,d_2} denotes term frequency of term t in document d_2.

We adapt the formula in 3.15 that is originally used for document similarity to a weight computation scheme.Formally, for term t in document d, we use the formula in 3.16 to

compute the weight of the t in d in order to create other company profile vector. For this purpose, we use the following formula:

$$kullbackLeibler(t,d) = (w_{t,d} \cdot \log(w_{t,d})) \tag{3.15}$$

where $w_{t,d}$ represents term frequency of term t in document d.

In order to create the related profile, we download the Wikipedia pages and the pages linked from these Wikipedia pages up to depth = 2 as we perform to build Company Wikipedia Page Profile. Then, the same preprocessing steps are applied to the downloading data. Then, the top K (100,250, and 500) most important terms in the parsed text are ranked according to their computed weights by 3.16 and Company Wikipedia Page Kullback-Leibler Profile is constructed.

According to the formula 3.16, when term frequency value of a term increases, the obtained weight value decreases. Therefore, initially, we obtain the weight values which are less than zero. Then, we need to normalize ranked terms by term weight which has the smallest value, so that profile values are mapped between 0 and 1. As you see from Apple Inc. examples for both related and unrelated profiles, the most important keyword for the company profile has the greatest value which is 1. For example, for Apple Inc., the first five keywords with their associated weights become as:

$$\begin{aligned} V_{profileKullbackLeibler} = [&('comput', 1.0), ('system', 0.983), ('product', 0.69), \\ &('softwar', 0.634), ('macintosh', 0.588)] \end{aligned}$$

Company Wikipedia Disambiguation Page Kullback-Leibler Profile

In order to create Company Wikipedia Disambiguation Page Kullback-Leibler Profile, we download and parse the Wikipedia disambiguation pages to get the other meanings of the company content as we generate Company Wikipedia Disambiguation Page Profile. Then, we rank the top K (100, 250, and 500) most important terms in the parsed text according to their computed weights by 3.16, and we construct Company Wikipedia Disambiguation Page Kullback-Leibler Profile. After performing the same normalization method that we discuss for Company Wikipedia Page Kullback-Leibler Profile, the first five keywords for Apple Inc. with their associated weights become as:

$$V_{profileDisambiguationKullbackLeibler} = [('nonprofit', 1.0), ('textdecor', 0.956), ('revis', 0.926), ('fruit', 0.834), ('list', 0.755)]$$

Company Wikipedia Page Noun Phrase Profile

Our previous company profiles contain vocabulary from linguistic categories both noun and verb. We believe that representing any company with its extracted significant nouns is more comprehensive way rather than representing that company with verbs extracted from company profiles. Therefore, we eliminate verbs from profiles to bring nouns into the forefront. Grammatically, adjectives and adverbs have a meaning when they are used with other words. Since adverbs do not make much contribution to represent company content noticeably, we discard adverbs from company profiles. Hence, so we construct a company profile vector including nouns and noun phrases.

Firstly, we tag [45] the given list of tokens using NLTK's postag module. Then, we explore chunking which segments and labels multi-token sequences. In order to do that, we define a rule that finds a chunk structure in a given text using NLTK's RegexpParser module. The rule is extracting single noun, noun + noun and, adjective+noun in a given text. The rule that we use is represented in 3.2.

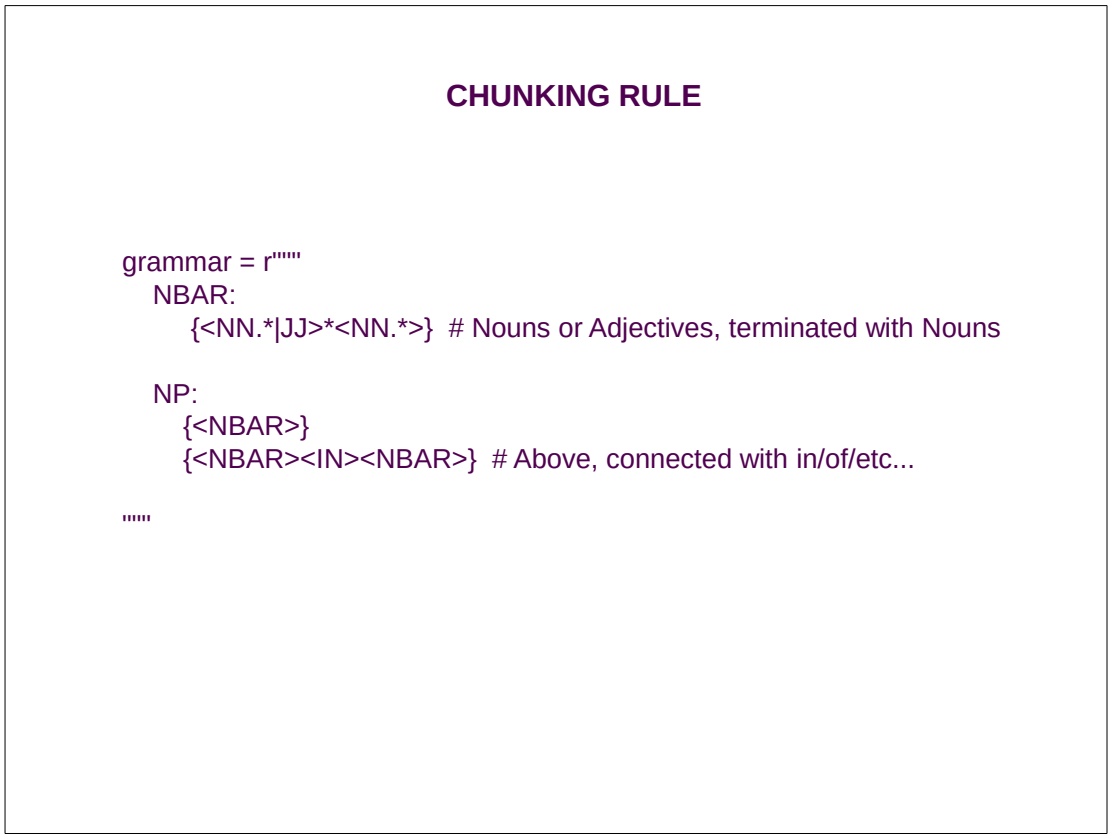

FIGURE 3.2: Chunking Rule for Extracting Noun + Noun Phrase

Formally, the rule says that any number of noun (NN) or adjective (JJ) is followed by any number of noun (NN). Using this grammar rule, a chunk parser is created, and a chunk tree is produced. For instance, given a sentence in the following:

This is the best digital camera.

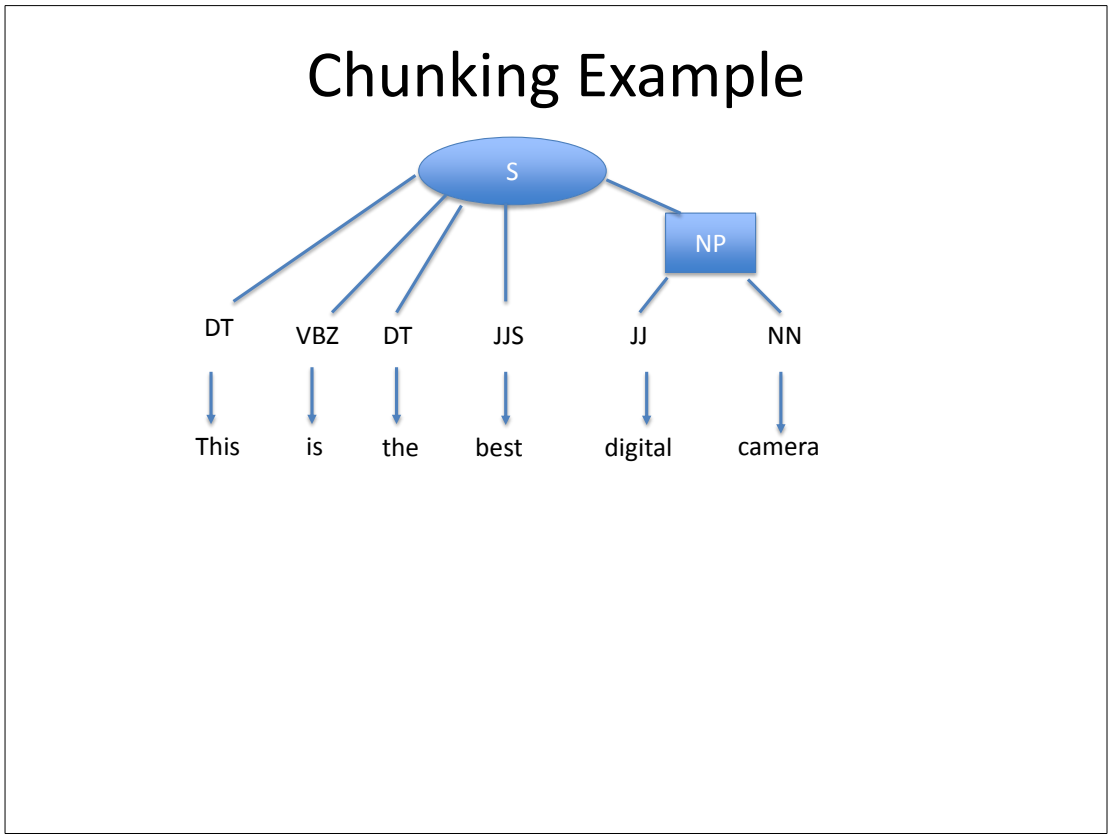

FIGURE 3.3: Chunking Example for Extracting Noun + Noun Phrase

The chunk tree based on the example is constructed in 3.3.

For our problem, we feed obtained tokens to the *chunker* as a parameter to obtain noun+noun and adjective+noun words from the text. Then, we select top 100,250 and 500 important keywords as company Wikipedia page noun phrase profile vector for Wikipedia company pages. For Apple Inc., the first five keywords based on Company Wikipedia Page Noun Phrase Profile are:

$$V_{profileNamephrase} = [('macintosh', 0.756), ('iphon', 0.731), ('ipod', 0.575), \\ ('ipad', 0.529), ('powerbook', 0.495)]$$

Company Wikipedia Disambiguation Page Noun Phrase Profile

In order to create company unrelated profile that includes noun and noun phrases, we apply the same chunking strategy to Wikipedia disambiguation pages for the corresponding company. Then, we select top 100,250 and 500 important keywords as company Wikipedia disambiguation page noun phrase profile vector. For Apple Inc., the first five

keywords based on Company Wikipedia Disambiguation Page Noun Phrase Profile are:

$$V_{profileDisambiguationNounphrase} = [('appel', 0.613), ('river', 0.371), ('automobil', 0.354), ('band', 0.297), ('custard', 0.278)]$$

Company Wikipedia Page Term Frequency Profile

Term frequency is often used in Information Retrieval and Text Mining in order to weight document terms. Term frequency captures how often a particular word appears in a document. After performing data preprocessing steps as we apply to construct other profiles, we build this entity profile vector based on the term frequency metric for Wikipedia company pages. We chose the top 100, 250 and 500 terms that have the highest term frequency score. In order to do this computation, we use this formula:

$$termfrequency(w_i, D) = \left(\frac{freq(w_i, D)}{maxfreq(w_k, D); \text{for word} w_k \in D} \right)$$

For Apple Inc., the first five keywords based on Company Wikipedia Page Term Frequency Profile are:

$$V_{profileTermFrequency} = [('comput', 1.0), ('iphon', 0.711), ('system', 0.546), ('disk', 0.538), ('macintosh', 0.134)]$$

The profile example for Apple Inc. shows that the weight of the most significant keyword is 1. However, except Company Wikipedia Page Kullback-Leibler Profile and Company Wikipedia Disambiguation Page Kullback-Leibler Profile, the most important keyword for the corresponding profiles has less than 1 weight value. The reason is the following: Since we use company Wikipedia page and links on that page, we have more than one document. Also, while computing a word weight, we gather all word weights in the documents, so the total weight of the word may become greater than 1. Therefore, we need to normalize profile terms with the greatest term weight in that profile, so that a word that makes the greatest contribution to the classification task has the value of 1.

Company Wikipedia Disambiguation Page Term Frequency Profile

We construct the entity unrelated profile vector based on the term frequency metric which is defined above for Wikipedia company disambiguation pages. We choose the top 100, 250 and 500 terms that have the highest term frequency score. Then, we do

the same normalization method that we discuss for Company Wikipedia Page Term Frequency Profile. For Apple Inc., the first five keywords based on Company Wikipedia Disambiguation Page Term Frequency Profile are:

$$V_{profileDisambiguationTermFrequency} = [('tomato', 1.0), ('cashew', 0.487), ('beatl', 0.476), ('band', 0.418), ('record', 0.417)]$$

Company Wikipedia Page Latent Semantic Indexing Profile

Latent Semantic Indexing (LSI) [46] is an Information Retrieval method that is capable of retrieving text based on the concepts that it contains. LSI has the ability to correlate semantically related terms that are hidden in a collection of documents. LSI is based on the principle that words that are used in the same contexts tend to have similar meanings.

LSI firstly constructs a term-document matrix, formally A, to represent the occurrences of the m terms within a collection of n documents. In a term-document matrix A, each term is represented by a row, and each document is represented by a column. Each cell $a_{i,j}$ in the matrix shows the number of times the associated term appears in the given document. This matrix is usually very large and very sparse.

Once a document matrix is constructed, each cell counts are modified using tf.idf weighting formula given by 3.14. As a result of that, rare words are weighted more heavily than common words. Next, LSI performs Singular Value Decomposition on the matrix A to determine patterns in the relationships between the terms and concepts used in the documents. Singular Value Decomposition computes the term and document vector spaces by transforming the single term-frequency matrix A into three other matrices. These are an m by r term-concept vector matrix T, an r by r singular values matrix S, and a n by r concept-document vector matrix D, which satisfy the following relations [47]

$$A = [TSD^T]$$

$$[S_{1,1} \geq S_{2,2} \geq S_{r,r} \geq 0]$$

Then, the singular value matrix S is truncated to size k by r, document vector size is truncated to n by k, and term vector matrix size is truncated to m by k. This is called dimensionality reduction which is one of the influential preprocessing techniques.

Along with this reduction, while reducing noise and other undesirable effects of the original space of A, Singular Value Decomposition preserves the most important semantic information in the text. In other words, it makes the best possible reconstruction of the matrix with the least possible information. The reduced set of matrix is that:

$$A_k = \left[T_k S_k D_k^T\right]$$

T_k matrix gives us the coordinates of each word on our concept space, D_k matrix gives us the coordinates of each document in our concept space, and S_k matrix of singular values gives us a clue about how many dimensions or 'concepts'we need to include. Based on these coordinates, documents or terms can be clustered easily using similarity metrics like cosine or others. Also, new concept space has fewer dimensions and the obtained matrix is a dense matrix rather than its previous case which is a sparse. Therefore, LSI is computationally powerful.

One of the drawback of LSI method is that it can not handle polysemy i.e., words with multiple meanings, effectively. It assumes that the same word means the same concept which causes problems for words like 'book'that have multiple meanings depending on which contexts they appear in.

We use LSI for company Wikipedia pages to extract semantically related keywords, and we generate other alternative profile. In order to do that, we construct document-term matrix including Wikipedia company page and each link on that page as documents in the columns of the matrix, and each word appearing in these documents forms the row of the matrix. When we apply LSI to the constructed matrix, the algorithm examines statistical word co-occurence patterns within documents, and discovers semantic structure in a given corpus. As a result, we extract top K keywords (100, 250, and 500), we create Company Wikipedia Page LSI Profile for the corresponding company.

During implementation, we use Python's Gensim package [48] that includes LSI to apply Singular Value Decomposition on the constructed matrix to extract more 'semantically related'keywords with their associated weights. Gensim implementation involves defining the topic number i.e., when the constructed matrix is mapped to the lower dimension, the number of target dimension in the transformed dimension. For our classification task, we obtain the topic number as the default topic number which is defined by Gensim (the default topic number is five). In this way, the five singular value which carries the most important semantic information is obtained while reducing noise in data. This means, we transform our constructed matrix via Latent Semantic Indexing into a latent 5-D space, so that the most significant keywords for each topic are ranked. Then, we obtain the top K keywords (10, 25, and 50) for each topic, and combine them, so Company

Wikipedia Page LSI Profile is generated. For Apple Inc., the most important 5 keywords for Company Wikipedia Page LSI Profile are:

$$V_{profileLSI} = [('macintosh', 0.331), ('iphon', 0.246), ('ipad', 0.177), ('ipod', 0.165), ('model', 0.132)]$$

Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile

We use the same LSI technique for company Wikipedia disambiguation pages. In this case, we construct document-term matrix including company Wikipedia disambiguation page and each link on that page. Similarly, we extract top K keywords (100, 250, and 500), and we create Company Wikipedia Disambiguation Page LSI Profile for the corresponding company.

We obtain the topic number as 5 which is defined by Gensim as default as while generating Company Wikipedia Page LSI Profile. Then, we obtain the top K keywords (10, 25, and 50) for each topic, and combine them, so Company Wikipedia Disambiguation Page LSI Profile is generated.

For Apple Inc., the most important 5 keywords for Company Wikipedia Disambiguation Page LSI Profile are:

$$V_{profileDisambiguationLSI} = [('fruit', 0.346), ('plant', 0.274), ('tree', 0.218), ('band', 0.198), ('flower', 0.167)]$$

Number of alternative meanings

We obtain the number of different links in Wikipedia Disambiguation Page for each of the organization in the dataset in order to see how disambiguation intensity influence the related or unrelated tweet evaluation process. More clearly, the number of different namesakes of a company name would help us if the company name is a very common name or not. This feature would help differentiating between companies that have generic company names like Borders or Delta and companies that have specific company names like Armani or Lennar. We use the number of different namesakes extracted from the Wikipedia disambiguation pages as the number of alternative meanings.

3.2.2.2 Categorical Features

Capital words

Capital words are more likely to be important words or named entity. We assume that, if the tweet contains a company name in capital case, it is more likely that the tweet is related to the company of interest.

Url

URL is also a strong indicator. If the tweet contains a link to a page that is in the same domain as the company homepage, or it has a link to company Wiki-Webpage, then it is more likely that the tweet is related to the company of interest.

Unigram

The rule is that if a tweet contains the full entity name (more than one word) such as "Apollo Hospital", then it is more likely to be tagged as related with the given organization.

Prepositions

The basic English grammar rule is that the prepositions, 'at', 'for' and 'of' commonly come in front of the organization names. Therefore, we define such information as an other feature that would help us whether a tweet refers about a given organization or not.

3.2.2.3 Feature Representation

We represent each tweet using a feature vector compose of the above defined features. For a given tweet T_i and company entity C_k pair, the feature vector is as follows:

$$F_j(T_i, C_k) = [M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, H_1, H_2, H_3, H_4]^T \tag{3.16}$$

The $M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8$ features are similarity features, which quantify how close a tweet overlaps with the entity and alternative profiles. More formally:

M_1 : is computed using the cosine similarity between the tweet feature vector and Company Wikipedia Page Profile.

M_2 : is computed using the cosine similarity between the tweet feature vector and Company Wikipedia Disambiguation Page Profile.

M_3 : is computed using the cosine similarity between the tweet feature vector and Company Review Page Profile.

M_4: is computed using the cosine similarity between the tweet feature vector and Company Wikipedia Page Noun Phrase Profile.

M_5: is computed using the cosine similarity between the tweet feature vector and Company Wikipedia Page Kullback-Leibler Profile.

M_6: is computed using the cosine similarity between the tweet feature vector and Company Wikipedia Page Term Frequency Profile.

M_7: is computed using the cosine similarity between the tweet feature vector and Company Wikipedia Page Latent Semantic Indexing Profile.

M_8: is computed using the Kullback-Leibler distance asymmetric similarity between the tweet feature vector and Company Wikipedia Page Profile.

Also, M_9 is number of alternative meanings feature. Lastly, the H_i features are the categorical features explained in previous section.

3.2.3 Used Supervised Classifiers

3.2.3.1 Logistic Regression

The goal of a classifier in our study is to mark tweets as True or False based on their feature vectors. We train a Logistic Regression for this task. Logistic Regression is a type of probabilistic statistical classification model, which is also used to predict a binary value from a binary predictor. It is used for predicting the outcome of a categorical dependent variable (i.e., a class label) based on one or more predictor variables (features). The principle of Logistic Regression is to find the maximum entropy distribution that is consistent with the given constraints [49]. The idea that just model what is known, and keep uniform distribution for what is unknown, i.e., is have maximal entropy. Weka [50] tool is used to implement Logistic Regression.

Logistic regression uses predicted probabilities in order to label future testing sample like Naive Bayes, J48 Decison Tree[51]. Statistical anlysis shows that, it has better

performance than the other prediction methods. Assuming that we have binary classes 0 and 1,Logistic regression estimates class probabilities directly via the following formula:

$$\Pr((1|(x_0, x_1 \dots x_n), (w_0, w_1 \dots w_n)) = \frac{1}{1 + \exp(-w_0 - w_1 \cdot x_1 - w_2 \cdot x_2 - \dots w_n \cdot x_n)}$$

where $x_1, \dots, x_n$ represent features, and $w_1, \dots, w_n$ represent associated feature weights. The given function produces p values between 0 and 1. Mathematically:

$$\Pr((1|(x_0, x_1 \dots x_n), (w_0, w_1 \dots w_n)) + \Pr((0|(x_0, x_1 \dots x_n), (w_0, w_1 \dots w_n)) = 1$$

Suppose that if $\frac{1}{1+\exp(-w_0-w_1 \cdot x_1-w_2 \cdot x_2-\dots w_n \cdot x_n)} \geq 0.5$ then $y = 1$; otherwise, $y = 0$.

As a parameter estimation method, Logistic Regression uses Maximum-Likelihood Estimation that selects the set of values of the model parameters, which maximizes the likelihood function[52]. We chose to use Logistic Regression as the supervised classifier. We employ Logistic Regression for only numerical features. For example, when we apply logistic algorithm to our training and test data for Wikipedia cosine similarity and review cosine similarity features, the output is here:

$$True_{class} = 2.3791 + cosineValue \cdot 39,205 + reviewCosineValue \cdot 13,5366$$

$$False_{class} = 5.52 \cdot cosineValue - 8.38 \cdot reviewCosineValue + 3,25$$

3.2.3.2 Majority Voting

Majority voting method [12] requires a set of internal classifiers. Majority voting algorithm chooses the particular class based on the largest number of votes or predictions that it receives. Majority rule is a decision rule that selects the alternative which has a majority, that is, more than half the votes.

For this classification problem, we select three different classification algorithms CM_1(X), CM_2(X), and CM_3(X)where X denotes classified sample, and CM_i(X) denotes the i'th classifier in order to classify sample X by Majority Voting algorithm. CM_1(X), CM_2(X), and CM_3(X) show the individually better performance in our classification task. CM_1(X) represents BFTree, CM_2(X) represents MLP, and CM_3(X) represents LADTree.

Majority voting aims to combine these 3 different classifier results such that the produced output is superior to any of these individual models. A common way to combine these rules is to: C(X) = majority voting CM_1(X), CM_2(X), CM_3(X), where C(X) is the

predicted label. Voting Classifier contributes to improve the performance; for instance, if CM_1(X), CM_2(X) classify X correctly, but CM_3(X) classifies incorrectly, there is a 1/3 probability of labeling it as incorrectly. However, the combined classifier will always give the correct classification.

3.2.3.3 Support Vector Machine

Support Vector Machine Algorithm can efficiently perform non-linear classification using a kernel function that, in a way, maps its inputs into high-dimensional feature spaces. A support vector machine constructs a hyperplane in a high dimensional space, which can be used for classification. A good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class. The documents that are on the hyperplane margins are called the support vectors. The new document is classified according to the following simple reasoning:

Documents in the direction of the normal vector are classified as positive; on the other hand, documents in the opposite direction of the normal vector are classified as negative.

Support Vector Machines usually produces good results. However, there is a blackbox problem, i.e., we do not know why it generates good results. Also, its parameters are needed to tuned, and a kernel function that can be linear, quadratic, gaussian or PolyKernel is needed to be chosen. Thus, there is no fix set of parameters for Support Vector Machines. For our classification problem, we employ Support Vector Machine as one of the classification algorithms. We obtain the best accuracy result for our classification task with PolyKernel function when the complexity parameter has a value of 2.8 and the type of the data transformation is normalizing. We find complexity parameter value using Weka's Parameter Selection Classifier, e.g., for a given parameter range, the algorithm selects the best parameter value for the corresponding classifier.

3.2.3.4 Multilayer Perceptron

This algorithm is based on the structure of our brain. Systematically, there are inputs, there is a hidden layer or there are hidden layers, and there are outputs. This algorithm is an effective algorithm, because, if the actual output is not equal the desired output, the error value is aimed to be minimized without any external effects. The algorithm minimizes the error value by updating weights with "Backpropagation Algorithm".

If there is no any hidden layer, the algorithm is called Perceptron Algorithm. The input-output representation is here:

$$f(x_1 \cdot w_1 + x_2 \cdot w_2 + ... + bias) = \text{output}$$

where x_i denotes features of input sample and w_i denotes associated sample weight.

f is a monotonically increasing, continuous function i.e., it's derivative is always positive. The function f is as follows:

$$f(x) = \text{sigmoid function} = \frac{1}{1+\exp(-x)}$$

In contrast, if there is a hidden layer or multiple hidden layers, the algorithm is called "Multilayer Feedforward Neural Network". It uses Backpropagation Algorithm [53] that minimizes the error by taking derivative of the error function. Backpropogation algorithm is prone to stuck in a local minumum. In order to avoid stucking in a local minumum, the optimization algorithms like simulated annealing and genetic algorithm is used.

We will present the problem mathematically. Since backpropagation uses the gradient descent method, the derivative of the squared error function with respect to the weights of the network should be calculated [53]. Assuming one output neuron, the squared error function is:

$$E = \frac{1}{2} \cdot (t - y)^2$$

where, E is the squared error, t is the target output for a training sample, and y is the actual output of the output neuron. The factor of $\frac{1}{2}$ is cancelled the exponent, when differentiating. So, error E depends on the output y. However, the output y depends on the weighted sum of all its input:

$$y = \sum_{i=1}^{n} w_i \cdot x_i$$

Here, n is the number of input units to the neuron, w_i is the i'th weight and x_i is the i'th input value to the neuron.

The given formula in above is only true for a neuron with a linear activation function i.e., that is the output is simply the weighted sum of the input. In general, a non-linear, differentiable activation function, α, is used. Thus, more correctly:

$$y = \alpha(net)$$

$$net = \sum_{i=1}^{n} w_i \cdot x_i$$

The partial derivative of an Error Function is computed with respect to w_i using the chain rule:

$$\frac{\partial E}{\partial w_i} = \left(\frac{dE}{d_y} \cdot \frac{d_y}{d_n et} \cdot \frac{\partial net}{\partial w_i} \right)$$

where $\frac{\partial E}{\partial w_i}$ denotes how the error changes when the weights are changed, $\frac{dE}{d_y}$ denotes the error change when the output is changed, $\frac{d_y}{d_n et}$ denotes how the output changes in the case of changing weighted sum, and $\frac{\partial net}{\partial w_i}$ denotes how the weighted sum changes as the weights change.

As the weighted sum net is the sum over all products $w_i \cdot x_i$, the partial derivative of the sum with respect to a weight w_i is the the corresponding input x_i. Similarly, the partial derivative of the sum with respect to an input value x_i is the weight w_i:

$$\frac{\partial net}{\partial w_i} = x_i$$
$$\frac{\partial net}{\partial x_i} = w_i$$

The derivative of the output y with respect to the weighted sum net is the derivative of the activation function α:

$$\frac{d_y}{d_{net}} = \frac{d}{d_{net}} \cdot \alpha$$

For that reason backpropagation algorithm chooses the activation function as to be differentiable. A commonly used activation function is the logistic function:

$$y = \frac{1}{1 + e^{-z}}$$

which has a derivative of:

$$\frac{dy}{dz} = y(1 - y)$$

When the network uses a logistic activation function, the derivative of the output y with respect to the weighted sum net is the same as the derivative of the logistic function:

$$\frac{dy}{dnet} = y(1-y)$$

Finally, the derivative of the error E with respect to the output y is:

$$\frac{dE}{dy} = \frac{d}{dy}\frac{1}{2}(t-y)^2$$

$$\frac{dE}{dy} = y - t$$

Combinatorially,

$$\frac{\partial E}{\partial w_i} = \left(\frac{dE}{d_y} \cdot \frac{d_y}{d_n et} \cdot \frac{\partial net}{\partial w_i}\right)$$

$$\frac{\partial E}{\partial w_i} = ((y-t) \cdot y \cdot (1-y) \cdot x_i)$$

To update the weight w_i using gradient descent, the error derivative function is multiplied by learning rate α

$$\Delta w_i = -\alpha \frac{\partial E}{\partial w_i}$$
$$\Delta w_i = \alpha(t-y)\alpha' x_i$$

For a linear neuron, the derivative of the activation function α is 1, which yields:

$$\Delta w_i = \alpha(t-y)x_i$$

This is the delta rule for perceptron learning. In backpropagation and perceptron learning, when the output y equals the desired output t, the change in weight Δw_i would be zero, which is exactly the desired case.

For our tweet classification problem, we employ Multilayer Perceptron as one of the classification algorithms that is considered by Majority Voting Classifier. We obtain the number of hidden layers as 1, the value of the learning rate α as 0.24 and the number of

the training time as 1000. We find these numerical values using Trial and error approach. We repeat varied numbers to reach the best accuracy value. With these numerical values, the perceptron algorithm give the best result.

3.2.3.5 Decision Tree

Decision tree[54] is one of the classification algorithms that is easy to understand and interpret. When a new item comes, the algorithm starts to check from the top of the tree to match the item's criteria against the node's criteria. In order to construct a decision tree, starting from the top node, the splitting attribute is determined for each iteration. Splitting attribute is determined mostly based on "Information Gain"[55] that tells us how important a given attribute of the feature vectors is. In other words, the splitting attribute is the one that has the highest Information Gain. Basically, Information Gain is computed using entropy that measures the degree of uncertainty in the group of samples.

In order to prevent Overfitting that occurs when the algorithm memorizes the training data rather than learning a tree, the stopping criteria is determined while the tree is constructed. If the entropy of the pair of the nodes is greater than common parent node, the splitting process is stopped.

The sample that is predicted is known as dependent variable, because, its label will be determined based on other attributes' values.

For our classification problem, we employ LADTree Decision Tree [8] as one of the classifier that shows good performance, and is considered by Majority Voting Classifier. Since LADTree is a binary classifier, the algorithm can distinguish between positive and negative samples successfully. The basic assumption of LAD model is that a binary point covered by some positive patterns, but not covered by any negative pattern is positive, and similarly, a binary point covered by some negative patterns, but not covered by positive pattern is negative. For a given data set, LAD model builds large a set of patterns and selects a subset of them which satisfies the above assumption.

Also, we use BFTree Decision Tree [9] as one of the classifiers that is regarded by Majority Voting Classifier. BFTree shows considerable performance that we represent in the next chapter. BFTree uses "best first approach", while constructing a tree. BF tree constructs binary trees, i.e., each internal node has two outgoing edges. The tree growing method attempts to maximize within-node purity. The "best"node is the node whose split leads to maximum reduction of impurity (e.g. Gini index [56] or information gain) among all nodes available for splitting. For our classification task, we use Gini Index in order to determine the best splitting attribute for BFTree.

3.2.3.6 Adaboost Algorithm

Boosting [57] is an approach in Machine learning based on the idea of constructing a highly accurate prediction rule by merging many relatively weak and inaccurate rules. In other words, AdaBoost is an algorithm for constructing a strong classifier as linear combination of simple weak classifiers $h_t(x)$:

$$f(x) = \sum_{t=1}^{T} \alpha_t \cdot h_t(x)$$

where t represents the training time, and α represents the learning rate.

Most boosting algorithms include weak learner classifiers which are only slightly correlated with the true classification, and combine them in to a final strong classifier. When weak classifiers are combined, they are typically weighted in some way that is usually related to the weak classifiers' accuracy. After a weak classifier is added, the data is re-weighted: examples that are misclassified gain weight, and examples that are classified correctly lose weight. Thus, future weak classifiers focus more on the examples that previous weak classifiers missed.

Unlike Neural Networks and Support Vector Machine, the AdaBoost training process selects only features that are known to improve the predictivity of the model. The algorithm reduces dimensionality, since irrelevant features do not need to be computed. In this way, the algorithm improves the execution time. Boosting is a meta-algorithm that reduces bias in supervised learning. We utilize Adaboost algorithm for both numerical and categorical features.

3.2.3.7 Simple Approach Algorithm

In this experiment, we apply the following simple approach. After doing some preprocessing techniques, a company profile vector is generated using company Wikipedia web page. Then, top 100 key words that have the highest tf.idf values are obtained.

Then, we assume that if a tweet includes one of the company profile vector term, a tweet would be relevant with a given company; otherwise it would be irrelevant.

In [26], the similar approach is used in order to select restaurant related tweets from tweet corpus. First, they pick the top occuring words in the reviews as keyword set, then they eliminate a tweet that do not contain any of those keywords from tweet collection. Using this approach, they generate restaurant related tweet data set that they use for Entity Matching problem.

3.2.3.8 Entity Ranking Algorithm

We use a general method to classify given tweet as related, or unrelated. For this purpose, we use language model which is proposed in [12], and we previously explain this method in Chapter 2. The model incorporates the entity information and general review language model. For our problem, the goal of this mixture model is to determine whether a given tweet is relevant or irrelevant with a corresponding company. Informally, when a tweet t is written about a company, each word in the t is drawn either from an entity (company) information, or generic review language.

In our adaptation, we use two language models: one is entity mention language and the other is review language model. Our review language model includes all review data for training and testing companies. Also, entity mention language consists of two profiles: one of the profiles consists of Wikipedia company keywords, and the other profile consists of Wikipedia disambiguation company keywords. For this algorithm, the general approach is here:

For a given tweet, each word in the tweet is chosen with α probability from entity mention language and (1- α) probability from review language model. Since we have two profiles, the algorithm computes the value for both of them based on the probability values of entity mention language and review language model. Then, the algorithm assigns the tweet as 'true', or 'false'depending on the returned greatness of the profile values. More specifically, if the obtained value of the Wikipedia disambiguation profile is lower than the Wikipedia company profile, the tweet is labelled as 'true', otherwise it is labelled as 'false'. We illustrate the algorithm for our problem in 3.4.

Now, we will explain the visualized system mathematically:

P_{e_1} : The company profile vector that includes relevant preprocessed keywords with the company,

P_{e_2} : The company profile vector that includes irrelevant preprocessed keywords with the company,

P_r : The review profile vector that includes both training and testing company review preprocessed keywords,

P_t : The tweet profile vector that includes preprocessed tweet keywords,

where P_{e_1} and P_{e_2} show our entity mention language models, and P_r denotes our general language model.

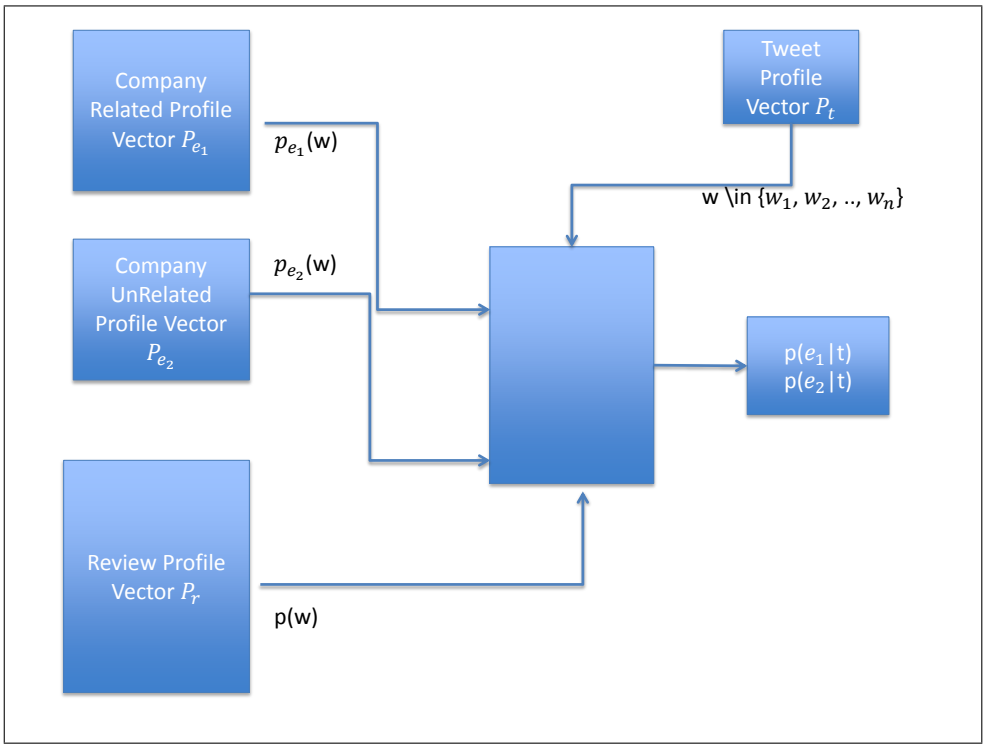

FIGURE 3.4: Entity Ranking Algorithm Representation as Visually

For a given tweet profile vector P_t, each word in P_t is generated independently. With probability α, a word is chosen with probability $p_{e_1}(w)$ from P_{e_1} or with probability $p_{e_2}(w)$ from P_{e_2}, and with probability $1-\alpha$, a word is chosen with probability $p(w)$ from P_r. For simplicity, we obtain α as 0.5. When we formalize that, we need to compute $p_{e_1}|t$ and $p_{e_2}|t$ such that

$$p_{e_1}|t = \sum_{w \in P_t}^{P_t} \log\left(1 + \frac{\alpha}{1-\alpha} \cdot \frac{p_{e_1}(w)}{p(w)}\right)$$

$$p_{e_2}|t = \sum_{w \in P_t}^{P_t} \log\left(1 + \frac{\alpha}{1-\alpha} \cdot \frac{p_{e_2}(w)}{p(w)}\right)$$

where

$$p_{e_1}(w) = \frac{\log(\frac{1}{f_w})}{\sum\limits_{w' \in P_{e_1}}^{P_{e_1}} \cdot \log(\frac{1}{f_{w'}})}$$

$$p_{e_2}(w) = \frac{\log(\frac{1}{f_w})}{\sum\limits_{w' \in P_{e_2}}^{P_{e_2}} \cdot \log(\frac{1}{f_{w'}})}$$

where f_w represents the frequency of the word in P_r. Lastly, we have:

$$p(w) = \frac{c(w, P_r) + 1}{\sum_{w' \in P_r}^{P_r} c(w', P_r) + |V|}$$

where $c(w, P_r)$ represents the frequency of w in P_r, and $|V|$ represents the vocabulary size. Based on the computed probabilities $p_{e_1}|t$ and $p_{e_2}|t$ firstly, we come to the conclusion that:

If $p_{e_1}|t \ > p_{e_2}|t$

then a tweet is relevant with a given company and labelled as 'true'.

If $p_{e_1}|t \ < p_{e_2}|t$

then a tweet is irrelevant with a given company and labelled as 'false'.

If $p_{e_1}|t \ = p_{e_2}|t$

then we do not know whether a tweet is relevant or irrelevant with a given company and labelled as 'unknown'.

Here, we do not evaluate 'unknown'tweets. Since tweet content is very limited and has many grammatical errors, many tweets do not overlap with both P_{e_1} and P_{e_2}. Also, the limited keyword set in P_{e_1} and P_{e_2} (as explained below) induce non-overlapping results. Table 4.18 in the next chapter confirms this observation. In this case, $p_{e_1}(w)$ and $p_{e_2}(w)$ have the value of 0. Therefore, a considerable number of tweet have been labelled as 'unknown'. As we do not have any evaluation criteria that measures the accuracy of the unknown tweets, we use different approach for the zero probability tweets. We assume that if the calculated probability equals to 0 for $p_{e_1}(w)$ and $p_{e_2}(w)$, the tweet will be labelled as 'false'. Here are the main reasons for this assumption.

The sources to gather company related keywords are better than the sources for unrelated keywords. With the help of Web crawling techniques, we select the most related keywords about a given company in the profile. However, P_{e_2} has infinite keyword set. In Entity Ranking Experiment, since we do not have infinite keyword set, P_{e_2} is limited to 100, 250 and 500 keywords. Thus, these keyword set values are very insufficient. Also, the sources for collecting irrelevant keywords are not clear as collecting related keyword set. That is P_{e_2} may include relevant keywords about a company that would lead to erroneous results. Therefore, we had to limit the number of P_{e_2} with 100, 250, and 500.

As a result, we assume that in the case of having 0 probability for both profiles, since the tweet do not overlap with P_{e_1} which includes the most relevant keywords about a given company, the probability of being irrelevant outweighs the probability of being relevant.

We generate P_{e_1} for Company Wikipedia Page Profile, Company Wikipedia Page Noun Phrase Profile, Company Wikipedia Page Kullback-Leibler Profile, Company Wikipedia Page Term Frequency Profile, and Company Wikipedia Page Latent Semantic Indexing Profile with different keyword sizes. Whereas, we construct P_{e_2} for Company Wikipedia Disambiguation Page Profile, Company Wikipedia Disambiguation Page Noun Phrase Profile, Company Wikipedia Disambiguation Page Kullback-Leibler profile, Company Wikipedia Disambiguation Page Term Frequency Profile, and Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile with the same sizes of keywords that we use for company related profiles. We enlarge P_r corpus including all review keywords that are obtained from training and testing review company profile vectors. Then, we apply the given algorithm. The results are discussed in the next chapter.

3.2.3.9 Threshold Classification

We also develop and study a simple classification technique by learning a threshold value from training data set. The value of similarity features represents how much company profiles overlaps with a given tweet. The algorithm learns the threshold value which gives the best accuracy on the training data. Then, this threshold value is used for unseen testing data. if the cosine similarity of a test tweet is under this threshold value, the tweet is assumed to be unrelated to the corresponding company. Otherwise the tweet is assumed to be related.

Initially, while learning the best threshold that gives the most accurate results, we extract Wikipedia company profile keyword set including the most important top 100,250,500, and 1000 keywords. We perform Threshold learning approach with all those different keyword sets, and we get the best result with top 100 most significant keywords of the company profile. Therefore, we do all other threshold experiments using company profile consisting of top 100 important keywords.

We use the Threshold Learning algorithm for the following profiles: Company Wikipedia Page Profile, Company Review Page Profile, Company Wikipedia Page Noun Phrase Profile, Company Wikipedia Page Kullback-Leibler Profile, Company Wikipedia Page Term Frequency Profile, Company Wikipedia Page Latent Semantic Indexing Profile, and Company Wikipedia Disambiguation Page Profile. In order to learn the threshold value, we use two similarity techniques, namely, Cosine similarity and Kullback-Leibler Divergence asymmetric similarity. The similarity explanations are presented next and the learned threshold values and performance values are presented in the next chapter.

- **Cosine Similarity**

  Cosine Similarity is one of the most popular similarity measures applied to text documents for Information Retrieval applications and clustering. Given two documents that are represented as term vectors, their similarity may be computed as the correlation between term vectors. The cosine similarity of any pair of vectors may be computed by taking their dot product, and dividing it by the product of their norms. Mathematically, given two document term vectors $\vec{a}^t$ and $\vec{b}^t$, their cosine similarity is:

$$SIM_{cosine}(\vec{a}^t, \vec{b}^t) = \left(\frac{\vec{a}^t \cdot \vec{b}^t}{|\vec{a}^t| \cdot |\vec{b}^t|} \right)$$

  where $\vec{a}^t$ and $\vec{b}^t$ are m dimensional vectors over the term set T. The cosine similarity is non-negative and bounded between [0,1]. If the value of the cosine similarity between two documents is 1, then these two documents are identical. In contrast, when the value is 0, than these two documents are totally different.

- **Kullback-Leibler Divergence Asymmetric Similarity**

  We define Kullback-Leibler Divergence above, and we present its mathematical equations (by 3.15). Here, we use Kullback -Leibler Distance to compute document similarity between company profile vector and a tweet vector. A basic property of Kullback-Leibler distance is its asymmetry. In other words, Kullback-Leibler distance between documents P and Q is not equal to the distance between Q and P. Therefore, we need to compute the KL-divergence twice. Formally, given two documents d_1 and d_2, after computing $D_{KL}d_1||d_2$ and $D_{KL}d_2||d_1$ based on the formula given above in eq.(3.15), we obtain the average of the two computed results, and assign the resulting value as the document similarity between two documents. As Kullback-Leibler distance does not always give results between 0 and 1, we conduct normalization to pull values between 0 and 1. In order to do that, we use the following mathematical function:

$$normalized_{value} = e^{-average}$$

  Kullback Leibler distance has different interpretation than Cosine Similarity. If Kullback-Leibler distance between two documents equals 0, then the documents are identical, and the similarity decreases when the value moves away from 0. Therefore, while computing threshold value based on training data, we assume that if threshold value is less than Kullback-Leibler distance, the tweet will be labeled as 'true'; otherwise, it will be labelled as 'false'. Similarly, for unseen tweet data, we assume that a tweet which has a similarity value that is greater than the

learned threshold are labelled as 'false'. This study is carried out only on Wikipedia company profiles. The results are presented in the next section.

Chapter 4

Experiments and Evaluation

4.1 Dataset Description

We use the data set of WePS-3 evaluation campaign that is a well-recognized international competition, and was held in 2010. In WePS-3, the problem of Web entity search are proposed. The competition include two tasks, one is focused on the problem of person name ambiguity, and the second task is related to Online Reputation Management for organizations. The second task focus on company name ambiguity. In this study, we deal with the second task.

Online Reputation Management (ORM) [58] Task consists of filtering tweet posts containing a given company name, and deciding whether a post belongs to the company or not. ORM consists of monitoring media, analysing what people say about an entity; also if necessary, contact with customers. This is because, negative comments on online media can seriously affect the reputation of a company. Therefore, popular company brands want to analyse the content of tweets in order to improve marketing strategies. However, when the entity name is ambiguous, filtering out the tweets is a very challenging task. ORM does this task by using Twitter posts; because, it is a critical source of real time reputation management, and it has a little context and no privacy criteria.

In the data set, the set of organization names are different in the training and testing portions of the data. For each organization in the data set, there is a company name and its home page URL. For each tweet, there exists a tweet identifier, the entity name, query, the author identifier, and the tweet content.

The trail corpus consists of 23 company names (17 English and 6 Spanish organizations) each of them has 100 tweets. In the training and testing datasets, there are 52 and 48 companies respectively. The companies are chosen from the DBpedia [59] which

is a knowledge-base that extracts structural information from Wikipedia pages. The automatic filter that detects company names match the common names, so it ensures the ambiguity of the company name.

The training and testing corpora have been annotated by Mechanical Turk Workers that enables employers to hire online workers for short-term tasks that computers don't do well. These workers annotated the tweets that mention the company apparently after determining whether each tweet mentions the company or not. The "true"label means that the tweet is associated to a company, whereas the "false"one means that the tweet is not relevant to a given company, and the unknown label indicates that the annotators are unable to make a decision. In our experiments, we do not consider the unknown labels.

4.2 Evaluation Metrics

In this part, the evaluation metrics that are commonly used in Information Retrieval are explained. *Precision* is the fraction of retrieved instances that are relevant, *recall* is the relevant instances that are retrieved. In other words, precision is the measure of the quality demonstrating that the algorithm returns relevant terms more than irrelevant terms; recall is the measure of the quantity demonstrating that the algorithm returns the most of relevant results. For example, for a text search on a set of documents, precision is the number of correct results divided by the number of all returned results, and for text search on a set of documents, recall is the number of correct results divided by the number of results that should have been returned.

In the classification task, some terms are used in order to compare classification results under test with external judgments. The terms "positive"and "negative"show the classifier's prediction, while "true"and "false"terms define whether the prediction corresponds to the judgment result. These terms are explained below:

True Positive (TP) : Tweets that are correctly labeled as belonging to the positive class.

True Negative (TN) : Tweets that are correctly labeled as belonging to the negative class.

False Positive (FP) : Tweets that are labeled as belonging to the positive class incorrectly.

False Negative (FN) : Tweets that are labeled as belonging to the negative class incorrectly.

We use the following metrics to study the performance of our classification process.

$$\begin{aligned}
\text{Accuracy} &= (TN + TP)/(TN + TP + FN + FP) \\
\text{Precision}^{+} &= TP/(TP + FP) \\
\text{Precision}^{-} &= TN/(TN + FN) \\
\text{Recall}^{+} &= TP/(TP + FN) \\
\text{Recall}^{-} &= TN/(TN + FP) \\
F^{+}_{measure} &= 2 \cdot Precision^{+} \cdot Recall^{+}/(Precision^{+} + Recall^{+}) \\
F^{-}_{measure} &= 2 \cdot Precision^{-} \cdot Recall^{-}/(Precision^{-} + Recall^{-})
\end{aligned}$$

4.3 Experiments

4.3.1 Experiment 1: Bag-of-Words Experiment

Initially, a simple baseline algorithm is designed using weighted bag of keywords from Twitter. These tests are performed on the testing data set. We perform leave-one-out cross validation. We train a support vector machine on the 47 of these companies and test it on the last company. We repeat this for all of the 48 companies. The performance of the classifier is shown in Table 4.1. This is our baseline experiment.

TABLE 4.1: Support Vector Machine on Testing Data only Using Bag of Weighted Tweet Keywords (Baseline).

Experiment	**Accuracy**
Baseline	59.7%

4.3.2 Experiment 2: Feature Extraction

As seen in Table 4.1, simple bag of words approach is not appropriate for the task of disambiguation. The results support our belief in the importance of utilizing external features. As we discuss in the previous chapter, we use both numerical and categorical features. Firstly, we will represent how numerical features contribute to the classification task both individually and combined with others.

TABLE 4.2: The abbreviations used for features and algorithms.

Name	Abbreviation
Wikipedia Cosine Similarity	$wiki_{cosineSim}$
Wikipedia Disambiguation Cosine Similarity	$disambig_{cosineSim}$
Kullback-Leibler Divergence Asymmetric Similarity	$kldiv_{Sim}$
Review Cosine Similarity	$review_{cosineSim}$
Wikipedia Kullback-Leibler Divergence Cosine Similarity	kl_{Sim}
Wikipedia Term Frequency Similarity	$termfreq_{Sim}$
Wikipedia Latent Semantic Indexing Similarity	$latent_{Sim}$
Number of Alternative Meanings	numberofmeanings
Capital name	*capital*
Url	url
Preposition	$being_{prep}$
Unigram	unigram
Multilayer Perceptron	M.P.
Logistic Regression	L.R.
Majority Voting	M. V.
J48 Decision Tree	J48
DecisionStump Decision Tree	Dstump
LADTree Decision Tree	LADTree
BFTree Decision Tree	BFTree
Adaboost Algorithm	Adaboost
Support Vector Machine	SVM
Entity Ranking Algorithm	E. R. A.
Simple Approach Algorithm	S. A. A.
Threshold Algorithm	T. A.
Wikipedia Two Profile Algorithm	W.T.P.A.
Naive Bayes Algorithm	N. Bayes
Attribute Selected Classifier	Attribute S. C.

Table 4.2 shows the abbreviations of features and algorithms that are used for both feature extraction and other experiments.

TABLE 4.3: Experiments with Selected Classifiers that give the best accuracy on numerical attributes.

Used Feature	**Experiment**	**Accuracy**
$wiki_{cosineSim}$	M. P.	62.22%
$disambig_{cosineSim}$	LADTree	62.23%
$kldiv_{Sim}$	SVM	51.2%
$review_{cosineSim}$	L. R.	56.6%
kl_{Sim}	LADTree	59.7%
$termfreq_{Sim}$	Adaboost	61.1%
$latent_{Sim}$	M. P.	62.1%
$numberofmeanings$	Dstump	55.1%
$wiki_{cosineSim}$ + $disambig_{cosineSim}$	L. R.	62%
$wiki_{cosineSim}$ + $kldiv_{Sim}$	Adaboost	62.2%
$wiki_{cosineSim}$ + $review_{cosineSim}$	L. R.	61.8 %
$wiki_{cosineSim}$ + $numberofmeanings$	Adaboost	64%
$wiki_{cosineSim}$+kl_{Sim}	M.P.	62.9%
$wiki_{cosineSim}$+$termfreq_{Sim}$	BayesNet	63%
$wiki_{cosineSim}$+$latent_{Sim}$	Adaboost	63%
$disambig_{cosineSim}$ +$kldiv_{Sim}$	SVM	51.8%
$disambig_{cosineSim}$ +$review_{cosineSim}$	J48	55.6%
$disambig_{cosineSim}$ +$numberofmeanings$	Adaboost	55.1%
$kldiv_{Sim}$+$review_{cosineSim}$	Adaboost	55.7%
$disambig_{cosineSim}$+kl_{Sim}	LADTree	60.6%
$disambig_{cosineSim}$+$termfreq_{Sim}$	Adaboost	63%
$disambig_{cosineSim}$+$latent_{Sim}$	LADTree	62.1%
kl_{Sim} + $kldiv_{Sim}$	SVM	59.2%
$termfreq_{Sim}$ + $kldiv_{Sim}$	Adaboost	61.1%
$latent_{Sim}$+ $kldiv_{Sim}$	Adaboost	62.1%

TABLE 4.4: Experiments with Selected Classifiers that give the best accuracy on numerical attributes.

Used Feature	**Experiment**	**Accuracy**
$numberofmeanings$ +kl_{Sim}	LADTree	61.3%
$numberofmeanings$ +$termfreq_{Sim}$	LADTree	62.9%
$numberofmeanings$+$latent_{Sim}$	LADTree	63.1%
$wiki_{cosineSim}$ + $numberofmeanings$+$disambig_{cosineSim}$	M. V.	64.59 %
$wiki_{cosineSim}$ + $disambig_{cosineSim}$+$kldiv_{Sim}$	SVM	61%
$wiki_{cosineSim}$ + $disambig_{cosineSim}$+$review_{cosineSim}$	M. P.	62.47%
$disambig_{cosineSim}$+$review_{cosineSim}$+$kldiv_{Sim}$	Adaboost	55.1%
$wiki_{cosineSim}$ + $kldiv_{Sim}$+ $review_{cosineSim}$+$numberofmeanings$	L. R.	61.5%
$wiki_{cosineSim}$ + $review_{cosineSim}$+ kl_{Sim}+$termfreq_{Sim}$+$latent_{Sim}$	M. P.	63.3%
$wiki_{cosineSim}$ + $disambig_{cosineSim}$ + $kldiv_{Sim}$ + $review_{cosineSim}$	Adaboost	62.2%
$wiki_{cosineSim}$ + $disambig_{cosineSim}$+ $numberofmeanings$ +$review_{cosineSim}$	M. P.	62.6%
kl_{Sim}+$termfreq_{Sim}$ + $latent_{Sim}$ + $kldiv_{Sim}$	SVM	61%
$wiki_{cosineSim}$+$disambig_{cosineSim}$+$review_{cosineSim}$+ kl_{Sim}+$termfreq_{Sim}$+$latent_{Sim}$	Adaboost	64.3%
$wiki_{cosineSim}$+$disambig_{cosineSim}$+$review_{cosineSim}$+ kl_{Sim}+ $termfreq_{Sim}$+ $latent_{Sim}$+$kldiv_{Sim}$	SVM	62.5%
$wiki_{cosineSim}$+$disambig_{cosineSim}$+$review_{cosineSim}$+ kl_{Sim}+$termfreq_{Sim}$+$latent_{Sim}$+$numberofmeanings$	BFTree	64.95%
$wiki_{cosineSim}$ + $disambig_{cosineSim}$+kl_{Sim}+ $termfreq_{Sim}$+$latent_{Sim}$+$kldiv_{Sim}$+ $review_{cosineSim}$ +$numberofmeanings$	Adaboost	64%

As you see in Table 4.4, $kldiv_{Sim}$ shows the worst performance individually when we compare with the other features. As for $wiki_{cosineSim}$, generally it gives the best results both individually and when it combines with other numerical features.

Although $disambig_{cosineSim}$ produces comparable results both individually and when it comes together with $wiki_{cosineSim}$, kl_{Sim}, $termfreq_{Sim}$, and $latent_{Sim}$; in the case of combining with other features except $wiki_{cosineSim}$, kl_{Sim}, $termfreq_{Sim}$, and $latent_{Sim}$, the accuracy becomes low. This shows that in order to make classification task accurately, this feature should be used with either one of the following features: $wiki_{cosineSim}$, kl_{Sim}, $termfreq_{Sim}$, $latent_{Sim}$ or combination of those features.

$review_{cosineSim}$ individually does not give comparable results as $wiki_{cosineSim}$, kl_{Sim}, $termfreq_{Sim}$, $latent_{Sim}$, and $disambig_{cosineSim}$. When it combines with $wiki_{cosineSim}$, the performance increases by 5%. When $review_{cosineSim}$ comes together with both $wiki_{cosineSim}$ and $disambig_{cosineSim}$, the performance increases 6% and the best accuracy is obtained by Multilayer Perceptron algorithm (62.47%). Beside, when we use all related similarity features, Multilayer Perceptron algorithm gives the best performance by 63.3%. Moreover, when we use all similarity features except $kldiv_{Sim}$, we obtain the best result by Adaboost algorithm (64.3%). This shows that 1% increase arise from $disambig_{cosineSim}$. However, when we use all similarity features, the best accuracy is obtained by Support Vector Machine algorithm (62.5%). It is clear that 1.8% decrease arise from the inefficiency of $kldiv_{Sim}$.

Since $kldiv_{Sim}$ makes a bad contribution to the classification process individually, when it is used with $wiki_{cosineSim}$, the classification accuracy decreases slightly (0.02). Results also show that $kldiv_{Sim}$ does not work well with $disambig_{cosineSim}$ as $wiki_{cosineSim}$. When $disambig_{cosineSim}$ and $kldiv_{Sim}$ are combined, the accuracy performance becomes 51.8% which is 10.5% below than the accuracy value which is obtained by $disambig_{cosineSim}$ individually. In addition, $kldiv_{Sim}$ decreases the classification performance 0.5% when it is used with $review_{cosineSim}$. In the case of bringing together $kldiv_{Sim}$, $review_{cosineSim}$ and $disambig_{cosineSim}$, the best performance is obtained by Adaboost (55.1%) algorithm. This result is slightly below than the obtained result by the association of $kldiv_{Sim}$ and $review_{cosineSim}$.

Also, $kldiv_{Sim}$ decreases the classification accuracy 0.7% when it is employed with kl_{sim}. However, when $kldiv_{Sim}$ combines either $termfreq_{Sim}$ or $latent_{Sim}$, the accuracy performance does not change. This shows $kldiv_{Sim}$ works better with $termfreq_{Sim}$ and $latent_{Sim}$ rather than other features. As we mention in the above paragraph, if all similarity features are employed, the best performance becomes 62.5%. It is clear that $kldiv_{Sim}$ decreases the accuracy value by 1.8%, since the accuracy value becomes 64.3% when all similarity features are used except $kldiv_{Sim}$. Also when $kldiv_{Sim}$ is used with kl_{Sim}, $termfreq_{Sim}$, and $latent_{Sim}$, the obtained accuracy becomes 61%. Therefore, in general, $kldiv_{Sim}$ has either negative effect or no effect to the our classification task both individually and when it is used with other similarity features.

The other important feature is that $numberofmeanings$. As you see from the table, $numberofmeanings$ does not achieve a high classification accuracy individually. Nevertheless, when it is combined with $wiki_{cosineSim}$, the performance increases dramatically. As this feature shows the intensity of the company ambiguity and $wiki_{cosineSim}$ shows the document similarity, their co-occurrence makes a great contribution to the classification task. Results show that these two features show the best performance by Adaboost algorithm. The obtained accuracy value is 64%. As you see from the following tables and explanations, this result is very close to the obtained best accuracy when we use all features. Similarly, when three of the features namely $numberofmeanings$, $wiki_{cosineSim}$, and $disambig_{cosineSim}$ come together, the best performance is provided by Majority Voting classifier (64.59%) that encapsulates Support Vector Machine, Multilayer Perceptron, and LADTree. $disambig_{cosineSim}$ increases the performance by approximately 0.5%. The success of these three classifiers may be interpreted as follows: the document similarity, the document dissimilarity and the name ambiguity value are important features for our evaluation task.

In addition, when $numberofmeanings$ is used with one of the following features: kl_{Sim}, $termfreq_{Sim}$, or $latent_{Sim}$, it has a considerable effect to our classification task in comparison to its single effect. Moreover, when we use $numberofmeanings$ with all similarity features except $kldiv_{Sim}$, the best accuracy is obtained by BFTree (64.95%). As you see from the table, this value is higher by approximately 0.6%, when we use all similarity features except $kldiv_{Sim}$. However, $kldiv_{Sim}$ decreases accuracy value as 0.95%, in the case of using with other features i.e., similarity features and the name ambiguity feature.

In spite of the fact that $kldiv_{Sim}$ shows the worst accuracy among numerical attributes, when it is used with other features, $kldiv_{Sim}$ does not decrease the accuracy significantly. We think the reason is that the decrease in accuracy is resulted by using with other features is balanced when it is used with $termfreq_{Sim}$ or $latent_{Sim}$. Even then, it does not contribute anything.

TABLE 4.5: Experiments with Selected Classifiers that give the best accuracy on categorical attributes

Used Feature	**Experiment**	**Accuracy**
capital	LADTree	51.8%
url	M. P.	51.2%
unigram	SVM	53.2%
$being_{prep}$	Adaboost	51%
All categorical features	N. Bayes	55.1%

Table 4.5 shows the performance of categorical features both individually and combined with other features. As you see from the table, unigram seems to show the best contribution, and other categorical features except unigram improve the performance approximately 2%. Actually, we expect that these categorical features would help to improve our classification accuracy both individually and combined with other features. Surprisingly, these features show the low accuracy. As a reason, our data set might not be suitable for these extracted features. As we have mention before, the tweet data has many grammatical errors. Since tweet text is informal, users make their tweet text without worrying about spell checking. Generally, users post their tweets with the purpose of chatting. For a user, sending of his-her message instantly is the most important step rather than the correctness of the content, so tweet contents has many abbreviations and statements that is very challenging for others to understand. Therefore, our categorical features that are mostly based on users do not show the efficient performance for Twitter data.

For example, we assume that, if a tweet includes the company name with capital letters, it is more likely that it is relevant to a particular company. Because of the reasons that we mention, Twitter users use company names with lower cases frequently. Therefore, this feature might not be good measure for Twitter data set. In principle, company names take one of the prepositions respectively "for", "of", or "at". This feature should have shown a lot better performance than 51%. Nevertheless, as tweet users do not pay attention to grammatical rules, most of the company names are used without any preposition. We believe that if we used these categorical features for normal data set that have more clear content, we might have obtained more satisfactory results.

TABLE 4.6: Experiment results with both numerical and categorical features

Experiment	**Accuracy**
M. P.	62.4%
AdaBoost	64.03%
BFTree	64.3%
LADTree	64.54%
M. V. (LADTree+BFTree)	64.1%
M. V. (LADTree+Adaboost)	64.5%
M. V. (LADTree+M.P.)	64.8%
M. V. (Adaboost+M. P.+LADTree)	64.9%
M. V. (BFTree+M. P.+LADTree)	65.7%
M. V. (Adaboost+LADTree+BFTree)	64.5%

Table 4.6 shows performance of some of the classifiers that show the best performance for our classification task. In this case, we use all numerical and categorical features. As you see from Table 4.6, firstly we show classifiers that produce the best performance individually. Then, we obtain several combinations of these classifiers that are included by Majority Voting. We get the best score using the selected classifiers respectively BFTree (gini index is used as splitting criteria, prepruning strategy is used as pruning strategy, and the number of folds is obtained as 6.), Multilayer Perceptron (learning rate = 0.28, training time = 1000 and number of hidden layers = 1), and LADTree. Although, Multilayer Perceptron does not provide good accuracy individually, when Multilayer Perceptron and BFTree come together with LADTree, they contribute to the classification accuracy noticeably. In contrast, Adaboost does not improve the classification accuracy as LADTree does as when it is used with BFTree and Multilayer Perceptron. Moreover, as you see in 4.6, when Adaboost is used with other classifiers by Majority Voting, it does not affect to the classification task.

TABLE 4.7: Experiment results with selected numerical and categorical features

Selected Features	**Experiment**	**Accuracy**
$wiki_{cosineSim}$+$numberofmeanings$+ $termfreq_{Sim}$+$latent_{Sim}$+unigram	Attribute S. C. (LADTree)	64.57%
$wiki_{cosineSim}$+$numberofmeanings$+ $termfreq_{Sim}$+$latent_{Sim}$+unigram	Attribute S. C. (M.V.)	65%

As seen in Table 4.7, we perform Attribute Selection in order to see which features are more valuable for the classification task. In order to do that, we select our features using the correlation feature selection (CFS) [60] criteria that is a kind of filtering method to choose feature subsets. The Correlation Feature Selection measure evaluates subsets of features on the basis of the principle that good feature subsets contain features that are highly correlated with the classification, but uncorrelated with each other. As search approach, we use best first that use greedy hill climbing, which iteratively evaluates a candidate subset of features, then modifies the subset, and evaluates whether the new subset is an improvement over the old or not.

Weka selects 5 attributes as selected attributes that show the best performance in the classification process. As you see from Table 4.7, these are respectively $wiki_{cosineSim}$, $numberofmeanings$, $termfreq_{Sim}$, $latent_{Sim}$, and unigram. With all these selected feature combinations, we perform additional experiments. We apply Majority Voting on these features with previously selected classifiers, namely, Multilayer Perceptron, BFTree, and LADTree. As you see from the table, the obtained accuracy with these features is

65%. When we compare to the Table 4.6, this shows other features make approximately 0.7% contribution to the overall classification task. Another observation based on this table is that, when we ignore other seven features, LADTree provides a slight improvement.

Table 4.8 presents abbreviations of used company profiles and metrics for the following experiments.

TABLE 4.8: The abbreviations used for company profiles and metrics.

Name	**Abbreviation**
Company Wikipedia Page Profile	WP
Company Wikipedia Disambiguation Page Profile	WDP
Company Review Page Profile	RP
Company Wikipedia Page Noun Phrase Profile	WNPP
Company Wikipedia Page Kullbeck-Leibler Profile	WKLP
Company Wikipedia Page Term Frequency Profile	WTFP
Company Wikipedia Page Latent Semantic Indexing Profile	WLSP
Company Wikipedia Disambiguation Page Noun Phrase Profile	WDNPP
Company Wikipedia Disambiguation Page Kullbeck-Leibler Profile	WDKLP
Company Wikipedia Disambiguation Page Term Frequency Profile	WDTFP
Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile	WDLSP
Precision for Positive Examples	P^+
Precision for Positive Examples	P^-
Recall for Positive Examples	R^+
Recall for Negative Examples	R^-
F Measure for Positive Examples	F^+
F Measure for Negative Examples	F^-

4.3.3 Experiment 3: Threshold Algorithm

We perform threshold experiments with several company profiles. The obtain results are presented in Table 4.9.

In this table, the threshold is learned from the cosine similarity value of the training data. The algorithm learns the threshold value which gives the best accuracy performance on the training data, and this threshold value is used for unseen testing data. if the cosine similarity of a test tweet is under this threshold value, the tweet is assumed to be unrelated to the corresponding company. Otherwise the tweet is assumed to be related. We perform Threshold learning approach with several different keyword sets i.e., 100,250,500, and 1000 for Company Wikipedia Page Profile, and we get the best result with top 100 most significant keywords of the profile. Therefore, we do all other threshold experiments using company profile consisting of top 100 important keywords. We did not try to obtain results with other sets of keyword for other profiles.

Table 4.9: Threshold Experiment.

Used Profile	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
RP (thr = 0.003)	61.3%	60.7%	42.5%	50%	66%	80.3%	72.5%
WNPP (thr = 0.001)	66.6%	66%	40.5%	50.3%	66.7%	85.3%	74.8%
WKLP (thr = 0.004)	67%	61.3%	56.5%	58.8%	70.5%	74.5%	72.4%
WTFP (thr = 0.005)	68.7%	67.5%	48%	56%	69%	83.4%	75.6%
WLSP (thr = 0.002)	69.9%	73.6%	43.3%	54.6%	68.6%	88.8%	77.4%
WP (thr = 0.001)	71%	69%	51%	59%	70%	84%	76%

As you see from the Table 4.9, as oppose to our expectation, review pages are not good sources to overlap with a tweet vector. Statistical analysis show that, both true negative and positive values decrease in comparison with other profiles. One of the primary reasons is that review pages mostly include terms that belong to daily language. This indicates that the learned cosine value between a tweet vector and a review profile vector might become higher than the learned cosine value with Company Wikipedia Noun Phrase Profile and Company Wikipedia Page Profile (the learned threshold with those companies is 0.001). Thus, our false negative values increase noticeably (approximately 600 more tweets are labelled as false negative when we compare with the Company Wikipedia Page Profile). This aspect influences true negative tweets in a negative way, too. This factor lowers measured precision(-), recall(-) and fmeasure(-). The other significant factor is that because of review pages comprise mostly of daily statements,

the majority of company profile vector keywords are keywords that are less relevant to the company. For that reason, missed significant keywords lessen true positive tweets at a considerable level. Since our true positive and negative values decrease, the accuracy of the review profile decreases dramatically.

The Accuracy-Threshold graph (4.1) for training data is represented for Company Review Page Profile. In this graph, the threshold is learned from training data, which is 0.003.

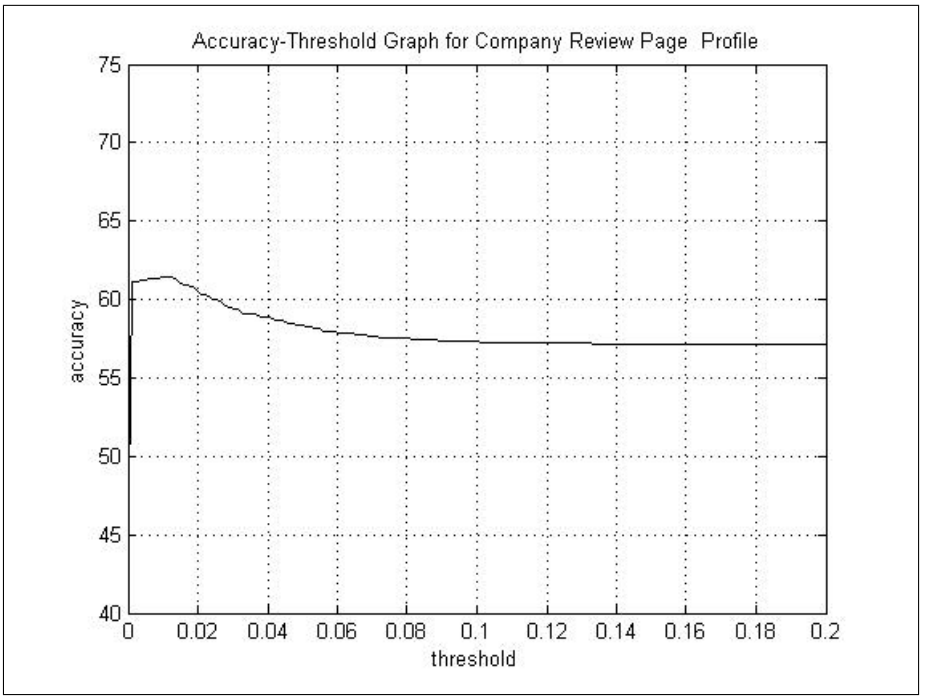

FIGURE 4.1: Accuracy-Threshold Graph for training data that represents the best threshold value for Company Review Page Profile.

Table 4.9 also shows that, the accuracy of Company Wikipedia Page Noun Phrase Profile is less than the Company Wikipedia Page Kullback-Leibler Profile, Company Wikipedia Page Term Frequency Profile, Company Wikipedia Page Latent Semantic Indexing Profile, and Company Wikipedia Page Profile. With this profile, our false negative values as high as Company Review Page Profile. This means many tweets are labeled as false incorrectly. That shows our learned threshold value from training data is rather higher for testing data. This proves the fact that since our training and testing tweet data set include different company tweets, the overlapping problem may occur, and this problem may challenge the classification task. As we mention in the previous section, our Company Wikipedia Page Noun Phrase Profile consists of keywords that are only noun or noun phrase. We exclude verb part of speeches from this profile. This seems to affect our positive tweets negatively. Thus, in comparison with our Company Wikipedia Page Profile, the number of true positive tweets decreases approximately by 750.

The Accuracy-Threshold graph (4.2) for training data is represented for Company Wikipedia Page Noun Phrase Profile. In this graph, the threshold is learned from training data, which is 0.001.

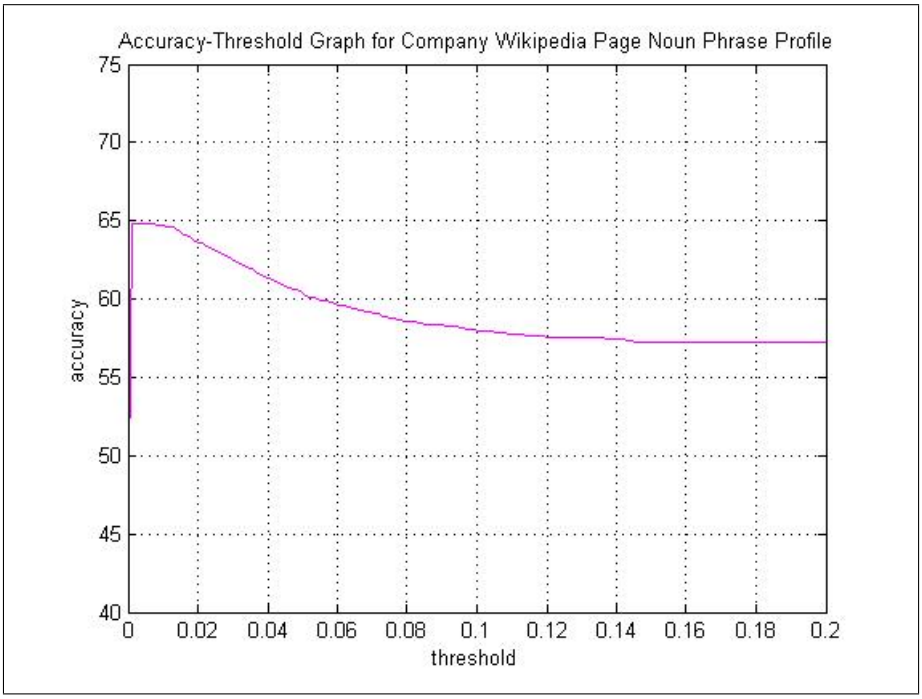

FIGURE 4.2: Accuracy-Threshold Graph for training data for Company Wikipedia Page Noun Phrase Profile.

Table 4.9 clearly shows that Company Wikipedia Page Kullback-Leibler Profile and Company Wikipedia Page Term Frequency Profile have similar performance values. As you see in 4.9, the learned threshold value is higher when we compare to other profiles. The reason is that, as we mention in the previous section, the most important term in term profile vector for both profiles has a weight of 1, which is higher than the weight of the most significant keyword of other profiles, so this factor increases the learned threshold value noticeably. Also, we see that Company Wikipedia Page Term Frequency Profile has higher recall value for negative examples. This shows our false positive values are higher for Wikipedia Page Term Frequency Profile than Company Wikipedia Page Kullback-Leibler Profile. The Accuracy-Threshold graphs (4.3 and 4.4) are represented for Company Wikipedia Page Kullback-Leibler Profile and Wikipedia Page Term Frequency Profile. The learned threshold value from company training data for Company Wikipedia Page Kullback-Leibler Profile equals 0.004, and the learned threshold value from company training data for Company Wikipedia Page Term Frequency Profile is 0.005.

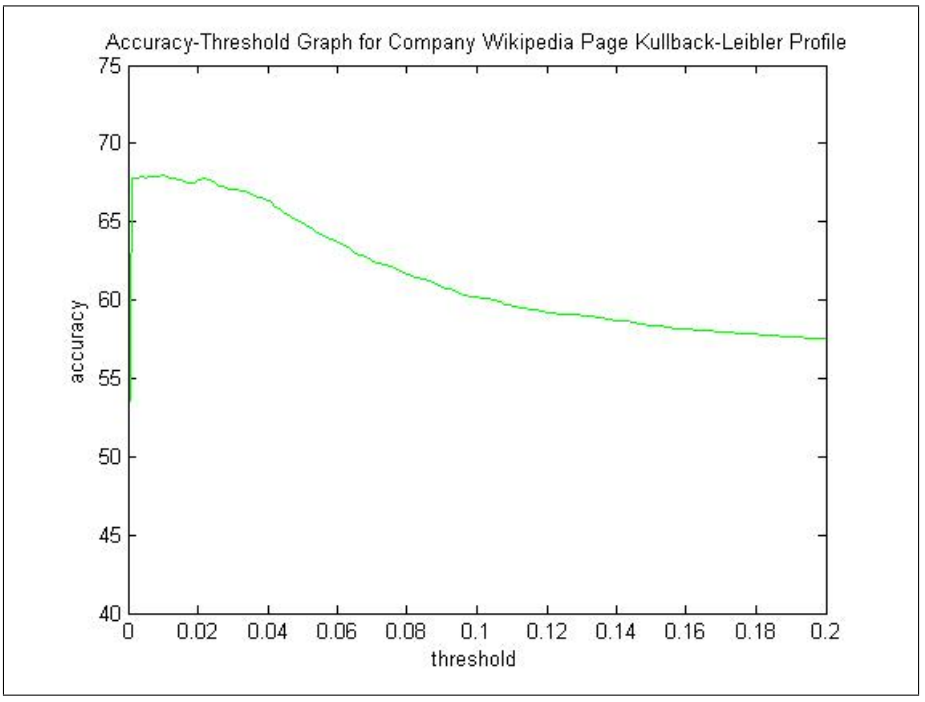

FIGURE 4.3: Accuracy-Threshold Graph for training data for Company Wikipedia Page Kullback-Leibler Profile.

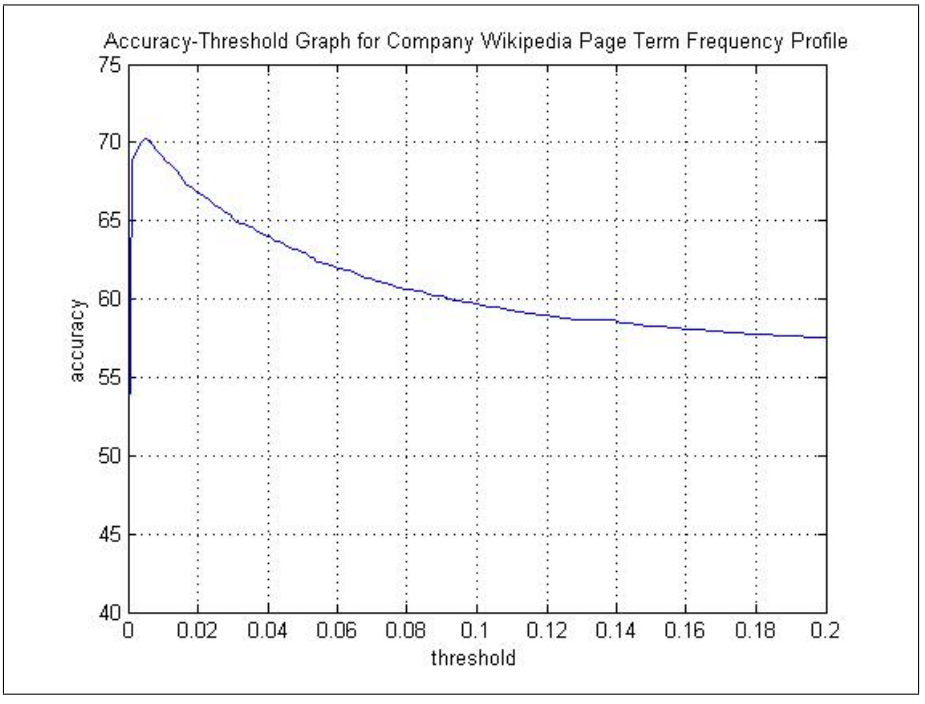

FIGURE 4.4: Accuracy-Threshold Graph for training data for Company Wikipedia Page Term Frequency Profile.

As you see from in Table 4.9, when we use Company Wikipedia Page Latent Semantic Indexing Profile for threshold classification, we obtain comparable results with Company Wikipedia Page Profile. As we mention in the previous chapter, LSI is computationally

powerful in order to find semantically related significant keywords among document corpus. Therefore, we obtain satisfactory performance results with Company Wikipedia Page Latent Semantic Indexing Profile. With this profile the obtained true positives are lower than the number of true positive values which are obtained by Company Wikipedia Page Profile, so the accuracy value is lower than Company Wikipedia Page Profile. Also, when we use Company Wikipedia Page Latent Semantic Indexing Profile, we obtain higher false negative values in comparison to Company Wikipedia Page Profile; thus, we have lower $recall^{+}$. More clearly, higher false negative values mean that i.e., the learned threshold value for company training data is high for test company set of data. The Accuracy-Threshold graph (4.5) for training data is represented for Company Wikipedia Page Latent Semantic Indexing Profile. In this graph, the threshold is learned from training data, which is 0.002.

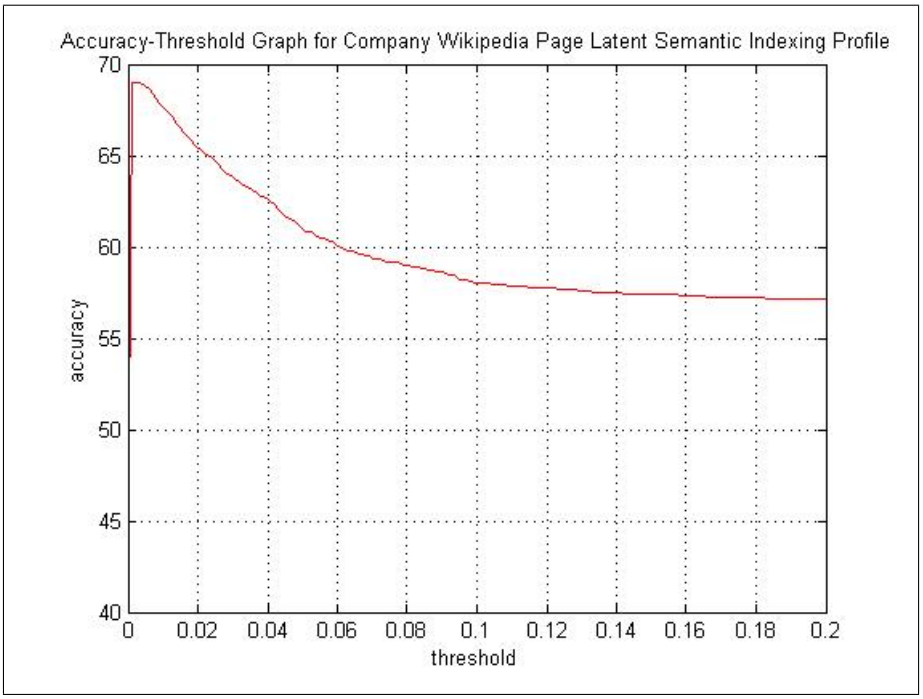

FIGURE 4.5: Accuracy-Threshold Graph for training data for Company Wikipedia Page Latent Semantic Indexing Profile.

As seen in Table 4.9, we obtain the best performance with Company Wikipedia Page Profile including keywords that are both nouns and verbs. Since, Wikipedia content is more formal, the profile vector includes keywords that are more related to a corresponding company. As Wikipedia content is frequently checked for misspellings, the Wikipedia content is cleaner that makes profile vector clearer as well. All these factors influence the classification accuracy of the Company Wikipedia Page Profile positively. With this profile, our true positive values increase noticeably. Also, the algorithm is very successful in finding true negative examples, too. Since the number of our false positive

values is low, our negative recall value is as high as the obtained negative recall value with Company Wikipedia Page Noun Phrase Profile.

The Accuracy-Threshold graph (4.6) for training data is represented for Company Wikipedia Page Profile. In this graph, the threshold is learned from training data, which is 0.001.

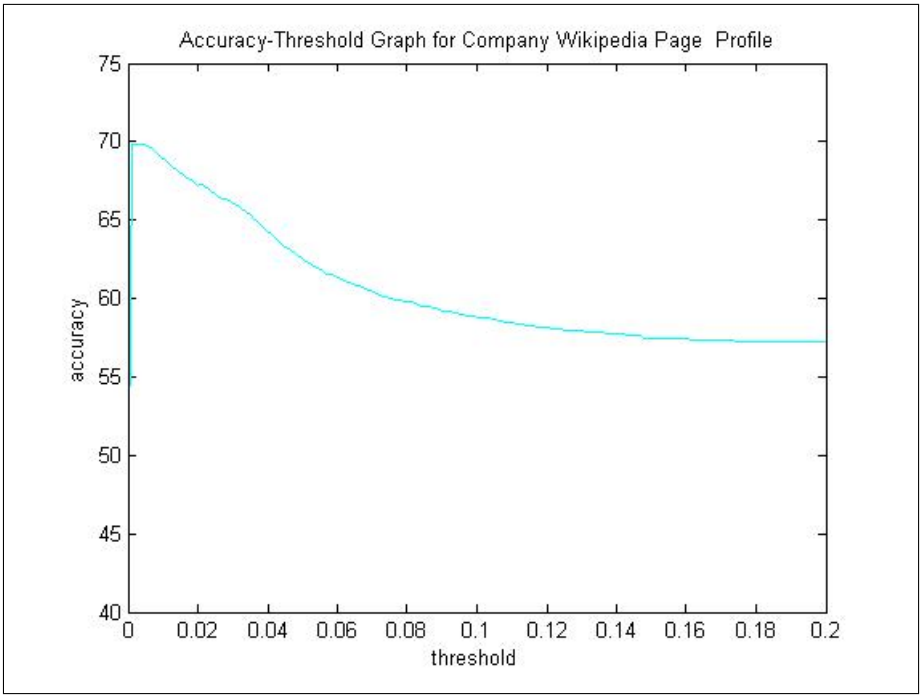

FIGURE 4.6: Accuracy-Threshold Graph for training data for Company Wikipedia Page Profile.

In summary, we can make the general following comment about our threshold experiment. Our method tends to label tweets as unrelated for all given profiles because our threshold is high. Because of this, we obtain high values for precision, recall, and f-measure with negative examples in comparison to the positive examples.

Table 4.10 shows the result of threshold experiment using only Company Wikipedia Page Profile. In this case, Kullback Leibler distance asymmetric similarity is used as to be learned threshold value. The learned threshold value from training data is 0.58. For this similarity, Wikipedia Company Page Profile shows the best performance which is approximately 58%. As you see from the previous table, Kullback-Leibler distance shows bad performance when it is compared with the cosine similarity. As other company profiles have similar decrease, we do not need to show the similarity performance obtained by those profiles in Table 4.10.

TABLE 4.10: Experiment Results with Kullback Leibler Distance.

Used Profile	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
WKLP	58.5%	96.2%	0.67%	1.3%	58.4%	99.9%	73.7%

4.3.4 Experiment 4: Wikipedia Two Profile Approach

In this experiment, additionally, we compute the cosine value between Company Wikipedia Disambiguation Page Profile and each company tweet. We assume that for a given tweet if the cosine similarity between the company Wikipedia profile vector and a tweet vector is greater than the cosine similarity between the Wikipedia disambiguation profile vector and a tweet vector, the tweet will be marked as related to the given company; otherwise, the tweet will be labeled as unrelated to the given company. In the case of being the equal cosine value, the tweet is labeled as unknown. However, we do not evaluate the unknown tweets. The obtained results are shown by Table 4.11.

TABLE 4.11: Experiment Results with using Company Wikipedia Page Profile and Company Wikipedia Disambiguation Page Profile

Used Profile	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
WP + WDP	70.65%	70.3%	85.6%	77.2%	71.3%	49.8%	58.7%

4.3.5 Experiment 5: Simple Approach Algorithm

Simple Approach Algorithm results are shown in Table 4.12.

TABLE 4.12: Simple Approach Algorithm Results using Wikipedia Company Profile

Used Profile	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
WCP	69.8%	66.7%	55%	60.3%	71.3%	80.3%	75.6%

Although this approach is very simple, the results are amazing. With this approach, because of the increase in the true positive tweets, the number of positive examples are higher than the previous experiments. Also, the number of true negative examples are as high as the previous experiments. As evident from the table, the accuracy of this algorithm is better than the more complicated algorithms shown in Table 4.6. This proves the fact that in data analysis, sometimes, the simpler approach produces more valuable results. The reason can be explained with bias-variance dilemma [61].

TABLE 4.13: Entity Ranking Classification Results for Company Wikipedia Page Noun Phrase Profile and Company Wikipedia Disambiguation Page Noun Phrase Profile for different keyword sets.

Keyword Set	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
(100,100)	64.9%	62.2%	40.8%	49.2%	65.9%	82.2%	73.2%
(250,100)	**65.2%**	59.1%	54.1%	56.5%	69%	73.2%	71%
(500,100)	62%	53.7%	64.5%	58.6%	70.3%	60.2%	64.8%
(250,250)	64.6%	58.8%	50.8%	54.5%	67.9%	74.5%	71%
(500,500)	62.9%	55.8%	54.4%	55%	67.9%	69.1%	68.5%

TABLE 4.14: Entity Ranking Classification Results for Company Wikipedia Page Kullback-Leibler Profile and Company Wikipedia Disambiguation Page Kullback-Leibler Profile for different keyword sets.

Keyword Set	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
(100,100)	**67.8%**	65.7%	48%	55.5%	68.8%	82%	74.8%
(250,100)	66.1%	59%	61.4%	60.2%	71.5%	69.5%	70.5%
(500,100)	61.1%	52.6%	68.6%	59.6%	71.3%	55.8%	62.6%
(250,250)	67.2%	61.69%	56.4%	58.9%	70.6%	74.8%	72.6%
(500,500)	63.3%	55.7%	59.5%	57.5%	69.5%	66.1%	67.8%

As the model becomes more complex, the model learns the training data very well, so the error of the training data decreases. However, this case is very undesirable for unseen testing data. Since, the model memorizes the training data, it will not generalize the training pattern for unseen testing data that leads to the higher variance. As a result of that, the predictions of the model will be less accurate. In contrast, models with higher bias tend to be relatively simple and generalizable for unseen testing data. As explained, our more complex algorithms might not have learned the general pattern as necessary, that causes the low performance results. Probably, this fact might have been valid for our threshold algorithm too.

4.3.6 Experiment 6: Entity Ranking Algorithm

For this algorithm, we use different profile vectors with different keyword sets. In tables (4.13 through 4.17), the keyword sets are represented as tuples. The first element of the tuple denotes the number of relevant keywords obtained from P_{e_1}, and the second element of the tuple represents the number of irrelevant keywords obtained from P_{e_2}. The obtained highest accuracy value is written with bold face for each profile.

As seen from the tables, the best accuracy is obtained with Company Wikipedia Page Latent Semantic Indexing Profile and Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile including 250 keyword set for P_{e_1} and 250 unrelated keywords

TABLE 4.15: Entity Ranking Classification Results for Company Wikipedia Page Profile and Company Wikipedia Disambiguation Page Profile for different keyword sets.

Keyword Set	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
(100,100)	**67.9%**	67%	45.4%	54.1%	68.2%	84%	75.3%
(250,100)	67.2%	60.7%	59%	59.8%	71.2%	72.7%	71.9%
(500,100)	64.4%	56%	68.3%	61.6%	73.1%	61.6%	66.8%
(250,250)	65.8%	59.9%	54.4%	57.1%	69.4%	73.9%	71.6%
(500,500)	65.3%	58.2%	60.2%	59.2%	70.8%	69%	69.9%

TABLE 4.16: Entity Ranking Classification Results for Company Wikipedia Page Term Frequency Profile and Company Wikipedia Page Term Frequency Profile for different keyword sets.

Keyword Set	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
(100,100)	65.4%	60.9%	47.9%	53.6%	67.6%	77.9%	72.4%
(250,100)	**66.3%**	59.6%	59.9%	59.8%	71.2%	70.9%	71%
(500,100)	61.8%	53.2%	69.2%	60.2%	71.9%	56.5%	63.3%
(250,250)	64.8%	58.1%	56.2%	57.1%	69.4%	71%	70.2%
(500,500)	62.1%	54%	60.8%	57.2%	69.2%	63%	65.9%

TABLE 4.17: Entity Ranking Classification Results for Company Wikipedia Page Latent Semantic Indexing Profile and Company Wikipedia Disambiguation Page Latent Semantic Indexing Profile for different keyword sets.

Keyword Set	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
(100,100)	67.7%	63.2%	54.1%	58.3%	70.2%	77.4%	73.6%
(250,100)	66.9%	60.1%	61.6%	60.9%	72%	70.7%	71.4%
(500,100)	66%	58.2%	65.4%	61.6%	72.8%	66.4%	69.5%
(250,250)	**69.1%**	64.5%	58.1%	61.1%	71.9%	77.1%	74.4%
(500,500)	68.3%	62.1%	61.3%	61.7%	72.5%	73.2%	72.9%

for P_{e_2}(4.17). As we mention in the previous section, Latent Semantic Indexing algorithm succeeds to find highly relevant keywords among all documents in a corpus using Singular Value Decomposition method. The algorithm selects the most correlated words considering all Wikipedia document corpus. Since we have higher true negative samples and lower false positive examples, our recall value for negative examples is higher for (250,250) keyword set.

As you see from tables, 4.13 and 4.16, company Wikipedia noun phrase profiles and company Wikipedia term frequency profiles reach the best accuracy with (250,100) keyword set. Since Company Wikipedia Page Noun Phrase Profile and Company Wikipedia Disambiguation Page Noun Phrase Profile have restricted keyword set, the best overlap between a tweet profile and a company related entity profile could be provided by enriching P_{e_1}. For P_{e_2}, 100 keyword set seems to be sufficient to find irrelevant tweets

without needing to enlarge irrelevant keyword set. Also, we may make the same comments for Company Wikipedia Page Term Frequency Profile and Company Wikipedia Disambiguation Page Term Frequency Profile.

Our Wikipedia profile vectors and Wikipedia Kullback Leibler profile vectors get the best accuracy with values (100,100) (Tables 4.15 and 4.14). We observe from the tables that when the keyword set increases, the performance of the algorithm decreases considerably. That shows that the increase in keyword set reduces the quality of the keyword set in company profile vectors P_{e_1} and P_{e_2}. That leads to the inaccurate probabilistic computation that affects the performance of the algorithm negatively.

It can be noted that as you see from the Entity Ranking formula given in the previous chapter, the Entity Ranking algorithm penalizes any word which exists in review language model frequently. On the other hand, the obtained probability will be higher if a word in a given tweet comes from entity profile. The major novel side of the algorithm is here. With this aspect, this algorithm is more efficient than the standard tf-idf approach. Suppose that for a given tweet, being relevant or irrelevant will be determined depending on the summation of the idf value of all terms in the tweet across documents. If the word "iphone"and "go"have equal idf values, both of these words will have equal weights for Apple Inc. company. However, "iphone"is an entity mention word that should have had higher weight more than word "like". Therefore, the approach of Entity Ranking Algorithm is more capable than such standard approaches.

4.3.7 General Evaluation

Table 4.19 shows average accuracy measure per company for different classifiers. Statistically significant improvement of classifiers over baseline are indicated by bold face. As seen from the table, except Wikipedia Two Profile Experiment (Using Company Wikipedia Page Profile and Company Wikipedia Disambiguation Page Profile), our results are statistically significant over our baseline algorithm.

Table 4.20 shows the performance of WePS-3 participants. Considering Threshold Algorithm that gives the best result as Our Proposed Method, it gets an accuracy of 71% which is lower than only LSIR and ITC-UT systems. Also, our other approaches, namely, Majority Voting, Entity Ranking Algorithm, Simple Approach Algorithm, and Wikipedia Two Profile Algorithm outperform than other systems, except LSIR and ITC-UT.

Our method tends to label tweets as unrelated because our threshold is high. Because of this, we obtain high values for precision, recall, and F-measure with negative examples.

TABLE 4.18: Number of Non-Overlapping Tweets using Wikipedia company Latent Semantic Indexing profiles for each Company in the Testing Data Set.

Company Name	**# Tweet**	**(100,100)**	**(250,100)**	**(500,100)**	**(250,250)**	**(500,500)**
Amazon	483	223	192	176	161	136
Apache	500	374	336	312	251	212
Apple	493	150	125	116	117	103
A. S. Roma	499	362	332	316	268	257
Blizzard Ent.	463	314	303	296	198	189
Camel	500	421	407	389	408	383
Canon	500	232	218	204	209	188
Cisco Systems	496	212	178	166	168	152
Cisco Systems	499	212	178	166	168	152
Cvs/pharmacy	477	367	327	320	310	297
Denver Nuggets	498	307	281	288	255	215
Deutsche Bank	465	125	117	112	53	51
Emory University	496	203	182	161	169	146
Fox Channel	499	252	219	188	201	162
Friday's	498	438	380	356	355	323
Gibson	469	367	345	332	325	298
GM	435	279	171	158	159	140
Jaguar Cars Ltd.	499	371	338	314	309	263
Johnnie Walker	497	11	11	9	10	9
JFK Airport	498	309	259	239	260	237
Kiss Band	500	430	406	379	390	338
Lexus	405	39	37	30	33	29
Liverpool FC	500	260	20	19	18	17
Lloyds Banking Group	473	26	26	25	23	22
Mtv	466	333	289	224	268	205
Macintosh	385	185	162	159	156	140
Mcdonald's	500	4	4	3	4	3
Mclaren Group	500	338	298	284	256	250
Metro Supermarket	401	263	237	212	240	204
A.C. Milan	500	229	209	191	217	197
Muse Band	500	308	256	233	245	219
Oracle	496	197	178	162	178	161
Orange	499	386	353	329	361	331
Paramount Group	453	379	355	351	323	309
Scorpions	500	9	9	6	9	6
Seat	443	361	346	297	341	289
Sharp Corporation	475	387	358	335	344	318
Sonic.net	474	427	406	387	415	392
Sony	396	84	71	67	73	69
Starbucks	445	288	278	242	282	243
Subway	500	324	302	289	292	280
Tesla Motors	500	320	265	241	269	231
Us Airways	471	385	361	334	357	323
Virgin Media	469	372	345	330	341	330
Yale University	498	243	214	187	208	179
Zoo Entertainment	478	392	381	357	351	318

Results show that Threshold Classification technique achieves good results with cheap computational cost.

TABLE 4.20: All System Results.

System	Accuracy	P^+	R^+	F^+	P^-	R^-	F^-
LSIR	%**83**	%71	%74	%**63**	%**84**	%52	%56
ITC-UT	%75	%75	%54	%49	%74	%6	%57
Our Proposed Method	%71	%69	%51	%59	%70	%**84**	%**76**
SINAI	%63	%**84**	%37	%29	%68	%**71**	%53
UVA	%56	%47	%41	%36	%6	%64	%55
KALMAR	%46	%48	%**75**	%47	%65	%25	%28

4.3.8 T-test Results

In order to measure the significance of accuracy change on experiments, we perform statistical t-test [62]. The independent samples t-test is used to test the hypothesis that the difference between the means of two samples is equal to 0 (this hypothesis is therefore called the null hypothesis). We conduct a paired t-test that looks at the difference between paired values in two, and takes into account the variation of values within each sample, and produces a single number known as a t-value. Since we do not know the mean of the two samples, we obtain the tail number as 2. When the obtained t-test value is less than the level of significance (traditionally chooses as 0.05), the null hypothesis is rejected and the conclusion is that the two means differ significantly. If the p-value associated with the t-test is small than the significance level, there is evidence to reject the null hypothesis. In other words, there is evidence that the means are significantly different at the significance level reported by the p-value. If the p-value associated with the t-test is not small than the significance level, there is not enough evidence to reject the null hypothesis, and we can conclude that there is evidence that the means are not different.

If the significant level is set at 0.05, it means that the rejection region comprises 5% of the sampling distribution. This 5% can be allocated to one side of the sampling distribution as in a one-tailed test or partitioned to both sides of the distribution as in a two-tailed test, with each tail (or rejection region) containing 2.5% of the distribution.

As we mention, the significant level usually is predetermined as 0.05. However, this value may change depending on the application. An informal interpretation of a p-value, based on a significance level of about 10%, might be:

If $p < 0.01$

then very strong presumption against null hypothesis.

If $0.01 < p < 0.05$

then strong presumption against null hypothesis.

If $0.05 < p < 0.1$

then low presumption against null hypothesis.

If $p > 1$

then no presumption against the null hypothesis.

We obtain the significance level as 0.06, and we measure t value between company samples (for each company, the obtained accuracy value for the corresponding experiment is considered.) for each experiment pair. T-test results for the evaluated method pairs are shown in Table 4.21. According to the stated significant value, statistically significant values are written as bold face.

TABLE 4.21: T-test Results for Experiment pairs

Experiment pairs	**p-value (significant level = 0.07)**
T. E.-S. A. A.	0.6
T. E.-Baseline	**0.0026**
T. E.-M. V.	0.1
T. E.-E. R. A. (WCP is obtained).	0.4
T. E.-E. R. A. (LSP is obtained).	0.7
T. E.-W. T. P. A.	0.7
Baseline-M. V.	**0.58**
Baseline-W. T. P. A.	0.32
Baseline.-S. A. A.	**0.021**
Baseline-E. R. (WCP is obtained)	**0.02**
Baseline-E. R. (LCP is obtained)	**0.017**
M. V.-E. R. A. (WCP is obtained)	0.2
M. V.-W. T. P. A.	0.6
M. V.-S. A. A.	0.2
M. V.-S. A. A.	0.2
M. V.-E. R. A. (LSP is obtained)	**0.01**
S. A. A.-E. R. A. (WCP is obtained)	0.6
S. A. A.-E. R. A. (WCP is obtained)	0.6
S. A. A.-W. T. P. A.	0.9
S. A. A.-E. R. A. (LSP is obtained).	0.2
E. R. A. (WCP is obtained)-E. R. A. (LSP is obtained)	0.24
E. R. A. (WCP is obtained)-W. T. P. A.	0.99
E. R. A. (WCP is obtained)-W. T. P. A.	0.99
W. T. P. A. + E. R. A. (LSP is obtained)	0.56

TABLE 4.19: Average Accuracy Measure along with p-values for Different Classifiers. Statistically significant improvement of classifiers over baseline are indicated by bold face. (t-test $p < 0.06$)

Company Name	**Baseline**	**M.V.**	**E. R. A.**	**S. A. A.**	**W.T.P. A.**	**T. A.**
Amazon.com	30.1%	36.5%	46.8%	40.8%	60%	38.1%
Apache	55.4%	62.2%	65.1%	75.5%	86.2%	80.3%
Apple	41.7%	61.4%	76.4%	71.1%	96.6%	77.1%
A. S. Roma	73.3%	77.8%	83.4%	79.6%	58.4%	78.5%
Blizzard E.	72.1%	80%	81.1%	81.667%	91.7%	84.7%
Camel	76.4%	83.3%	88.6%	93.1%	52.7%	90.2%
Canon	52.8%	64.4%	61.5%	71.7%	92.8%	80.5%
Cisco S.	30.3%	44.5%	63.6%	65%	78.5%	58.9%
CVS/P.	27.6%	34.4%	32.2%	37.9%	72.7%	37.3%
Denver N.	67.3%	69.1%	64.9%	73.7%	24.6%	75%
Deutsche B.	44.3%	59.5%	88.2%	70.9%	69.1%	74.3%
Emory U.	51.1%	50.6%	59.5%	59.2%	54.8%	59.7%
Fox C.	46.4%	50%	57.4%	60%	34.1%	60.5%
Friday's	75.3%	75.2%	50%	50%	62.5%	81.3 %
Gibson	85.2%	87.1%	85.4%	86.6%	80.6%	89.9%
GM	54.5%	64.5%	81%	57.2%	75.7%	60.5%
Jaguar Cars	63%	70.2%	60.8%	65.8%	88.1%	81.5%
Johnnie W.	74.5%	74.6%	80.1%	79%	78.4%	79.7%
John F. K. A.	47.4%	51.4%	64.5%	69.2%	68.2%	68.9%
Kiss Band	78.4%	80.8%	85.9%	85.2%	54.8%	84.1%
Lexus	84%	92.7%	86.4%	85.4%	85.2%	26.9%
Liverpool FC	64.4%	61.6%	69.1%	68.4%	55%	67.5%
Lloyds B. G.	73.8%	74.8%	62.6%	11.3%	27.9%	80.3%
Mtv	26.2%	35.5%	31.2%	37.4%	80.9%	37.4%
Macintosh	44.5%	53.7%	72.2%	65.3%	84.7%	63.2%
Mcdonald's	20.2%	29.2%	50%	50%	81.6%	45.4%
Mclaren G.	60.7%	66.3%	67.5%	66.5%	49.4%	69.1%
Metro S.	90.7%	87.8%	78.7%	84.2%	68.2%	88.8%
A.C. Milan	57%	56.2%	59.6%	63.6%	45.2%	66%
Muse Band	37.8%	44.4%	62.5%	56%	77%	57%
Oracle	47.7%	57.5%	68.2%	65%	91.1%	73.3%
Orange	88.4%	85.3%	81.7%	84.7%	54.9%	89%
Paramount G.	81.4%	78.5%	82.7%	76	69.5%	83.5%
Scorpions	40.3%	51.2%	71%	60.7%	82.5%	63.3%
Seat	88.5%	88.7%	85.7%	89.1%	53%	81.7%
Sharp C.	82.3%	83.3%	86.3%	85.1%	64.2%	84.8%
Sonic.net	83.8%	84.5%	89.2%	87.9%	76.3%	81%
Sony	14%	89%	78%	63.3%	76.1%	19.1%
Starbucks	12%	15%	36.3%	22.8%	69.5%	20.2%
Subway	54.4%	60.4%	68.5%	70%	86%	73%
Tesla M.	65.3%	67.8%	68.8%	80.6%	78.3%	79.9%
Us Airways	88.2%	89.4%	82.3%	92%	59.3%	93.1%
Virgin Media	80%	83.5%	80.6%	88.6%	49.1%	89.7%
Yale U.	39%	49.3%	56.2%	56%	81.5%	62.2%
Zoo E.	90.3%	90.2%	93.8%	80.9%	17.3%	78.5%
Average	59.7%	65.7%	69.1%	69.9%	70.6%	70.9%
t-test	($p < 0.06$)	**0.058**	**0.0017**	**0.021**	0.32	**0.0026**

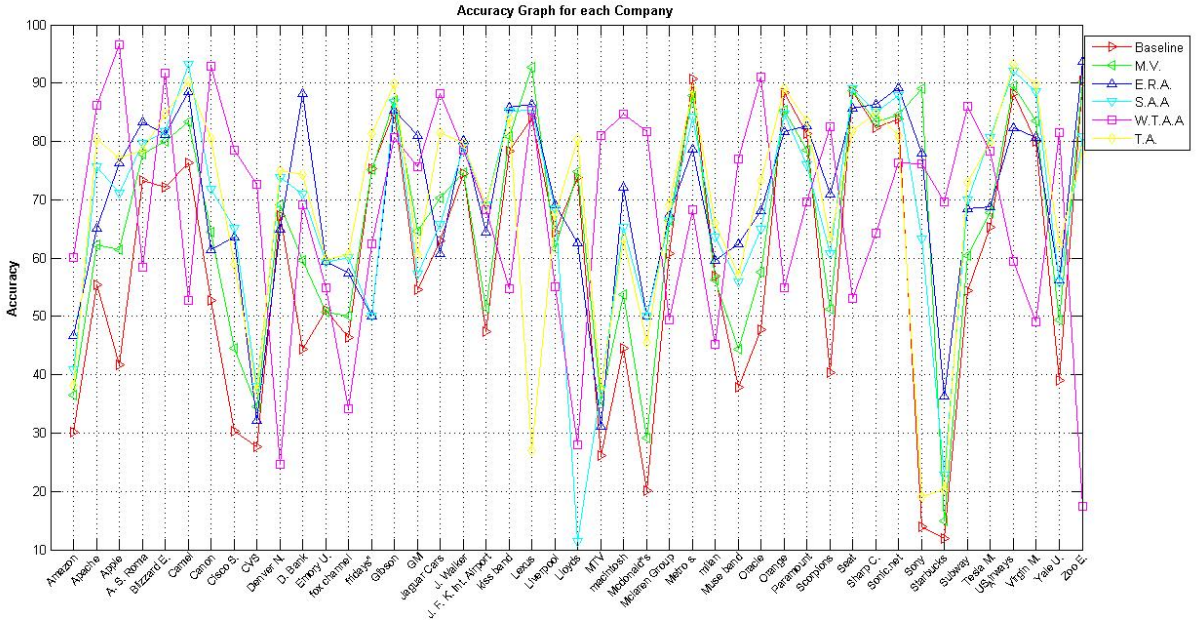

FIGURE 4.7: Accuracy Graph for testing data for each classification algorithm.

Chapter 5

Conclusion

5.1 Overall Analyze and Future Work

Researchers analyze twitter data for different purposes: finding influential ones, opinion mining, sentiment analysis, categorizing tweets, summarizing tweets, etc. In some of these tasks, like opinion and sentiment mining, the classification of the tweets based on entities requires an important preprocessing step, as the accuracy of further analysis depends on this step. In this study, we emphasise on the problem of finding related tweets to a given organization. This is a challenging task due to the organization name ambiguity. This task is more challenging due to specific three problems: the tweets and organization contain little information,misspellings in tweet text and the organizations in training data are different with those in test data.

We use external resources to enrich the information of organization. We realize an efficient classification process with the help of entity profiles, which we construct using different information sources. We observe that the accuracy of our classification technique depends on the quality of the entity profiles. However, from the analysis of the test set tweets we observe that in 4.18, there is a significant amount of tweets, which do not have any overlapping words with the corresponding company profile keywords. Therefore, this shows that as a future work, we need to increase the quality of our company profiles. We can use some techniques to do this job.

Using Twitter stream might be a good way to enrich our profiles with highly correlated keywords that are related with a given company. As it has been mentioned in Related Work section, in [4], we show that authors enrich the profile keywords that are both related and unrelated with a given company using Twitter stream. While inspecting Twitter stream, if the basic profile and a tweet have overlapping words, all words co-occurring with profile keywords in these tweets added to the profile. On the other hand,

if the inspected tweet and a basic profile do not have any overlapping word, they add all words in that tweet (ones above a certain threshold) to the company unrelated profile. In [4], it is also shown that the accuracy becomes as 0.84 using this enriching technique. We can apply the same technique to enlarge our company related and unrelated profiles. This might be essential for us, since our classification techniques that we use produce higher or lower accuracy results based on the overlapping level of a company profile and a tweet. As it is seen in 4.19, some companies such as *Amazon* or *Starbucks* show very low accuracy, when statistically significant classifiers over baseline perform on those testing data set. When we measure the relatedness value of *Amazon* and *Starbucks*, both have higher relatedness value. This observation shows that the reason of the low performance is highly correlated with having poor quality company profile. Therefore, the quality factor should be improved for future work.

Moreover, we may want to enrich the existing keywords that are relevant to the target company Web page even if they do not in company Web page. For this purpose, the techniques that are explained in [32] or [63] can be used for our classification task. In order to do that, we will need to analyse the keywords for finding a relationship between existing ones and newly obtained keywords that are semantically relevant with each other. For this purpose, we will benefit from Wikipedia graph structure. Structurally, Wikipedia can be viewed as a directed graph with vertices and edges corresponding to its entities. Since company home pages have restricted information, we may want to use each entity in the company Web page to generate Wikipedia graph structure. For this purpose, we need to take each entity as a vertex of the graph and if two entities are semantically related, these should be connected via edge. The hyperlinks in Wikipedia pages linking to the other Wikipedia articles for the graph we want to construct are potential edges. As a goal, we may construct edges linking to entities in the same topic. Also, the weight between two entities is computed in order to see the most related entities. For this purpose, we may want to use page ranking algorithm to rank related entities.

Also we can use a Tweet clustering approach which is mentioned in [63]. Like [63], we can link terms in company tweets to Wikipedia pages and use Wikipedia's link structure to do tweet clustering. In order to do that, firstly we need to extract term features in Twitter microblogs. We may use microblog tokens as features if those tokens exist as a Wikipedia anchor text at least once or if those tokens are used as Wikipedia article titles. Then, we will need to determine most relevant articles with a given microblog based on the extracted feature tokens. Lastly, we will need to do document similarity between candidate articles and cluster those articles as "related"or "unrelated"with a certain company.

When we analyse how we can produce more satisfactory results for future work, we can conclude that the word quality of unrelated terms is as important as word quality of relevant terms. We mention about gathering company related terms is a finite process; in contrast, the situation is not the same for collecting company unrelated keywords. As unrelated keyword set is infinite, gathering quality unrelated terms about a given company becomes very difficult task. Therefore, sometimes a keyword might exist both company related profile and unrelated profile that leads to erroneous results. For example, when we analyse first 100 keyword of "Apple"Inc., we notice that "game"and "video"keywords both exist company Wikipedia profile and company Wikipedia disambiguation profile. Therefore, as a future work, we need to develop a strategy that will prevent misplacement of such those terms. In [4], the threshold limit is proposed while generating the company unrelated profiles. If the weight of the candidate word is above a certain threshold, it is added to the profile. However, their system is based on inspecting tweets in Twitter stream, so the system is dynamic. Therefore, this method might not be valid for our system which is static, so we need to discover other convenient methods to solve that problem.

5.2 Possible Implications

Our method is scalable for other systems and precedes highly expensive information extraction system. Since the method is text-based and does not depend on any HTML structure, this classification system can be applicable for other social networks or blogs easily.

Appendix A

Additional Stopword List

TABLE A.1: Additional stop word list

alcatel	amadeus	apollo	armani	barclays	bart	blockbuster	boingo
cadillac	craft	delta	dunlop	edmunds	elf	emperor	fender
folio	foxtel	fujitsu	harpers	impulse	linux	lamborghini	lufthansa
liquid	luxor	lynx	mack	magnum	mandalay	marriott	marvel
mdm	mgm	mercedes	mercer	nikon	nordic	philips	pierce
pioneer	renaissance	renault	rover	shin	southwest	yamaha	borders
borders	best	buy	cme	delta	dunkin	ford	gap
leap	frog	lennar	opera	overstock	palm	rim	southwest
sprint	tam	warner	amazon	apache	apple	rome	a.s.
blizzard	camel	canon	cisco	cvs	denver	nugget	deutzhe
emory	friday's	gibson	gm	jaguar	johnie	kennedy	kiss
lexus	liverpool	lyds	macintosh	mcdonalds	mclaren	metro	milan
muse	oracle	orange	paramount	scarpions	seat	sharp	sonic
sony	starbucks	subway	tesla	us	virgin	yale	zoo
hundred	thousand	million	billion	trillion	date	annually	annual
year	quarterly	yearly	quarter	month	monthly	week	weekly
day	daily	january	february	march	april	may	june
july	august	september	october	november	december	null	want
monday	tuesday	wednesday	thursday	friday	saturday	sunday	one
one	two	three	four	five	six	seven	eight
nine	ten	eleven	twelve	thirteen	fourteen	fifteen	sixteen
seventeen	eighteen	nineteen	twenty	thirty	forty	fifty	sixty
seventy	eighty	ninety	first	second	third	fourth	fifth
sixth	seventh	eighth	ninth	tenth	i	ii	iii
iv	v	vi	vii	viii	ix	x	xi
xii	xiii	xiv	xv	xvi	xvii	xviii	xix
xx	able	out	who	can	his	her	me
rather	way	just	did	never	too	up	were
him	we	us	you	she	them	from	else
ever	which	he	where	wasn	what	so	since
how	because	hence	therefore	however	about	its	despite
actually	have	had	am	only	one	didn	either
often	later	all	after	when	more	has	other
are	off	also	unless	any	until	certain	through
would	could	within	may	yes	both	now	neither
nor	than	less	here	best	each	weren	been
called	nevertheless	although	over	day	years	first	end
around	while	based	most	per	under	without	before
include	consist	during	almost	among	along	instead	back
even	though	between	due	back	some	being	cache
ones	only	onto	others	otherwise	ought	ours	ourselves
outside	over	overall	own	particular	particularly	perhaps	placed
please	plus	possible	presumably	probably	provides	que	quite
really	reasonably	regarding	regardless	regards	relatively	respectively	right
said	same	saw	say	saying	says	secondly	see
seeing	seem	seemed	seeming	seems	seen	self	selves

TABLE A.2: Additional stop word list

sensible	sent	serious	seriously	several	shall	should	shouldn′t
somebody	somehow	someone	something	sometime	sometimes	somewhat	somewhere
soon	sorry	specified	specify	specifying	still	sub	sup
sure	take	taken	tell	tends	than	thank	thanks
thanx	that′s	thats	theirs	themselves	then	thence	there′s
thereafter	thereby	therefore	therein	theres	thereupon	they′d	they′ll
they′re	they′ve	think	thoroughly	those	though	throughout	thus
together	took	toward	towards	tried	tries	truly	try
trying	twice	under	unfortunately	unlikely	until	unto	up
upon	use	used	useful	uses	using	usually	value
various	very	via	viz	vs	want	wants	we′d
we′ll	we′re	we′ve	welcome	well	went	weren′t	what′s
whatever	whence	whenever	where	where′s	whereafter	whereas	whereby
wherein	whereupon	wherever	whether	while	whither	why	who′s
whoever	whole	whom	whose	will	willing	wish	with
won′t	wonder	would	wouldn′t	yet	you′d	you′ll	you′re
you′ve	yours	yourself	yourselves	zero	okay	old	once
a′s	about	above	according	accordingly	across	afterwards	again
against	ain′*t*	all	allow	almost	alone	already	nobody
always	amongst	another	anybody	anyhow	anyone	anything	anyway
anyways	anywhere	apart	appear	appreciate	appropriate	aren	aside
ask	asking	associated	available	away	awfully	became	become
becomes	becoming	beforehand	behind	believe	below	beside	besides
better	beyond	brief	came	can	can′*t*	cannot	cant
cause	causes	certain	certainly	changes	clearly	come	comes
concerning	consequently	consider	considering	containing	corresponding	could	couldn′t
course	currently	definitely	described	despite	didn′t	different	do
does	doesn′t	doing	don′t	done	down	downwards	edu
elsewhere	enough	entirely	especially	et	etc	every	everybody
everyone	everything	everywhere	ex	exactly	example	except	far
few	followed	following	follows	former	formerly	further	furthermore
get	gets	getting	given	gives	go	goes	going
gone	got	gotten	greetings	hadn′t	happens	hardly	hasn′t
haven′t	having	he′s	hello	help	hence	here	here′s
hereafter	hereby	herein	hereupon	hers	herself	hi	himself
hither	hopefully	how	howbeit	however	i′d	i′ll	i′m
i′ve	ie	ignored	immediate	inasmuch	inc	indeed	indicate
indicated	indicates	inner	insofar	into	inward	isn′t	it′d
it′ll	it′s	itself	just	keep	keeps	kept	know
known	knows	last	lately	later	latter	latterly	least
less	lest	let	let′s	like	liked	likely	little
look	looking	looks	ltd	mainly	many	may	maybe
mean	meanwhile	merely	might	more	moreover	most	mostly
much	must	myself	name	namely	nd	near	nearly
necessary	need	needs	neither	never	nevertheless	new	next
non	none	noone	nor	normally	not	nothing	novel
nowhere	obviously	often	oh	ok			

Bibliography

[1] Twitter Official Web Site. http:https://twitter.com/.

[2] Wikipedia Official Web Site. http://en.wikipedia.org/wiki/Twitter.

[3] WePS-3 Data Set. http://nlp.uned.es/weps/.

[4] S. R. Yerva, Z. Miklós, and K. Aberer. Entity-based classification of twitter messages. *IJCSA*, 9(1):88–115, 2012.

[5] Kullback Leibler Distance. http://staff.science.uva.nl/~tsagias/?p=185.

[6] Cosine Similarity. http://en.wikipedia.org/wiki/Cosine_similarity.

[7] G. James. Majority vote classifiers: Theory and applications, 1998.

[8] A. J., S. K. P., and K. Y. Feature selection in top-down visual attention model using weka. *International Journal of Computer Applications*, 24(4):38–43, June 2011. Published by Foundation of Computer Science.

[9] N. Kumar, G. P. Obi Reddy, and S. Chatterji. Evaluation of best first decision tree on categorical soil survey data for land capability classification. *International Journal of Computer Applications*, 72(4):5–8, June 2013. Published by Foundation of Computer Science, New York, USA.

[10] Multilayer Perceptron. http://en.wikipedia.org/wiki/Multilayer_perceptron.

[11] Attribute Selected Classifier. http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/AttributeSelectedClassifier.html.

[12] N. Dalvi, R. Kumar, B. Pang, and A.Tomkins. Matching reviews to objects using a language model. In *EMNLP*, pages 609–618. ACL, 2009. ISBN 978-1-932432-63-3. URL http://dblp.uni-trier.de/db/conf/emnlp/emnlp2009.html#DalviKPT09.

[13] S. R. Yerva, Z. Miklós, and K. Aberer. It was easy, when apples and blackberries were only fruits. In *CLEF Notebook Papers/LABs/Workshops*, 2010.

[14] S. R. Yerva, M. Catasta, G. Demartini, and K. Aberer. Entity disambiguation in tweets leveraging user social profiles. In *IRI*, pages 120–128, 2013.

[15] S. R. Yerva, Z. Miklós, and K. Aberer. What have fruits to do with technology?: the case of orange, blackberry and apple. In *WIMS*, page 48, 2011.

[16] F. Perez-Tellez, D. Pinto, J. Cardiff, and P. Rosso. On the difficulty of clustering company tweets. In *Proceedings of the 2d International Workshop on Search and Mining User-generated Contents*, SMUC '10, pages 95–102, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0386-6. doi: 10.1145/1871985.1872001. URL `http://doi.acm.org/10.1145/1871985.1872001`.

[17] D. M. Agustín, R. M. Unanue, A. P. G.-Plaza, and V. Fresno. Unsupervised real-time company name disambiguation in twitter. In *Sixth International AAAI Conference on Weblogs and Social Media*, 2012.

[18] S.Zhang and H. Yu. Supervised and semi-supervised methods based organization name disambiguity. In *PACLIC*, pages 615–621, 2011.

[19] Paul Kalmar. Bootstrapping websites for classification of organization names on twitter, 2010.

[20] M. A. GarcÃa-Cumbreras, M. GarcÃa-Vega, F. MartÃnez-Santiago, and J. M. Perea-Ortega. Sinai at weps-3: Online reputation management. In *Working Notes for the CLEF 2010 Workshop*, Padua, Italy, 2010.

[21] D. V. Kalashnikov, S. Chen, R. Nuray, S. Mehrotra, and N. Ashish. Disambiguation algorithm for people search on the web. In *Proc. of the IEEE 23rd International Conference on Data Engineering IEEE ICDE 2007, short paper*, Istanbul, Turkey, April 16–20 2007.

[22] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM SDM*, 2005.

[23] E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR*, pages 27–34, 2006.

[24] W. Shen, J. Han, and J. Wang. A probabilistic model for linking named entities in web text with heterogeneous information networks. In *SIGMOD Conference*, pages 1199–1210, 2014.

[25] Y. H. Chiang, A. Doan, and J. F. Naughton. Modeling entity evolution for temporal record matching. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1175–1186, New York,

NY, USA, 2014. ACM. ISBN 978-1-4503-2376-5. doi: 10.1145/2588555.2588560. URL http://doi.acm.org/10.1145/2588555.2588560.

[26] N. Dalvi, R. Kumar, and B. Pang. Object matching in tweets with spatial models. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 43–52, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124303. URL http://doi.acm.org/10.1145/2124295.2124303.

[27] N. Dalvi, R. Kumar, B. Pang, and A. Tomkins. A translation model for matching reviews to objects. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 167–176, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-512-3. doi: 10.1145/1645953.1645977. URL http://doi.acm.org/10.1145/1645953.1645977.

[28] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2011.

[29] T. Xu and D. W. Oard. Wikipedia-based topic clustering for microblogs. *Proc. Am. Soc. Info. Sci. Tech.*, 48(1):1–10, 2011. URL http://dx.doi.org/10.1002/meet.2011.14504801186.

[30] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan. Hydra: large-scale social identity linkage via heterogeneous behavior modeling. In *SIGMOD Conference*, pages 51–62, 2014.

[31] W. Feng and J. Wang. Retweet or Not?: Personalized Tweet Re-ranking. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 577–586, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1869-3. doi: 10.1145/2433396.2433470. URL http://doi.acm.org/10.1145/2433396.2433470.

[32] W. Zhang, D. Wang, G. Xue, and H. Zha. Advertising keywords recommendation for short-text web pages using wikipedia. *ACM TIST*, 3(2):36, 2012. URL http://dblp.uni-trier.de/db/journals/tist/tist3.html#ZhangWXZ12.

[33] I. Hulpus, C. Hayes, M. Karnstedt, and D. Greene. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 465–474, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1869-3. doi: 10.1145/2433396.2433454. URL http://doi.acm.org/10.1145/2433396.2433454.

[34] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In Nicoletta Calzolari Conference Chair, Khalid Choukri, Bente Maegaard,

Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation LREC'10*, Valletta, Malta, May 2010. European Language Resources Association ELRA. ISBN 2-9517408-6-7.

[35] NLTK Official Web Site. http://www.nltk.org/.

[36] Wordnet Lemmatizer. https://nltk.googlecode.com/svn/trunk/doc/api/nltk.stem.wordnet.WordNetLemmatizer-class.html.

[37] yelp restaurant. http://www.yelp.com/.

[38] Pissed Consumer. http://www.pissedconsumer.com/.

[39] Amazon Web site. http://www.amazon.com/.

[40] cnet. http://www.cnet.com/.

[41] Trip Advisor Web site. http://www.tripadvisor.com/.

[42] Python Enchant Library Official Web Site. https://pypi.python.org/pypi/pyenchant/.

[43] Levensthein Distance. http://en.wikipedia.org/wiki/Levenshtein_distance.

[44] Norvig's Web site. http://norvig.com/spell-correct.html.

[45] Chunking Web Site. http://www.nltk.org/book/ch07.html.

[46] Latent Semantic Indexing Web site. http://technowiki.wordpress.com/2011/08/27/latent-semantic-analysis-lsa-tutorial/, .

[47] Latent Semantic Indexing Wikipedia Web site. http://en.wikipedia.org/wiki/Latent_semantic_analysis, .

[48] Gensim Package. https://pypi.python.org/pypi/gensiml.

[49] Maximum Entropy Classifier for Text Classification. http://www.kamalnigam.com/papers/maxent-ijcaiws99.pdf.

[50] Weka Official Web Site. http://www.cs.waikato.ac.nz/ml/weka/.

[51] j48 Web site. http://www.d.umn.edu/~padhy005/Chapter5.html.

[52] Maximum Likelihood Web site. http://en.wikipedia.org/wiki/Maximum_likelihood.

[53] Backpropogation Algorithm Wikipedia Web site. http://en.wikipedia.org/wiki/Backpropagation.

[54] Decision Tree Web site. http://en.wikipedia.org/wiki/Decision_tree_learning.

[55] Information Gain Web site. http://en.wikipedia.org/wiki/Information_gain_in_decision_trees.

[56] Gini Index Web site. http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Gini_coefficient.html.

[57] R. E. Schapire. Explaining adaboost.

[58] E. Amigó, J. Artiles, J. Gonzalo , D. Spina, B. Liu, and A. Corujo. WePS-3 Evaluation Campaign: Overview of the On-line Reputation Management Task. In *Conference on Multilingual and Multimodal Information Access Evaluation (CLEF)*, 2010.

[59] DBPEDÄ°A Web Site. http://wiki.dbpedia.org/About.

[60] CFS Web Site. http://en.wikipedia.org/wiki/Feature_selection.

[61] Bias-Variance Web Site . http://scott.fortmann-roe.com/docs/BiasVariance.html.

[62] Statistical T-test Wikipedia Web Site. http://en.wikipedia.org/wiki/Student's_t-test.

[63] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 787–788, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-597-7. doi: 10.1145/1277741.1277909. URL http://doi.acm.org/10.1145/1277741.1277909.