Information Retrieval Draft for NLP Project

AnalyzeIT

Emmanouil Manousogiannis
4727517
e.manousogiannis@student.tudelft.nl

Alexandros Stavroulakis
4747933
a.stavroulakis@student.tudelft.nl

Sofia Tsoni
4747941
s.tsoni@student.tudelft.nl

Georgia Zarnomitrou
4724089
g.zarnomitrou@student.tudelft.nl

ABSTRACT

This paper provides a detailed summary of our implementation of an aspect and document level sentiment analysis system using a lexicon-based approach. The paper starts with an introduction to the problem statement, the motivation behind our choice to investigate this topic as well as the relevant research that already exists in this domain. In addition, we present an overall plan of the steps one has to follow to tackle this research problem. Next we present a comprehensive description of all the implementation decisions we took while creating our system (a link to all our scripts and code can be found here [14]). Then we display the results we have gotten from various experiments. Following that we present all the insights we have gained through the procedure of implementing an aspect level sentiment analysis system. Lastly, we offer suggestions of how our system could be further improved in the future.

1 INTRODUCTION

1.1 Problem Statement

In this day and age everyone wishes to travel around the world and visit new and interesting places. These traveling experiences however can be ruined by staying in hotels that do not meet a person's expectations. Also every hotel's management wishes to attract as many clients as possible and for that they have to turn their hotel into the ideal place for vacation. Considering the plethora of hotel reviews one can encounter online it is impossible to quickly sift through all of them and discern the writer's feelings or complaints. So, in this report we address the important and difficult NLP problem of correctly identifying the sentiment expressed in a natural language text. Specifically, we decided to investigate how a sentiment analysis system works and then implement one that receives as input hotel reviews and outputs the overall sentiment expressed in them. Besides the overall sentiment expressed, individual sentiment identification is performed for specific aspects of a hotel. The aspects we take into account in our analysis are: quality of service, location of the hotel, quality of the rooms and monetary cost.

1.2 Motivation of the Problem

Travel planing and hotel booking using famous websites is a major commercial use nowadays. As a consequence, there is an increasing amount of users that actively provide public feedback, based on their experience of a product or service.

As mentioned above this large amount of available review data, plays a vital role both for the managerial stuff of a hotel as well as prospective future clients. However, the fact there are so many human generated reviews makes it impossible for a user to read all of them and form a well rounded opinion. Thus it is really necessary to have a system that can go through these large quantities of reviews quickly and provide the user with other people's negative or positive opinions regarding the most important aspects of a hotel. The existence of such a system could be useful for users that search for a hotel to spend a pleasant stay. Specifically, they could use the system to go through various reviews for a multitude of hotels until they find accommodations which meet their personal needs and desires. The importance of the proposed system is even greater for hotel owners and a hotel's management because it can be used to find which aspects of their hotel does not meet their clients expectations. In this case the system could be used to extract feedback from various users regarding different aspects of a hotel's services which could then be used to identify problem areas that need to be improved.

1.3 Overall Plan to tackle the problem

In order to achieve our goal of correctly identifying the sentiment expressed in a review both on a document as well as an aspect level, there are several steps that have to be followed.

The first and most challenging part was the implementation of a web crawler that would allow us to get recent reviews from our website of interest Booking.com. Unfortunately, due to some failed attempts and time constraints we eventually decided to use readily available data from the website TripAdvisor. Then we focused on the crucial step of preprocessing these reviews, in order to keep the important parts that yield information regarding the sentiment expressed and discard the unnecessary content.

Next from the remaining content we had to extract the parts that refer to specific aspects of a hotel. In our implementation we chose to focus on four aspects (location, quality of service, quality of rooms and monetary cost) due to their importance and because we think they are the most representative aspects of a hotel. We reached this decision after we visited websites like Booking and Tripadvisor and created a list of all the aspect terms they use in their reviews and found the aspects that were common for all of them. So we split the review in categories based on these aspects and then assign a positive or negative polarity to each of them separately. In addition, we extract the sentiment expressed for the hotel as a whole from each review and present an overall sentiment polarity. The above procedure can be seen in figure **??**. The last step was to test our system with a portion of the reviews we gathered and get the polarities assigned to each review.

2 BACKGROUND

The problem of Sentiment Analysis in Product Reviews has been researched a lot using many different methodologies. The goal of our project was to approach the problem using an aspect based methodology. This means that we focused on the different specific aspects that appear in hotel reviews, in order to provide more detailed and abstract information about each part of a hotel that a reviewer is evaluating. Below we present some of the existing research in this study field, which was used as inspiration for our sentiment analysis implementation.

In [1] the researchers predict the rating of reviews based on a customer's opinion for different aspects of the product. Their system, initially identifies the features that are relevant to customers when evaluating a certain type of product, as well as their relative importance. After that the system extracts the opinion of the customer about the different features of the product from the review. To implement this, they use Wordnet to associate the sentences with a product attribute. The relative importance of the features, combined with the user opinions construct a *Vector of Feature Intensities* which represents the review. This vector will be used as input to a classifier in order to classify the review into different rating categories. Their system was evaluated with 1000 hotels from Booking.com.

In a similar approach to the aforementioned paper, this paper [7] considers the aspect level of mobile phone reviews from Amazon.com. The proposed system here is also using WordNet to determine the orientation of sentences, based on opinion words and their synonyms and antonyms. All the features of the product on which reviews are given are then identified, and the orientation of the sentence for each feature is determined. The total polarity of the given sentence is determined on the basis of the majority of opinion words.

Using a different approach [2] Luigi DI Carro and Matteo Grella, perform sentiment analysis of a text, based on a set of rules that are applied on a syntactic level, with the help of a syntactic dependency parser. Based on the same principal, the group of researchers in [3] consider the dependencies between the words of a sentence in a document, in order to retrieve useful relationships between some words that indicate a specific aspect and some other words that include emotion and hence can be assigned with a positive or negative polarity. These relationships can be determined by searching for specific syntactic rules.

In [4] the authors, use another interesting method which is extracting a lexicon specifically for the domain of the hotels, with semantically relevant words based on a given corpus. The corpus they use is from Tripadvisor. However it is restricted only to reviews of hotels, which should also be written in English. The lexicon is afterwards generated based on the vocabulary in the available data set. Each entry in the lexicon is defined by a token and its part-of-speech (POS) tag.

3 ASPECT LEVEL SENTIMENT ANALYSIS

In this section we give a detailed description of the steps we followed in our implementation of a sentiment identification system. Our code is written solely in *Python*, where we used the main packages **nltk**, to handle the natural language processing parts and **pandas** for the management of the dataframes we constructed.

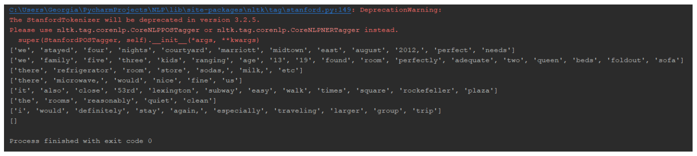
3.1 Preprocessing

The stage of preprocessing plays an important role in the sentiment analysis procedure. Preprocessing is used in order to clean and extract only the necessary information from the text and avoid the redundant content. The preprocessing stage can be separated in many smaller steps, below we describe in more detail the ones we followed in our implementation.

The first step we took was to gather the data, namely **Data Acquisition** step. Initially our goal was to gather hotel reviews from a large website with many users, the options where Booking.com and TripAdvisor. Getting data from Booking.com was more difficult than expected, because there are no free available datasets. Thus we used the dataset by Carnegie Mellon University with 20.000 hotel reviews by TripAdvisor [9]. The reviews are mostly in English, but there are also some in French. In our implementation we only deal those written in the English language.

After we collected our data the next step was the **cleaning of data**. With this step starts the actual preprocessing stage. The first pre-processing step was the cleaning of the reviews by removing any punctuation marks or other special characters. In this step we also transformed all the words into their lowercase format. Next we performed **tokenization** which means to split the sentences in words, for that purpose we used the built-in function from the nltk package.

The next step we performed was the **removal of commonly used words** like "a", "the" etc. This is another pre-processing step that was achieved with the help of the external library *nltk*. In more detail, we used the stoplist provided by the *nltk* package to remove all the words of the reviews included in that list. An example of our implementation of these three first preprocessing steps are depicted in figure 1.

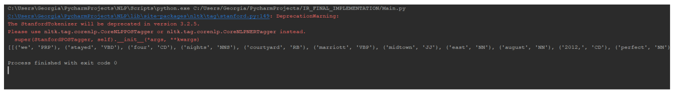


```
C:\Users\Georgia\PycharmProjects\NLP\lib\site-packages\nltk\tag\stanford.py:149: DeprecationWarning:
The StanfordTokenizer will be deprecated in version 3.2.5.
Please use nltk.tag.corenlp.CoreNLPPOSTagger or nltk.tag.corenlp.CoreNLPNERTagger instead.
  super(StanfordPOSTagger, self).__init__(*args, **kwargs)
['we', 'stayed', 'four', 'nights', 'courtyard', 'marriott', 'midtown', 'east', 'august', '2012,', 'perfect', 'needs']
['we', 'family', 'five', 'three', 'kids', 'ranging', 'age', '13', '19', 'found', 'room', 'perfectly', 'adequate', 'two', 'queen', 'beds', 'foldout', 'sofa']
['there', 'refrigerator', 'room', 'store', 'sodas,', 'milk,', 'etc']
['there', 'microwave,', 'would', 'nice', 'fine', 'us']
['it', 'also', 'close', '53rd', 'lexington', 'subway', 'easy', 'walk', 'times', 'square', 'rockefeller', 'plaza']
['the', 'rooms', 'reasonably', 'quiet', 'clean']
['i', 'would', 'definitely', 'stay', 'again,', 'especially', 'traveling', 'larger', 'group', 'trip']
[]

Process finished with exit code 0
```

Figure 1: Example of a review after the first three preprocessing steps: cleaning of data, tokenization and removal of stop words

One of the most important parts of the pre-processing stage concerns the **Part Of Speech Tagging** task. In order to complete the subsequent steps of our implementation it was necessary to know which part of speech each word was. This is useful because some parts of speech are more important than others in sentiment analysis (e.g adjectives are very useful for sentiment identification). In order to achieve this goal firstly we used the POS tagger by the *nltk* library (see figure 2) however the results were not satisfactory. Thus we searched for other POS Taggers and we ended up with the Standford POS Tagger, which had much better results. However the Stanford tagger is extremely slower than the aforementioned nltk POS tagger. The results that support the above arguments are presented in the Experiments section 4.1.



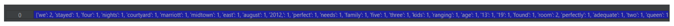
```
C:\Users\Georgia\PycharmProjects\NLP\Scripts\python.exe C:/Users/Georgia/PycharmProjects/IR_FINAL_IMPLEMENTATION/Main.py
C:\Users\Georgia\PycharmProjects\NLP\lib\site-packages\nltk\tag\stanford.py:149: DeprecationWarning:
The StanfordTokenizer will be deprecated in version 3.2.5.
Please use nltk.tag.corenlp.CoreNLPPOSTagger or nltk.tag.corenlp.CoreNLPNERTagger instead.
  super(StanfordPOSTagger, self).__init__(*args, **kwargs)
[[('we', 'PRP'), ('stayed', 'VBD'), ('four', 'CD'), ('nights', 'NNS'), ('courtyard', 'RB'), ('marriott', 'VBP'), ('midtown', 'JJ'), ('east', 'NN'), ('august', 'NN'), ('2012,', 'CD'), ('perfect', 'NN')

Process finished with exit code 0
```

Figure 2: Example of a review after implementation of POS-tagging from nltk package

Next we performed a **lemmatization** of all the words in a review meaning we converted them to their common base form. Firstly we tried the Porter stemmer but again the results were not the best possible, so we used the WordnetLemmatizer from the *nltk* library.

In order to get the sentiment of each word we needed to know how many times it appeared in each review. Thus the next step was to count the **Word Frequency** (see figure 3).



```
0    {'we': 2, 'stayed': 1, 'four': 1, 'nights': 1, 'courtyard': 1, 'marriott': 1, 'midtown': 1, 'east': 1, 'august': 1, '2012,': 1, 'perfect': 1, 'needs': 1, 'family': 1, 'five': 1, 'three': 1, 'kids': 1, 'ranging': 1, 'age': 1, '13': 1, '19': 1, 'found': 1, 'room': 2, 'perfectly': 1, 'adequate': 1, 'two': 1, 'queen': 1
```

Figure 3: Example of word frequency counter for a review

Finally we reached the **Aspect extraction** step, which refers to the identification and separation of those sentences or sentence fragments of the review that express an opinion regarding an aspect of the hotel. For this step we created a list with the aspects that we want to search for throughout the reviews, more specifically we consider the sentiment for the rooms, the location, the service and the value of the hotel. In order to gather as much information as possible regarding these aspects from the reviews we created a list for each aspect. Each list contains the most commonly used words associated with each aspect expanded with the help of a small function that searches for synonyms and antonyms using Wordnet. Our approach entails searching the review for these keywords that are related to our four aspects and extracting the sentence or phrase it is in. Then this sentence or phrase are analyzed in order to find the sentiment it wishes to convey.

3.2 Sentiment

For sentiment extraction we tried some different approaches, in order to choose the one that results in better accuracy. In order to extract a positive or negative sentiment, from a word related to one of our extracted aspects, we initially used the SentiWordNet corpus in Python. After the syntactical analysis performed by our Tagger, the input to the sentiwordnet was the part of speech of the word, i.e. adjective adverb, combined with the word itself. The result was one positive and one negative value, where the larger value was the one that determined the final polarity of the word.
However, we noticed that SentiWordNet did not always predict the correct values for each word, which finally led to totally misleading results. To address this issue, we followed two different approaches. The first one was to derive a corpus of positive and negative words (opinion lexicon), with which we could check the returned sentiment from SentiWordNet, and the second was to create our own domain specific lexicon.

In the first approach, we simply checked whether the returned sentiment from SentiWordNet was in agreement with the positive and negative words corpus we had. In case of disagreement, we considered only the value of the opinion lexicon. However the opinion lexicon does not contain a value, just tags regarding the positivity or negativity of the word. Thus we give a value with the sign specified by the opinion lexicon.
In the second approach, we created our own lexicon, based on the following procedure [8]. Using a dataset of 20,000 hotel reviews, different to the ones we used for testing of course, we kept track of any adjective, adverb or verb that appeared to be relevant to one of our aspects. Based on that, we counted for each of these words, how many positive or negative reviews they were present in. We then computed the final positive and negative frequency of each word:

$$frec_{+} = \frac{pos(w)}{POS}.$$
$$frec_{-} = \frac{neg(w)}{NEG}.$$

where NEG and POS are the total number of positive and negative reviews in the 20,000 review dataset.
Finally, the sentiment of each word was computed like below.

$$sent(w) = PD(w)^2 * sign(PD(w)).$$
$$\text{where } PD(w) = \frac{freq^{+} - freq^{-}}{freq^{+} + freq^{-}}.$$

Our final lexicon consisted of more than 6000 words, and each one was assigned a positive or negative sentiment value. The results we got for each of these approaches can be seen in the experiment section 4.2.

3.3 Dependency

It is common when referring to a specific subject matter that you notice some standard writing structures. Such is the case with hotel reviews where the majority of people use similar writing style to express their feelings about hotels they stayed in. Thus it is crucial to construct a set of rules that includes the most common syntactic structures that are used when someone is expressing his or her opinion about a hotel. For that reason we decided to use a dependency parser, and more specifically the Standford Dependency parser[13], in order to exploit the existence of these structures and construct different rules that refer to hotel reviews. Using these rules we were able to detect the sentiment expressed for different aspects in a review. The result we get from the dependency parser is a list of lots of different rules between pairs of words. So after many experimentations we found the most important rules for our goal of sentiment detection in hotel reviews. Next we present the rules that we chose in order to detect the sentiment of the aspects that we want and the reasoning behind this choice.

- **amod** → Adjectival modifier. For instance in the review "Great location. Awful service." this rule is expressed as **((location, NN), amod, (Great, JJ))** and **((service, NN), amod, (Awful, JJ))**. Considering natural languages it is obvious that the part of speech with which we express sentiment the most are adjectives. Thus our first rule was to use the dependency parser to locate pairs of words that referred to one of the four aspects we are interested in and an accompanying adjective. Then we used the adjective to calculate the sentiment.

- **dobj** → Direct object. For instance in the sentence "I like the room.", the produced rule is **((loved, VBD), dobj, (room, NN))**. Another observation we made was the existence of these kinds of word pairs where there was a direct reference to one of the four aspects we were interested in. So we added this rule in our sentiment identification system to get the sentiment expressed for our aspects through verbs, since verbs represent another part of speech that expresses sentiment.
- **nmod** → Noun modifier. For example the rule of the sentence "Friendliness and helpfulness with fluent English by the staff." is **((Friendliness, NNS), nmod, (staff, NN))**. Next we noticed this dependency rule that shows another way one of our aspect words (nouns) can be linked to adjectives.
- **nsubj and nsubjpass** → Nominal subject and Nominal passive subject. For example in "The location was great.", the rule is **((great, JJ), nsubj, (location, NN))**. These are just two more kind of dependencies that appear in the Stanford parser that associate our aspects (noun words) to verbs and adjectives.
- **neg** → Negation modifier. With this rule we can detect the existence of a negation. For example "The room wasn't clean." we can find that **((clean, JJ), neg, (n't, RB))**. So we can detect if an adjective has negation which means that we have to inverse its sentiment. As far as this rule is concerned we noticed discrepancies in the pair of words created by the Stanford tagger so we wrote additional code to account for this event.

Using all of the above rules, we can detect the different aspects of every review and the important words that are linked with them. So, having found these words and also having calculated the strength of their sentiment, we were able to calculate and assign a final polarity to each aspect we want to detect in a review.

3.4 Aspect-Level Baseline Algorithm

In this section we present the earliest stage of our implementation of an aspect based sentiment analysis algorithm. We began our implementation by creating a baseline algorithm that detects sentiment with the help of one simple rule, we only use the adjectives in a review to determine the polarity of the sentiment expressed. This rule is based on the fact that an adjective is one of the most important parts of speech people use to express their opinions.

In this implementation we make the assumption that each sentence expresses sentiment for one of the four aspects we are checking. Specifically, in our baseline we search for sentences that refer to only one of the four aspects, locate the adjectives in that sentence (using the part of speech tagger) and add the sentiment value assigned to them by the "SentiWordnet" corpus. If the resulting sentiment is negative we classify that review as negative regarding that aspect otherwise as positive (zero values are ignored).

3.5 Final Algorithm Implementation

After the implementation of the aforementioned baseline we started considering ways in which we could exploit the syntactic structures that exist in natural languages to better detect sentiment in our reviews. After some research and experimentation we decided to perform aspect level sentiment analysis using the Stanford parser and the rules described in section 3.3 for the identification of the dependencies. Our final algorithm consists of several steps that we describe in detail in this section.

After obtaining each review the first step is to split it in order to distinguish the different sentences. The next step is to extract each of these sentences and search if there is a reference to one of the four aspects by looking for keywords from the lists we created for each aspect. If such a reference exists we pass the sentence through the dependency parser. The reason we check for a reference to one of the aspects is to discard unwanted content, because parsing a sentence through the Stanford parser is a very time consuming procedure. Through the parser we produce the dependencies between the words of the sentence as mentioned in a previous section. Having now all the dependencies between the words in each sentence of each review, all that we have to do is to detect the sentence fragments where different aspects are mentioned and compute their polarity.

Although, a necessary step before that, is to compute the sentiment of each word in every sentence. For that purpose we experimented with different lexicons in order to specify the sentiment of each word. As mentioned in section 3.2, we started using the SentiWordNet, which didn't provide us with good enough results. Thus we turned our direction to an opinion lexicon, which was more appropriate for our case. In more detail we search each word in SentiWordNet, specifying also its frequency and the part of speech that it is for the current sentence. After that we check if the result we get is also verified by the opinion lexicon. If the word for example seem to be positive in the SentiWordNet, but it is in the negative list of words based on the opinion lexicon, then we have a conflict. In this case we overwrite the sentiment value provided by the SentiWordNet and if the word is considered as positive by the opinion lexicon we set its value to +0.5. Eventually the result will be just positive or negative, but since we will have to sum all the values of the words characterizing each aspect we chose the value 0.5, since it shows a strong enough sentiment without exaggerations. The result of that procedure is a list for each sentence, which contains dictionaries, with key the word and value the sentiment of that word.

After the sentiment identification we have the sentiment of each word, so we can compute the polarity of each aspect inside each review. In order to achieve this we conduct the following steps:

(1) The first step is to pass the dependencies produced by the parser and the sentiment of each word of a review to the appropriate function.
(2) Then we have to decide which rules of the dependency parser we will consider. The reason for choosing specific rules is mentioned in section 3.3.
(3) Next we process each sentence of the review separately and find all the pairs of words that satisfy any of the rules we have chosen.

However in the case of negation (neg) the parser does not always provide us with the correct pairs, so we decided to handle this case differently. For that purpose we created the negation function which is called when a negative word is located in the review. Our code can produce the correct sentiment only when the words "no, neither, nor, not" are found.

(4) After finding these pairs, we check if any of the two words in a pair refers to an aspect of the hotel (this is achieved with the help of the four lists of words we created, one for each aspect). If indeed there is a reference then we assign to that aspect the sentiment value of the other word in the pair. This value has already been calculated from the function we described above.
The case of the negation is different in this step too. In more detail when a negation is found we take into consideration two pairs of the dependency parser. One with the aspect and the negation word and one with the negation word and the word that characterizes the aspect (oppositely). Thus the value returned by this function is minus the sentiment of the word that characterizes the aspect.

(5) As a final step, after processing all the sentences of a review, we sum up the values that were assigned to the same aspect in order to produce the final sentiment polarity of each aspect of each review.

A crucial matter that has to be mentioned is the run time of the process for constructing the dependencies between the words. As we discovered by experimentation, Standford dependency parser is highly time consuming, something that is not preferable in such application, where we want to parse thousands of reviews. Although the reason that led us to eventually use this parser is that despite its execution time, the results were significantly better that any other methods that we tested.

3.6 Document Level Sentiment Analysis

In this project we mainly focused on sentiment analysis on aspect level of the hotel reviews. However, we also performed document level sentiment analysis in order to retrieve the total polarity of a review. We considered different options in order to extract this polarity. One of them, was to simply add up the extracted polarity for each aspect , however this could be dangerous as there could be many other aspects or perspectives that the user reviewed, which were not related to the detected aspects.
Hence, we decided to perform Part-of-Speech tagging again and then, extract the sentiment of every adjective detected in the review, based either on SentiWordNet or in our lexicon approaches. Despite, the simplistic nature of this approach the results seem to be quite good as can be seen in the following sections.

4 EXPERIMENTS

As we have already mentioned sentiment analysis implementation is a complicated task that requires following a long series of steps. Also due to the fact that sentiment analysis is such an important and well researched field of study sometimes we were faced with important decisions regarding the different built-in preprocessing functions we would use in our implementations. In order to check the ones best suited for our purposes we ran a series of tests. In addition, in order to check our final algorithm's performance we ran an experiment to compare it with our baseline algorithm. Below we present the various results we got from these experiments.

4.1 POS-TAGGERS

In the course of our implementation we noticed that one of the problem areas was that sometimes the built-in tagger function of the nltk package assigned wrong tags to some words. This in turn affected the following steps where the POS-tagger was used and of course the system's performance. So in order to resolve this issue we searched for other ways to obtain part of speech tags for our words and found the part of speech Stanford tagger. In tables 1 and 2 we show the results we got when applying the nltk and Stanford tagger to our baseline algorithm respectively for 50 reviews.

Aspect	Accuracy	Precision	Recall	F-score
Location	0.82	1.0	0.82	0.9
Rooms	0.63	0.85	0.6	0.7
Service	0.87	0.95	0.9	0.92
Price	1.0	1.0	1.0	1.0

Table 1: Table with metrics for Baseline Algorithm using nltk tagger

Aspect	Accuracy	Precision	Recall	F-score
Location	0.88	1.0	0.88	0.94
Rooms	0.72	0.9375	0.75	0.83
Service	0.9	0.95	0.95	0.95
Price	1.0	1.0	1.0	1.0

Table 2: Table with metrics for Baseline Algorithm using Stanford tagger

Looking at the results we get from our algorithm when we use the Stanford tagger instead of the one from the nltk package we can see a clear improvement in the precision, recall and accuracy of the system. However the reason we only used 50 reviews to compare these built-in functions was that the Stanford tagger takes a lot more time to produce labels for each word. Specifically, the nltk tagger took less than 3 minutes to perform the calculations whereas the Stanford tagger took close to 30 minutes. So even though the Stanford tagger works better (as can be seen from tables 1 and 2) we have chosen the nltk tagger for the implementation of our final algorithm.

4.2 Domain specific lexicon

Another problem area we identified when implementing our sentiment analysis system was the inaccurate sentiment values assigned to some words by the SentiWordnet corpus. This was a problem we had expected since the lexicon in SentiWordnet is not domain specific. Thus we searched for ways to adjust the values assigned to each word so that they would correctly reflect their use regarding hotels and found two solutions. One was the creation of a new lexicon by us as described in section 3.2. The other was the creation of two files one containing positive and the other negative words concerning our specific domain (hotels). These files were used to check and if necessary correct the values assigned by the SentiWordnet corpus. In tables 3 and 4 we present the results we got from our final algorithm for both scenarios when applied on 150 reviews.

Aspect	Accuracy	Precision	Recall	F-score
Location	0.86	0.98	0.86	0.92
Rooms	0.77	0.89	0.76	0.82
Service	0.81	0.93	0.82	0.88
Price	0.56	0.77	0.63	0.69
Document	0.96	0.93	0.97	0.95

Table 3: Table with metrics for Final Algorithm using Lexicon we created

Aspect	Accuracy	Precision	Recall	F-score
Location	0.91	1.0	0.93	0.96
Rooms	0.8	0.93	0.84	0.88
Service	0.86	0.96	0.89	0.92
Price	0.5	0.85	0.75	0.8
Document	0.96	0.92	0.97	0.95

Table 4: Table with metrics for Final Algorithm using Negative & Positive word files

Looking at tables 3 and 4 we can see that the implementation using negative and positive word files outperforms the one where we use the lexicon we generated. We observed that for all four aspects the results we get for precision, recall and accuracy are much better so we decided to use that approach in our final implementation.

4.3 Comparison of baseline and final algorithm

Lastly, we conducted an experiment using 400 reviews to compare the performance of our final algorithm with that of our baseline algorithm. We can see the performance of both algorithms in tables 5 and the 6.

Aspect	Accuracy	Precision	Recall	F-score
Location	0.73	0.94	0.77	0.84
Rooms	0.77	0.93	0.78	0.85
Service	0.8	0.9	0.83	0.86
Price	0.63	0.91	0.77	0.84
Documents	0.91	0.93	0.94	0.94

Table 5: Table with metrics for Baseline Algorithm using nltk tagger and SentiWordnet

In tables 5 and 6 we present the results we got from the initial Baseline algorithm and our final algorithm. Table 5 shows the precision, accuracy, recall and f-score we got when we ran the baseline algorithm using the nltk part-of-speech tagger and the SentiWordnet corpus to acquire the sentiment value for the words in a review. Table 6 shows the precision, recall, accuracy and f-score values we got when we ran the same 400 experiments with our final algorithm. This algorithm uses the Stanford parser, nltk tagger and acquires the sentiment value of each word from a combination of the SentiWordnet corpus and two files we created with specific domain negative and positive words for hotels.

Aspect	Accuracy	Precision	Recall	F-score
Location	0.85	0.97	0.89	0.93
Rooms	0.77	0.95	0.81	0.88
Service	0.87	0.98	0.91	0.95
Price	0.60	0.86	0.81	0.83
Documents	0.91	0.93	0.94	0.94

Table 6: Table with metrics for Final Algorithm using nltk and Positive & Negative word files

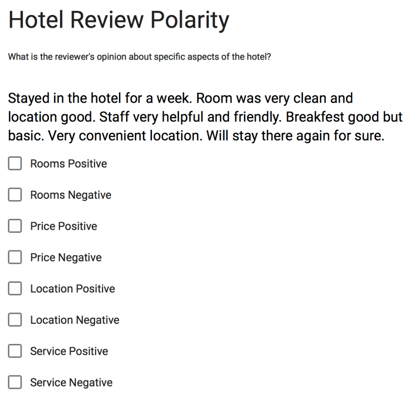
It is obvious from the results in tables 5 and 6 that our final algorithm outperforms the baseline algorithm for all aspects except the aspect value. As expected the use of natural language structural dependencies improves the precision, sensitivity and accuracy of our system. The Stanford parser allows us to better pinpoint the specific words that are associated with our aspect terms which in turn leads to more accurate assignment of sentiment values. Unfortunately, that is not the case for the aspect price where surprisingly our baseline algorithm seems to be working slightly better. This is probably a result of the limited number of reviews used for this experiment, however this was unavoidable because the Stanford tagger performs slowly and requires a lot of memory.

4.4 Human Assessment

The problem with the textual reviews is that each human apprehends the information provided differently. A sentence that may have a positive sense for someone, may have a negative sense for another. Thus we decided to create a form and ask people to identify if an aspect is mentioned in the review and if the opinion of the reviewer is positive or negative. The form [1] contains ten reviews and eight options for each review, four aspects and if the opinion about that aspect is positive or negative as shown in 4.

We got answers from approximately 20 persons and even though the number of the reviews assessed was not high enough we found inconsistency between the most common choice of the users and the rating in stars in Tripadvisor. For example one interesting thing happened with the following the sentence: *"Stayed in a king suite for 11 nights and yes it costs us a bit but we were happy with the standard of room, the location and the friendliness of the staff."* The reviewer has rated the value of the hotel with 5 stars, however most of the people who helped us with the questionnaire thought that it had a negative sense. Our system did not find sentiment for the aspect value and that happens because even though we have the word *cost* in our list of aspect synonyms the phrase dependency created by the parser is *((u'costs', u'VBZ'), u'dobj', (u'bit', u'RB'))* and apparently the word *"bit"* does not have positive sentiment

[1]https://docs.google.com/forms/d/e/1FAIpQLSed3y73WP81FnatcVF3jPkmnDnY1geYZ5nlATqhDG7GfHx_PA/viewform


Hotel Review Polarity

What is the reviewer's opinion about specific aspects of the hotel?

Stayed in the hotel for a week. Room was very clean and location good. Staff very helpful and friendly. Breakfest good but basic. Very convenient location. Will stay there again for sure.

- [ ] Rooms Positive
- [ ] Rooms Negative
- [ ] Price Positive
- [ ] Price Negative
- [ ] Location Positive
- [ ] Location Negative
- [ ] Service Positive
- [ ] Service Negative

Figure 4: Example of human assessment system

for that part of speech tag. We faced similar problems also with other reviews and it was interesting to test how people can perceive things in a completely different way. Our system seems to produce similar results with the official rating in Tripadvisor and not the one by the people who helped us.

5 CONCLUSIONS

In this section we present the insights we have gained from the procedure of implementing a sentiment analysis system.

The most important observation we made when implementing this system was the fact that all the steps in the procedure are intertwined. This causes serious problems because if even one of the tasks works less than optimally that has a negative effect on the performance of the subsequent steps and eventually on the performance of the system. This was one of the major issues we had to contend with since there are no functions that perfectly perform the preprocessing steps part-of-speech-tagging and lemmatization.

Besides the preprocessing steps another factor that plays a crucial role in sentiment analysis implementation is the need for domain-specific corpora. As we have mentioned in previous sections a general corpus like SentiWordnet that assigns sentiment values to words without taking into account the context affects a sentiment analysis system negatively. Unfortunately, there are not many existing domain specific corpora which means that to create an accurate sentiment identification system you are required to create an accompanying corpus. This is a hard and time consuming task that could be avoided if everybody shared the lexicons they generated.

Another observation we made was that exploiting the intricacies of natural languages (as we did in our final implementation) garners much better results than using generalized rules (as the one in our baseline algorithm). This may be an obvious observation but an important one. In addition, when creating our final implementation we were forced to make decisions that affected negatively our system's performance in order to keep a balance between the execution time, memory required and accuracy of the system (e.g using the nltk tagger even though the Stanford tagger worked better due to execution time issues).

Finally based on the runs we performed and the presented experiments we can conclude that the most difficult aspect for our system is the value. People usually use complicated phrases to express their like or dislike of that aspect. On the other hand the most clearly stated opinions were those concerning the service of the hotels. For example most of the reviews included words like *"friendly, helpful"* when referring to the service of a hotel.

6 FUTURE WORK

In this section we list some tasks we would have liked to include in our implementation but we were unable to do due to time constraints.

The most important issue we faced in our project was about the aspects. We created a list of aspects we want to identify in the reviews, but as expected we couldn't include every word related to each aspect. When we tried including all the synonyms of the words we thought best related to an aspect, we actually got worse results. So we created a list with what we considered to be most the important and indicative words.

Consequently the first thing we would like to investigate further would be the automatic aspect extraction task. The most well known methods for this purpose are the pLSI and the LDA [12], which both are probabilistic methods. We investigated the LDA method to some degree, but due to time limits of this project we couldn't fully implement it. However we believe that it would help us extract much more information that now is lost due to the fixed list of aspects. Besides aspect term extraction we would have liked to use more complex dependency rules. This would help to better link the aspect terms with all the descriptive words associated with them in a sentence.

Another issue that needs further improvement is the total run time of the whole process. With the code we have created till now, it is not possible to run for the 20000 reviews that our dataset contains. We have figured out that the Standford parser (both pos tagger and dependency parser) is the most consuming part of our code, so it would be interesting to either search for other parsers, or find a way to improve the time of that parser in another way. What we did for that issue was to run the parser only for the sentences that included an aspect. Another improvement could be to handle differently every sentence before passing it through the dependency parser, in order to focus in specific parts of the sentence that we want. This will lead to execution time reduction since the parser will have less work to perform.

REFERENCES

[1] Jorge Carrillo de Albornoz, Laura Plaza, Pablo Gervas, and Alberto Diaz, (2008) A Joint Model of Feature Mining and Sentiment Analysis for Product Review Rating

[2] Luigi Di Caro, Matteo Grella, Sentiment analysis via dependency parsing.

[3] Ugan Yasavur, Jorge Travieso, Christine Lisetti, Naphtali Rishe, Sentiment Analysis Using Dependency Trees and Named-Entities

[4] GrÃďbner, Dietmar & Zanker, Markus & Fliedl, GÃijnther & Fuchs, Matthias. (2012). Classification of Customer Reviews based on Sentiment Analysis. 19th Conference on Information and Communication Technologies in Tourism (ENTER). $10.1007/978 - 3 - 7091 - 1142 - 0_40$.

[5] Taboada, M. & Grieve, J. (2004). Analyzing Appraisal Automatically, Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text.

[6] Vikram Elango and Govindrajan Narayanan. 2014. Sentiment Analysis for Hotel Reviews, http://cs229.stanford.edu/projects2014.html

[7] Sharma, Richa Nigam, Shweta Jain, Rekha. (2014). Mining of Product Reviews at Aspect Level. International Journal in Foundations of Computer Science and Technology. 4. 10.5121/ijfcst.2014.4308.

[8] Lu, Yichao and Dong, Ruihai and Smyth, Barry "Context-Aware Sentiment Detection from Ratings",Springer International Publishing 2016

[9] TripAdvisor dataset used
[10] http://www.cs.cmu.edu/~jiweil/html/hotel-review.html
[11] (https://nlp.stanford.edu/software/lex-parser.shtml)
[12] Aggarwal, Charu & Zhai, ChengXiang. (2012). Mining Text Data. 10.1007/978-1-4614-3223-4.
[13] Stanford typed dependencies manual
[14] https://www.dropbox.com/s/2ytqcbbamk79oub/Final_Code.zip?dl=0