Feature Based Opinion Mining: A Modern Approach

Norbert Eke
norbert.eke@alumni.ubc.ca
UBC Okanagan

Dr. Jeffrey Andrews
jeff.andrews@ubc.ca
UBC Okanagan

Dr. Abdallah Mohamed
abdallah.mohamed@ubc.ca
UBC Okanagan

ABSTRACT

In a world where customers can buy products with a few clicks online, future customers must consider the opinions and satisfaction levels of previous customers. In order to allow one to understand what previous customers have said, the design of an automated technique that summarizes opinions of thousands of customers is desirable. A promising technique has been developed that combines continuous vector representation models, natural language processing techniques and statistical machine learning models. This technique has been tested on labelled datasets and it extracts over 80% of opinions correctly. Future research can focus on improving the technique's limitations on edge cases.

General Terms

Opinion Mining, Feature Extraction, Opinion Detection

Keywords

Word Vector Spaces, Statistical Machine learning, Natural Language Understanding, Word Embeddings

1. INTRODUCTION

Customer reviews are essential for establishing what previous customers' opinions and satisfaction levels are towards a product, entity, event or experience related to a brand or company. Nowadays, one can find hundreds of online reviews for most commercial products that are sold online. Difficulties arise when a person is trying to analyze and interpret a large number of customer reviews. Categorizing and summarizing customer opinions based on features found in a reviews is the problem commonly known as feature-based opinion mining of customer reviews. Feature-based (customer) opinion mining consists of three main components: feature, opinion and opinion mining. A feature is the characteristics or aspects of a product. An opinion is a subjective belief, which is the result of emotion or interpretation of facts and opinion mining is the detection of patterns among opinions. Feature based opinion mining is a special kind of text summarization, in which one is trying to identify specific features of the product that customers have positive or negative opinions on. Essentially, a feature-based opinion mining system provides a summary of all the positive and negative opinions found about each feature of the product present in customer reviews. This research addresses a number of concerns related to customer reviews. There are too many reviews about products on the internet. In fact there are over 26 thousand reviews posted on Yelp every minute (Shrestha, 2016). An assumption can be made that no one reads all of the reviews, and it can be backed up with some statistics: 88% of consumers form an opinion by reading up to 10 reviews (Shrestha, 2016). One can made the argument that people who only read a few reviews will not get a fully accurate representation of what everyone's opinion is about a certain product. The research problem consists of needing a way to interpret the content of millions of reviews, without reading them all. The main goal of this research project is to come up with a technique that automatically detects opinion phrases straight from the customer reviews. Such a technique provides a summary of all opinions, which serves as convenience for customers, business intelligence for companies and enhances business customer relationships.

2. PREVIOUS WORK

2.1 Overview of previous work

Hu & Liu (2004) were the first researchers to formulate the problem of opinion detection as a natural language processing problem. They named this type of problem 'feature based opinion summarization'. Hu and Liu's approach consisted of a system of algorithms that make use of several data mining and natural language processing techniques in order to extract specific features from customer reviews. Their objective was to produce a feature-based summary of a large number of customer reviews. In their paper, Hu and Liu also mentioned that they aim to find what people like and dislike about a given product. This first approach by Hu and Liu was considered the golden standard in mid 2000s, and almost all recent approaches share some common characteristic with the original work.

In 2006, Ku et al. proposed an algorithm for opinion extraction, summarization, and tracking. The work of Hu and Liu was the basis for an approach presented by Ren and Wang, who suggested that reviews could be pre-processed automatically and customers could be provided with the generalized information (Ren & Wang, 2007). Later on, Pang & Lee (2008) wrote about experimenting with natural language understanding models, various classification techniques and multi-document opinion-oriented summarization techniques.

In the first five to ten years after Hu and Liu (2004)'s work, almost every researcher approached the problem using natural language processing and understanding, sentiment classification and summarization techniques. Few researchers have looked at feature based opinion mining, as a problem which could be solved using artificial intelligence, particularly, machine learning.

In 2013, Mikolov et al. proposed two models for computing continuous vector representations of words. These models are machine learning models, which are trained to reconstruct linguistic contexts of words by converting words into multidimensional word vectors. Mikolov et al. (2013)'s model encodes the meaning of words, and relationship between words in such a way that contextual meaning of the text is preserved in the form of high dimensional vectors.

The above mentioned natural language processing approaches and many others contributed enormously towards solving the problem of feature-based opinion summarization of customer reviews. However, one might say that linguistic machine learning models could potentially offer a better solution towards more accurate feature based opinion mining. Vector representation of words models were not applied to this problem, and the use of such models could potentially decipher the positive and negative opinions or experiences customers have had.

2.2 In-depth analysis of previous approaches

In terms of previous feature-based opinion mining systems there are a variety of approaches taken by numerous researchers. Hu and Liu's approach in 2004 started out the trend of natural language processing algorithms towards detection of opinion phrases. Their approach consisted of Parts of Speech (POS) tagging, then using a version of the Apriori algorithm (Agrawal & Srikant, 1994) to obtain the frequently occurring nouns. Most frequent nouns would become candidate features, or aspects. All the candidate features need to go through two phases of candidate feature pruning. The first phase is called 'compactness pruning', and it checks candidate aspects that are more than two words and removes the non-compact, longer phrases. The second phase of the pruning process is called 'redundancy pruning' and it removes aspects/features with a p-support less than 3, where p-support is "number of sentences that a feature appears in as a noun or noun phrase" (Hu & Liu, 2004, pp 171). After all candidate features have been pruned, an opinion phrase is formed by looking up the nearest adjective of the feature. Last step in Hu and Liu's approach is to look up the polarity of the descriptor word (modifier) from a sentiment lexicon.

Popescu & Etzioni (2007) came up with a slightly different approach in 2007. Their approach was called Opine and they incorporated a 'Point-wise Mutual Information' (PMI) scoring system inside Hu and Liu's algorithm system, which resulted in a 22% improvement on opinion phrase detection. Popescu and Etzioni's Opine system "evaluates each noun phrase by computing the PMI scores between the phrase and meronymy discriminators associated with the product class." (Popescu & Etzioni, 2007, pp 341). The PMI score between phrases are calculated from web search engine result counts (Turney, 2001).

Scaffidi et al. (2007) built on this work by calculating the probability of an identified frequent noun to be a feature, which improved Popescu and Etzioni's version of a feature based opinion mining system. They called their approach the 'Red Opal'.

In 2009, a handful of researchers introduced some machine learning techniques into the problem of feature based opinion mining. Go et al. (2009) incorporated machine learning classifiers such as Naive Bayes, Maximum Entropy classifiers and Support Vector Machines to classify features, opinion phrases and their polarity. Go et al.'s work was one of the first steps made towards the transition of using more machine learning techniques instead of natural language processing techniques to solve the problem of feature based opinion mining.

In 2010, Moghaddam and Ester introduced the Opinion Digger, a feature based opinion mining system that uses data mining to identify potential aspects/features. This was one of the systems that changed the approach, by mainly using data mining tools, instead of natural language processing techniques. The Opinion Digger finds frequent nouns, then mines opinion patterns. After some patterns have been identified, it filters out non-aspect/non-feature words, then rates aspects/features using a customized version of the k-nearest neighbour (kNN) algorithm. Once all the potential features/aspects have been found, Moghaddam and Ester suggest that "sentiments are usually the nearest adjectives in the same sentence segment, which describe the quality of the aspect" (Moghaddam & Ester, 2010). In this case, quality of aspects is referred to opinion modifiers or descriptor words.

Zhao et al. (2011) approached the problem from a different perspective. Using Latent Dirichlet allocation techniques applied to some Twitter data, they were able to come up with an algorithm to extract key phrases, that could potentially be opinion phrases. Their algorithm consisted of candidate key phrase generation based on keyword ranking, then finish it off by extracting key phrases by using a 'Topical PageRank' (TPR) scoring system to rank key phrases. This approach changed the academic perspective on key phrase extraction.

Spina et al. (2012) formulated the problem as an information retrieval and opinion target identification, and they applied four different methods to solve the problem. They looked at Term Frequency - Inverse Document Frequency (TF-IDF) ratios and log-likelihood ratios to identify patterns from words in the data, then used 'parsimonious language models' (PLM) and opinion oriented methods that extract targets of opinions to generate topic specific sentiment lexicons. Spina et al.'s work should be applied to any feature based opinion mining system, since it solves the issue of what/who is the opinion target. Spina et al. reported that the TF-IDF ratios worked better that the other three methods.

Das and Kannan took a more statistical approach to the problem by looking at "topical aspects in microblogs" (Das & Kannan, 2014). Their approach involved looking at local and global indicators within the data. Global topical aspect indicators looked at uniqueness of words by measuring how strongly a phrase is correlated with a target entity. Local topical aspect indicators used Gaussian mixture models and some probabilistic models combined with the Expectation-maximization (EM) algorithm to identify observations like burstiness and diversity of phrases. In this case, burstiness is meant to measure how frequently a phrase is used, while diversity is meant to measure how differently a phrase is used.

In 2016, Ejieh proposed an algorithm called Microblog Aspect Miner (MAM), which incorporates data science techniques into a feature based opinion mining system. The MAM algorithm uses Mikolov et al.'s (2013) word2vec model to convert words into high dimensional vectors. These vectors will serve as features for a K-means clustering algorithm, in order to isolate a cluster of frequent nouns. Ejieh calculates "Aspect-Product Similarity Threshold' values, which are used to 'rank how relevant an aspect/feature is to a product" (Ejieh, 2016). Ejieh also introduced a 'subjectivity module' to detect if a post expresses an opinion, and he noted that non-opinion words will be discarded. All of these components improved the overall performance of opinion detection system, while it inspired some of the components of technique designed in this research project and described in section 4.

Finally, in 2017 Sanger et al. have looked at expanding a document's dictionary by adding in the synonyms of words used in the text. They called this technique "synonym expansion" (Sanger et al., 2017). They have also performed clustering of words, after the synonyms have been added to the document's dictionary. Their results are promising and their system of algorithms could be considered to be added inside a feature extraction natural language processing technique.

3. RESEARCH QUESTION AND OBJECTIVES

The main goal of this research project was designing a novel approach that could analyze customer reviews and gain valuable insight into customer review. Such an approach would be looking into what exactly customers are saying about a specific product, event or experience related to a brand or company. This research will focus on combining a vector representations of words model with statistical classification models and natural language processing techniques to produce a unique system of algorithms solving the feature based opinion mining problem. This modern approach applied to the problem of feature-based opinion summarization could potentially extract valuable information from customer reviews. In order to design a technique that accomplished the above goals, the approach needs to be developed and validated on one or more annotated benchmark data sets. After understanding the performance, the technique could be used to analyze new data sets of real customer reviews in which customers express their positive and negative experiences. This research project initially focused on answering the following questions:

Q1: 'What are the benefits of continuous vector representation of word models for feature-based opinion summarization of customer reviews?'

Q2: 'Can enough valuable insight be extracted from customer reviews? Can this insight be grouped or categorized into different kinds of opinions that have been said about a product/entity/event/experience related to a company ?'

Q3: 'What kind of limitations would such a "customer opinion categorizer and summarizer" have?'

4. TECHNIQUE DESIGNED

Before getting too far ahead, one needs to get familiar with opinion phrases, in order to understand the technique. Opinion phrases consists of feature word(s), like 'touch screen' and a descriptor word, like 'good'. Figure 1 illustrates the concept of opinion phrases.

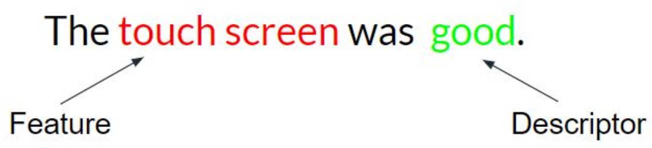
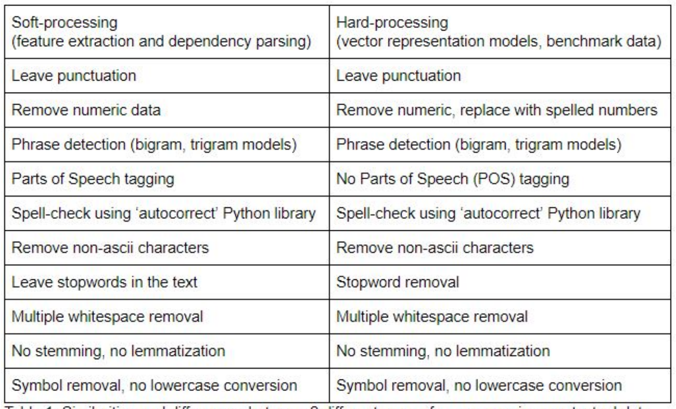


Figure 1: Opinion phrases are feature - descriptor word pairs, like 'good - touch screen'.



Soft-processing (feature extraction and dependency parsing)	Hard-processing (vector representation models, benchmark data)
Leave punctuation	Leave punctuation
Remove numeric data	Remove numeric, replace with spelled numbers
Phrase detection (bigram, trigram models)	Phrase detection (bigram, trigram models)
Parts of Speech tagging	No Parts of Speech (POS) tagging
Spell-check using 'autocorrect' Python library	Spell-check using 'autocorrect' Python library
Remove non-ascii characters	Remove non-ascii characters
Leave stopwords in the text	Stopword removal
Multiple whitespace removal	Multiple whitespace removal
No stemming, no lemmatization	No stemming, no lemmatization
Symbol removal, no lowercase conversion	Symbol removal, no lowercase conversion

Table 1: Similarities and differences between 2 different ways of pre-processing raw textual data

4.1 Textual Data

In this project two subsets of annotated datasets have been used for developing and testing the technique: Computer reviews dataset (Hu & Liu, 2004) and TripAdvisor Hotel reviews dataset (Wang et al., 2010; 2011). Another dataset named the "Adjective-Noun Word pair benchmark dataset" (Borth et al., 2013) has played an important role in this project, by serving as training data for machine learning classification models.

4.2 Text Preprocessing

There is no silver bullet algorithm for text preprocessing. There are many kinds of text processing techniques and algorithms used in natural language processing, and all of them are useful for different data cleaning and data processing scenarios. In this research project numerous text processing techniques were used, forming a couple of similar text cleaning algorithm systems. In the proposed technique text cleaning was needed for four different purposes: feature extraction (section 4.3.1), processing of words for vector representation of words model (section 4.3.2), dependency parsing (section 4.3.3) and processing of the labelled benchmark datasets (section 4.4.2). Out of these four techniques feature extraction and dependency parsing needed identical text cleaning, while the input for vector representation and processing of benchmark datasets needed a different kind of processing. Essentially, the first kind of processing is more forgiving on cleaning text, thus being called 'soft-processing' in this project, while the second kind of text processing is tougher on cleaning up text, being called 'hard-processing'. The similarities and differences can be seen in the following table.

The main difference between the two different ways of preprocessing is the punctuation removal, parts of speech tagging and stopword removal. These differences can be justified by the needs of the technique. Feature extraction and dependency parsing make use of stop-words, punctuation and parts of speech tagging, while word vector representation models and the labelled benchmark datasets do not need the three techniques above mentioned.

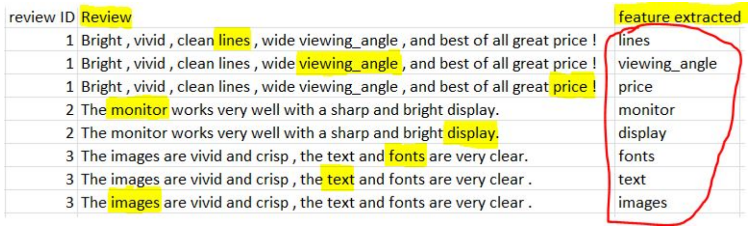
4.3 Interaction between Deep Learning and Natural Language Processing

4.3.1 Feature Extraction

Feature extraction applied in natural language processing has been wildly researched since the early 2000's, and many methods have been developed throughout the years. A small summary of previous works in this area can be found in section 2. Shortly, there are mainly 5 kinds of feature extraction methods: frequency based methods, relation based methods, hybrid methods, supervised learning based methods and topic modelling based methods. Each type of methods have pros and cons. On a personal note, hybrid approaches tend to work the best, by combining multiple approaches together and widening the scope of the algorithm. Any approach that has been tested and it performs well on benchmark datasets is acceptable. For this project's purpose an open source feature extraction package was used, created by Akshay Bansal. The author of the package described his implementation as an "application in the field of natural language processing in order to find and implement a novel algorithm to solve the problem of measuring realtime comments made by users on products" (Bansal, 2014). The whole implementation is available publicly online and it is a hybrid approach combining a relation and frequency based method. For this research project's purposes this implementation of the feature extraction technique was considered good enough in order to proceed forward with the real focus of this project, which is centered around relation vectors (section 4.4.3). Bansal's implementation makes use of natural language processing techniques like parts of speech tagging, tokenization, lemmatization and many others. In the implementation of the proposed technique, feature extraction plays quite an important role, however before even running the feature extraction algorithm, the raw input data needs to be cleaned up. For pre-processing the 'soft processing' system of algorithms is preferred over 'hard processing' since the feature extraction algorithms makes use of punctuation, stopwords and parts of speech tagging, which should not be removed from the raw input documents during the pre-processing steps. Once the raw input documents have been cleaned up, the feature extraction algorithm identifies potential features within the text, based on word frequencies and parts of speech tagging. All features extracted will be essential in one of the technique's upcoming step, which is Feature - Dependency Word Linkage described in section 4.5. A demonstration of what feature extraction does can be seen in Figure 2.

4.3.2 Continuous Vector Representations of Words Using Word2Vec

Word2Vec is Google's deep learning model that allows the computation of vector representations of words. This word2vec model takes a text corpus as input and produces high dimensional word vectors as output. In one of Google's blog posts the algorithm is described as "it constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and



review ID	Review	feature extracted
1	Bright , vivid , clean lines , wide viewing_angle , and best of all great price !	lines
1	Bright , vivid , clean lines , wide viewing_angle , and best of all great price !	viewing_angle
1	Bright , vivid , clean lines , wide viewing_angle , and best of all great price !	price
2	The monitor works very well with a sharp and bright display.	monitor
2	The monitor works very well with a sharp and bright display.	display
3	The images are vivid and crisp , the text and fonts are very clear.	fonts
3	The images are vivid and crisp , the text and fonts are very clear .	text
3	The images are vivid and crisp , the text and fonts are very clear .	images

Figure 2: Example of features extracted from sample sentences. The highlighted words represent the features identified in the sample sentence.

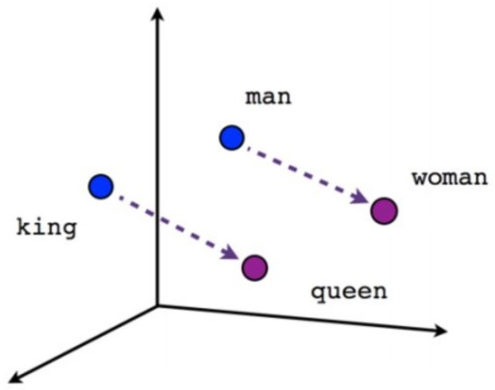


Figure 3: Three dimensional vector space, which contains the vectors 'man', 'woman', 'king', 'queen'. In this three dimensional vector space, 'man' and 'woman' are related to each other the same way as 'king' and 'queen' are related to each other, since the vectors in between them describe the male-female relationship.

machine learning applications" (Mikolov et al., 2013). These word vectors are also called word embeddings, and they are the output of the word2vec model. Essentially, with the help of the word2vec words get turned into high dimensional vectors. The word2vec model is trained to reconstructs the linguistic context of words by placing the vectors of words used in the same context close to each other. Word embeddings are also useful, since they "capture the semantic similarity between words by placing the vectors of similar words closer to each other" (Mikolov et al., 2013). Low dimensionality word embeddings, or word vectors can be visualized in so called word vector spaces. A good example of a simple, three dimensional vector space can be seen in Figure 3.

The word2vec model has 2 different architectures: cbow (continuous bag of words) and skip-gram. The difference between the two architectures is that cbow predicts the current word given the context, while skip-gram predicts the surrounding words given the current word. During the entire length of this research project Google's 300 dimensional pre-trained skip-gram model (Mikolov et al., 2013) was used. This model is a static model that was trained on close to 100 billion words and it contains 300 dimensional vectors for 3 million unique words. Throughout the process of this research project this model was used as a static lookup table for the vector representation of a specific word. What this means is that every document to be analyzed was cleaned and pre-processed, then every word from the processed documents was fed into the word2vec deep learning model, allowing the model to output a 300 dimensional vector representation for every unique word in the documents. Sadly, it was close to impossible to properly train a word2vec model just for the purposes of this research project, since these kinds of deep learning models need to learn vector representations of words from text corpuses of billions of words and there was no access to such dataset in this short period of time. Future improvements on the system of algorithms could include the use of an improved word vector representation model or the training of a model that uses a training corpus from the same domain knowledge area as the future textual data to be analyzed.

4.3.3 Dependency Parsing

Dependency parsing is a natural language processing technique, which analyzes the textual data syntactically. It looks at the structure of the sentences and builds a dependency tree showing how words relate or depend on other words in the sentence. Dependency parsing can be performed using different methods. Two different methods that have been considered and looked at. The first one makes use of context free grammar (Bird, 2009) and the second methods takes a model based approach. As a personal note, the most accurate dependency parsers come from carefully trained models, like the Stanford Parsing model family (Chen & Manning, 2014), or modern industry standard models like RedShift (Honnibal, 2013) and Spacy (Honnibal, 2017). For this project's purposes SpaCy python package's dependency parsing model was deemed to be a good fit as it builds dependency trees effectively and efficiently. In the technique implemented, before using dependency parsing the 'soft processing' technique mentioned in section 4.2 is used to clean up the documents. For dependency parsing, the 'soft processing' system of algorithms is preferred over 'hard processing' since dependency parsing makes use of punctuation and stopwords, which should not be removed from the raw input documents during the pre-processing steps. SpaCy's dependency parser (Honnibal, 2017) segments the input documents sentence by sentence, labels the parts of speech in the sentences with the help of their internal model, which then builds a dependency tree similar to the one in Figure 4.

The produced dependency tree is an essential piece of Feature - Dependency Word Linkage described in section 4.5.

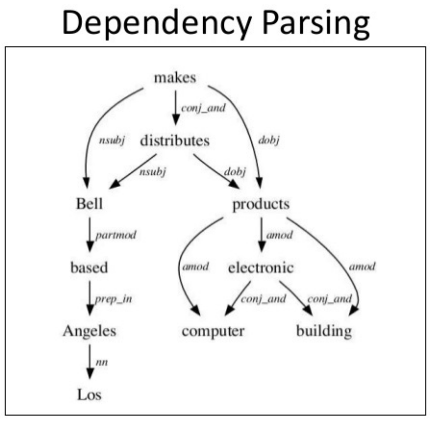


Figure 4: Sample dependency tree: arrows represent the dependency relation between words.

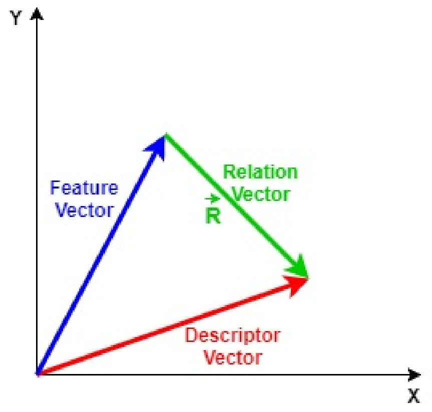


Figure 5: Two dimensional visual representation of a feature descriptor relation vector (R vector)

4.4 Supervised Learning

In order to perform supervised learning, and build 'smart systems of algorithms' that know how to identify opinion phrases, there needed to be an annotated training dataset(s), in which features and descriptors are labelled, so a model could be trained to detect opinion phrases. Such datasets have been identified in section 4.1.

4.4.1 Feature Descriptor Relation Vectors

The main focus of this research project was the relationship between feature words and descriptor words. Keeping textual data as words is not particularly useful. To gain insight into the context of the words used in the customer reviews, words needed to be converted into high dimensional vectors using word embeddings. Using the vector representation of words model, a feature word can be converted into a feature vector and a descriptor word can be converted into a descriptor vector. After converting words into vectors, the relationship between feature and descriptor words can be simplified to the difference between the feature vector and descriptor vector. In this research project, the difference of the two vectors is referred to as the feature descriptor relation vector or just simply, the relation vector. A visualization of how a relation vector can be found in shown below, in Figure 5.

These so called relation vectors have some important properties. Not every subtraction of two random vectors has the same features as a relation vector. Through careful analysis of the characteristics and features of these relation vectors it was identified that the difference of all feature and descriptor vectors is in some way different than the difference of two random vectors. The difference of feature and descriptor vectors is an 'encoding' of the semantic relationship between a feature and descriptor word. Two random words' vector representations' difference does not 'encode' the same semantic relationship as a relation vector does. Thus, there are 2 kinds of relation vectors : real and non-real. A real relation vector is the difference of vectors of valid feature and descriptor vectors. A non-real relation vector is a random word pair encoded in vectors, that makes no sense. For example a real relation vector would be the vector representation of the words 'nice' and 'desk', while a non-real relation vector would be the vector representation of any two words that do not make any semantical sense together, like 'going' and 'desk'. Knowing the difference between a real and a non-real relation vector, one can see how there needs to be a way to differentiate between these 2 types of relation vector, thus the idea of building a relation vector classification model came about (section 4.4.4).

4.4.2 Training Data Processing

In the early phases of the research project the use of properly labelled textual data, like the Computer dataset (Hu & Liu, 2004) or the Hotel dataset (Wang et al., 2010; 2011) was crucial, however the already labelled datasets needed textual data preprocessing. The 'hard processing' technique mentioned in section 4.2 was chosen for the benchmark training data processing. Keeping textual data as words is not useful at all, instead words need to be converted into high dimensional vectors using word embeddings. In order to produce good word embeddings, the removal of punctuation, numbers and stop-words are needed, thus the decision to use 'hard processing' over 'soft processing' was easy to make.

4.4.3 Feature - Descriptor labelling

After data preprocessing, labels for feature and descriptor word were needed. These labels usually come from annotated datasets, like Computer dataset (Hu & Liu, 2004) or the Hotel dataset (Wang et al., 2010; 2011). Some datasets only have the features labelled, some have both features and descriptors labelled. In case descriptors are not labelled, the inconvenient process of a person going through a subset of the labelled data and annotate the descriptor word for the already labelled feature word is needed. In this case one needs to look for the description (what is it said) about the feature word. In order to keep track of feature - descriptor word pairs present in the training documents, one needs to create a table with the columns containing review number, review content, feature word, descriptor word present in a specific review. One needs to recognize that this process is tedious, but it is needed to be done only once, when one is intending on training a Relation Vector Classification Model (section 4.4.4). This whole research project's goal is to design an automated technique that summarizes opinions from thousands of documents. In normal circumstances this process would never be needed, however this project's approach is learning relation vectors through a supervised learning, thus there needs to be some training data.

4.4.4 Relation Vector Classification Models

With the use of labelled training data described in section 4.4.2 and 4.4.3 it was possible to train multiple classification models. After some experimentation with training and testing over a dozen kinds of classification models, half a dozen were chosen as the most effective models at classifying real and non-real relation vectors. The models include two LASSO models, one Support Vector Machine (SVM) model, one Bagging model, one Random Forest model, one Linear Discriminant (LDA) model and a Gradient Boosting model. By the end of the research project there were 2 different training datasets, thus there are two models of each kind, one model trained on the first dataset, and the other trained on the second dataset. All the models and even the training data will be made public on this research project's GitHub repository (Eke, 2017). The performance of all models listed above is discussed in section 5. Figure 6 shows the logistics behind the supervised relation vector classification models.

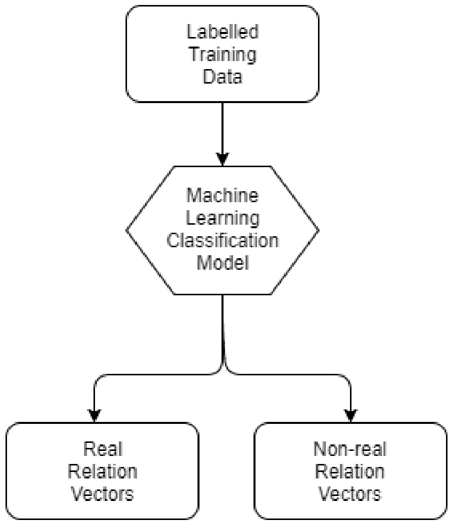

Labelled Training Data
Machine Learning Classification Model
Real Relation Vectors
Non-real Relation Vectors

Figure 6: Simple visual representation of the supervised relation vector classification model.

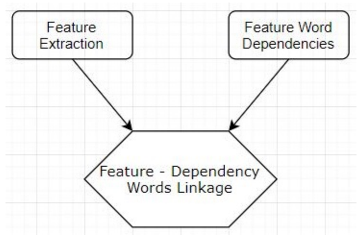


Figure 7: Structural diagram of feature - dependency word linkage.

4.5 Feature - Dependency Word Linkage

The technique's probably most important steps is called the word linkage. Knowing about feature words, descriptor words and relation vectors, one needs to keep in mind that the whole technique is about finding the proper feature descriptor word pairs. This task is getting accomplished by identifying potential relation vectors. The purpose of word linkage is simple: use the output of two previously discussed algorithms, feature extraction and dependency parsing to create candidate relation vectors. The feature extractor identifies all the features in the text, but it does not have a way to find the descriptor word. Word linkage takes care of finding the descriptor words by having the feature extractor work together with the dependency parser. The feature extractor passes all features extracted from the text to the dependency parser, which takes all features and builds a dependency tree for the sentences that each feature is present in. A structural diagram of the word linkage technique is shown, in Figure 7.

After the feature extractor and dependency parser have finished their job, a dependency tree parser produces a list

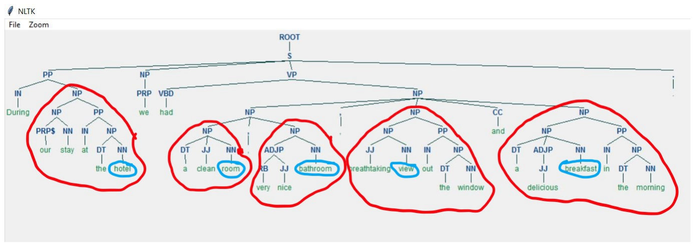


Figure 8: Example of what the dependency tree parser does. The words circled in blue are feature words, and the group of words circled in red are the list of words that the dependency parser picked out to be dependent or related to the feature word.

of words that are dependent to the feature word present in each parsed sentence. The dependency tree parser takes in a dependency tree and returns a list of words that match a custom criteria within the dependency tree. Such a criteria can be defined as return all the word nodes within the tree that are the parent node, grandparent node, sibling nodes and children nodes of the feature word's node. One can call this custom criteria as the 'tree segmentation criteria'. After lots of experimentation the best tree segmentation criteria was found to be the example criteria mentioned above. This process is shown in Figure 8.

Once the dependency tree parses has done its job, the word linkages can be created by taking each feature vector and 'linking' it with feature word's list of dependency word's vectors. To be more clear, a feature word is paired up with all of the words that depend on the feature word, thus creating many candidate feature - descriptor word pairs, all being encoded in potential relation vectors. There is one flaw: as you can see on the list of words circled in red in Figure 8, not all words dependent on the feature are descriptor words. Some word pairs' vector representations are real relation vectors, but most of them are non-real relation vectors. The next section deals with filtering out the non-real relation vectors.

4.6 Feature Descriptor Filtering

In Section 4.4.4 it was described how a few relation vector classification models were created. These models have been trained with the sole purpose of classifying real and non-real relation vectors. The classification process comes along really handy when one needs to filter out all the non-real relation vectors created in the word linkage process. The filtering process is simple: feed into one of the classification models all the candidate relation vectors created by the word linkage algorithm. The machine learning model will throw out all the relation vectors classified as non-real, keeping only the vectors classified as real relation vectors, thus finding all the real feature - descriptor word pairs. A logistics diagram of how this is achieved can be seen in Figure 9.

4.7 Opinion Phrase Polarity

Once all the real opinion phrases have been identified by the relation vector classification model, the only few steps left are classifying the polarity of opinion phrases, creating a feature breakdown of all opinion phrases, so they form clusters for each individual feature and creating a summary of all the positive and negative opinion phrases, so summarizing everything as a feature based opinion mining system. Classifying opinion phrase polarity can be considered a sentiment analysis problem. In this specific case, there are 2 approaches that one could take. The first one is to train (or use an already trained) a classification model to classify real relation vectors as a vector that expresses either positive or negative sentiments. The second approach is training (or use an already trained) a model to classify feature descriptor word pairs as word pairs that express either positive or negative sentiments. The first approach is exactly what a data science approach would look like, and the second approach is exactly what a sentiment analysis system would do. For the purposes of this project both approaches have been tried out. In figure 10, one can see the logistics of training an opinion polarity classification model.

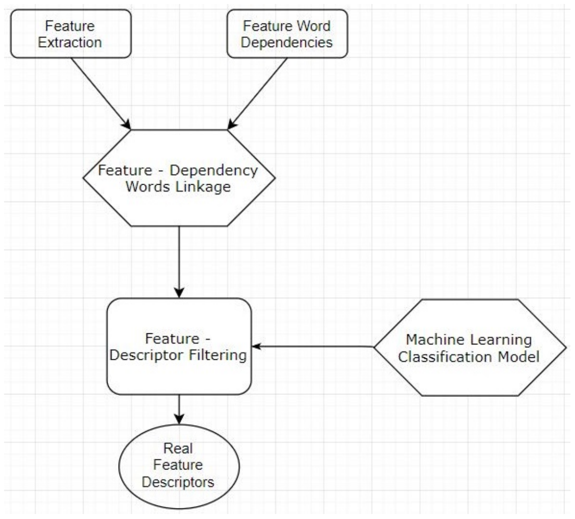

Feature Extraction
Feature Word Dependencies
Feature - Dependency Words Linkage
Feature - Descriptor Filtering
Machine Learning Classification Model
Real Feature Descriptors

Figure 9: Logistics of how the non-real feature descriptor relation vectors get filtered out using a machine learning classification model of one's choice.

For the first approach Britz's (2017) implementation of Kim's "Convolutional Neural Networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks" (Kim, 2014) was considered. This implementation was great and really straight forward to use. A CNN model for text classification was trained on Pang and Lee's publically available labelled movie review data set (Pang & Lee, 2004). For the second approach there are way too many sentiment classifiers publically available online and for this project the library called TextBlob (Loria, 2017) was chosen, since it offers a quite accurate sentiment analyzer with the execution of only a few lines of code. TextBlob has a sentiment classifier model built in the library, thus it classifies opinion phrases as positive or negative (or even neutral). After both approaches have been tried out, the second approach outperformed the first approach, thus the second approach (TextBlob) made it into the technique's final algorithms system, while the first approach (CNN model) is not used in the current version of the technique.

4.8 Feature breakdown of all opinion phrases

Once the polarity of opinion phrases was determined, in order to summarize all positive and negative opinions, one needs to have a feature breakdown of all the opinion phrases

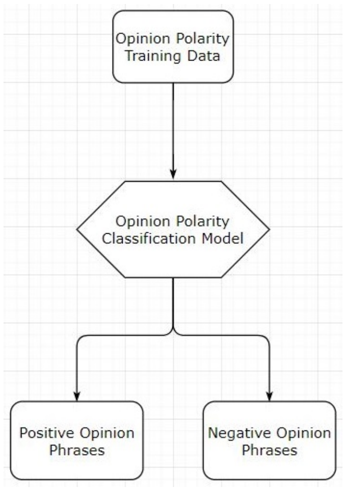
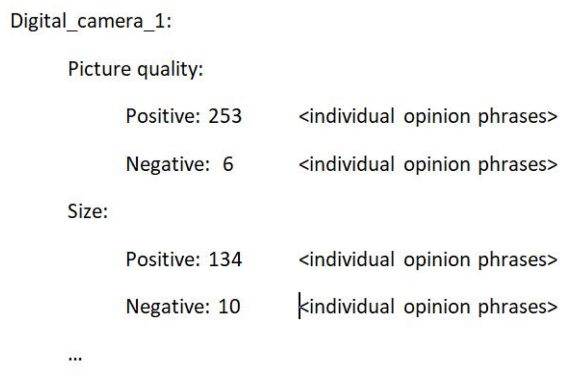

Opinion Polarity Training Data
Opinion Polarity Classification Model
Positive Opinion Phrases
Negative Opinion Phrases

Figure 10: Classification of positive and negative opinion phrases.



```
Digital_camera_1:

    Picture quality:

        Positive: 253      <individual opinion phrases>

        Negative:  6       <individual opinion phrases>

    Size:

        Positive: 134      <individual opinion phrases>

        Negative: 10       <individual opinion phrases>

    ...
```

Figure 11: Sample output from a Feature based Opinion Mining System.

classified under the feature that they belong to. This can be accomplished two ways: cross-referencing the list of features identified by the feature extractor, with each opinion phrases' feature word, then creating a list of opinion phrases belonging to each feature extracted or clustering all relation vectors representing the opinion phrases. For the purposes project the easier and more straight forward approach, the cross-referencing approach was implemented into the final algorithm system, with the mindset of keeping things simple. The clustering of opinion phrases is possible and it was done as exploratory data analysis right at the beginning of the research project. A detailed description of such analysis can be found in section 5.1.

4.9 Feature Based Opinion Mining System

Finally, figure 11 shows the summary of all the results. This type of feature breakdown summary is the output of a real feature based opinion mining system.

4.10 The whole system of algorithms and models all together

To summarize the whole technique in a high level, one paragraph overview, the raw customer reviews get preprocessed in two different ways, then a feature extractor extracts all potential features. After feature extraction, a dependency parser is applied to find a list of dependent words to each feature. All words get converted into high dimensional vectors, with the help of word embedding models. Labelled training data gets pre-processed, then it gets fed into multiple machine learning classification models, which learn the characteristics of opinion phrases in form of relation vectors. After doing so, the feature - dependency word linkage connects each feature with all of its dependency words to form potential relation vectors. The 'Feature - Dependency Word Linkage' creates many non-real feature descriptor relation vectors, which need to be filtered out. With the help of the already trained relation vector classification models most of the non real relation vectors can be filtered out, leaving only the real feature descriptors. Once all the real feature descriptors have been found, the polarity of opinion phrases is getting classifies as positive or negative. Finally, using a feature breakdown, a summary of all the positive and negative opinions found about each feature of the product present in the customer reviews is created. In figure 12 one can see how all of the previously mentioned algorithms and techniques work together to form the system of algorithms, that make the whole technique work.

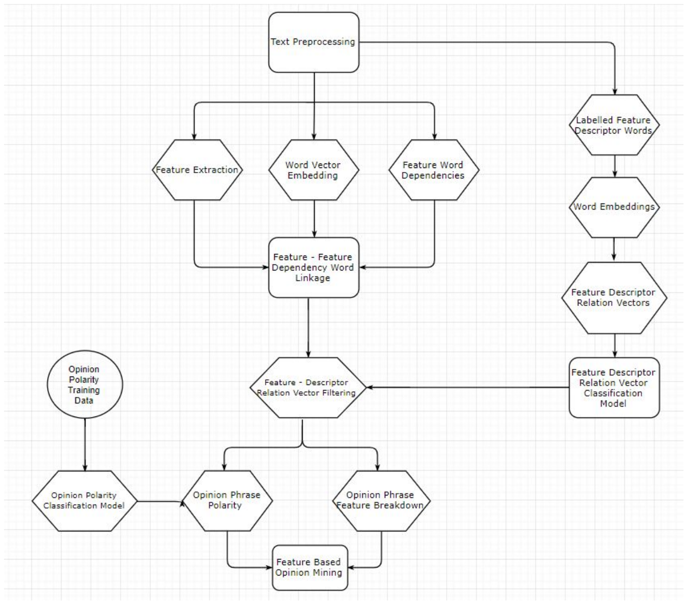

Text Preprocessing
Labelled Feature Descriptor Words
Feature Extraction
Word Vector Embedding
Feature Word Dependencies
Word Embeddings
Feature - Feature Dependency Word Linkage
Feature Descriptor Relation Vectors
Opinion Polarity Training Data
Feature - Descriptor Relation Vector Filtering
Feature Descriptor Relation Vector Classification Model
Opinion Polarity Classification Model
Opinion Phrase Polarity
Opinion Phrase Feature Breakdown
Feature Based Opinion Mining

Figure 12: Diagram of the whole system of algorithms interacting with each other to create a modern Feature based Opinion Mining system.

5. DISCUSSION AND RESULTS

In the early phases of the project, trying to understand how the vector representations of words behaved in the vector space was crucial for the progression of the research. Since usually the word vectors consist of a few hundred dimensions, there is little chance to fully understand these specific vector spaces. Data visualization techniques are advanced enough to be able to deal with high dimensional data. Dimensionality reduction and high dimensional data visualization techniques like Classical Multidimensional Scaling, Principal component Analysis, Local Linear Embedding

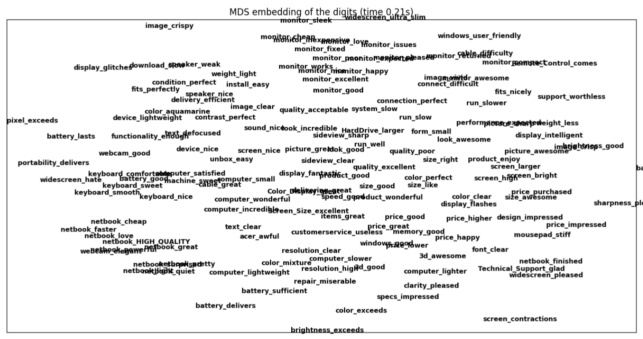


Figure 13: Classical Multidimensional Scaling applied to 300 dimensional relation vectors to project the vector space to 2 dimensions.

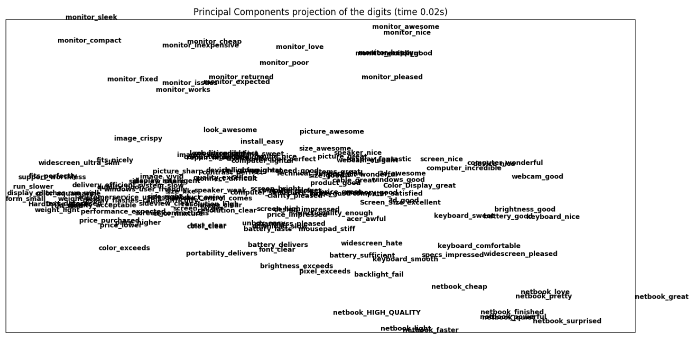


Figure 15: Principal Component Analysis applied to 300 dimensional relation vectors to project the vector space to 2 dimensions.

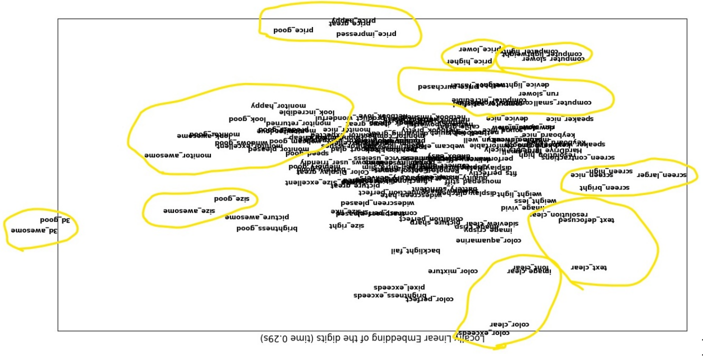


Figure 14: Locally Linear Embedding applied to 300 dimensional relation vectors to project the vector space to 2 dimensions.

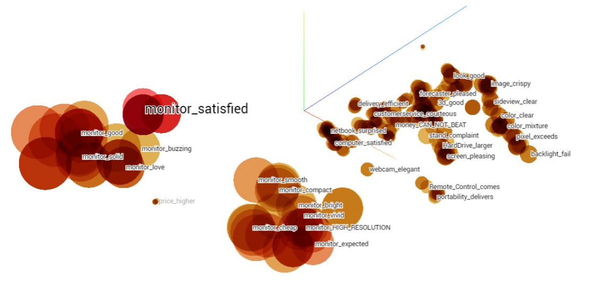


Figure 16: t-Distributed Stochastic Neighbor Embedding (t-SNE) applied to 300 dimensional relation vectors to project the vector space to 3 dimensions.

and t-Distributed Stochastic Neighbor Embedding (Van der Maaten, 2015) allow to catch a glimpse of how a projection of a high dimensional word vector space might look like. A few of these visualizations of the relation vectors from the Computer dataset can be seen in the next few figures.

The figures above give a sense of how the 300 dimensional vector space filled with relation vectors might look like. One can notice how the opinions about the same features are close together in the vector space. This can be seen in the circled groups on Figure 14, where several clusters are clearly visible. Figure 16 shows some of the groups that were discovered, when the technique called t-SNE "arranges the points in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points" (Van der Maaten, 2016). The data suggests that the best high dimensionality data visualization tool comes from Google's TensorFlow team, who made publically available the Embedding projection tool (Smilkov et al., 2016). Using this tool the visualization of the high dimensionality data became enjoyable and interpretable.

5.1 Exploratory Data Analysis

Since this section contains only exploratory analysis of a subset of the data, no detailed results will be shown, only conclusion drawn from the data and characteristics learned from it. Looking at figures 13 to 16, one can only see big clusters of words, some closer, some farther away from other words. Naturally one's first instinct would be to apply clustering on the word pairs seen in the images to see what kind of groups emerge from the vector space data. There were two different approaches towards clustering: word vector based or relation vector based clustering. Both approaches have been tried out with the sole purpose of exploratory data analysis, just to find out more about the vectors in these word vector spaces. Firstly, all unique word's vector representation taken from the Computer dataset (Hu & Liu, 2004) have been considered. A word similarity matrix has been created, where each word's similarity was compared with every other word. Cosine similarity was used to measure the similarity between high dimensional vectors, as this measurement is pretty common to be used with the word embeddings. Alternatively Euclidean distance can be used as well. Based on the word similarity matrix, the most common unsupervised clustering algorithms have been used in order to find groups within the word vectors. Hierarchical Clustering, K-Means Clustering, Affinity Propagation Clustering, Partitioning Around Medoids and Mixture Model-based Clustering was tried. The only results, where the groups within the data made any sense were achieved by Mixture Model-based Clustering. The Computer dataset, having 1763 unique clean, pre-processed words was shown to have 6 groups within the data. The groups can be described vectors that encode connection words, nouns, verbs, descriptive words, numbers and technological jargon words. These 6 groups made a lot of sense, so automatically there was an idea of trying to find vectors that connect vectors from the

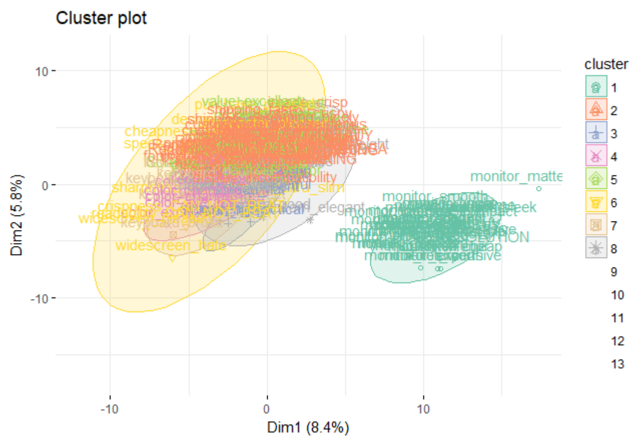


Figure 17: First dimension plotted against the second dimension from the 300 dimensional labelled relation vectors dataset. Clusters can be difficult to see, but some of the clusters clearly show all the opinions about one specific feature.

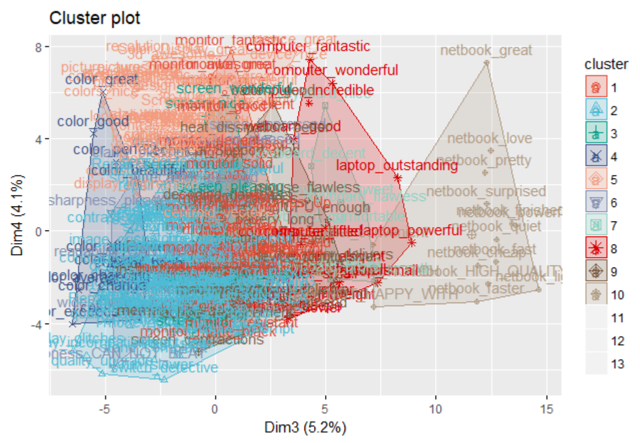


Figure 18: Third dimension plotted against the fourth dimension from the 300 dimensional labelled relation vectors dataset. Clusters can be difficult to see, but some of the clusters clearly show all the opinions about one specific feature.

nouns group with vectors from the descriptive words group or connect vectors from tech jargon group with the vectors from the descriptive words group. This is how the whole concept of feature descriptor relation vectors, described in section 4.4.1, came up. This exploratory data analysis process seemed effective, since the concept of relation vectors was born. The next step seemed like the clustering of all valid relation vectors, that came from the labelled feature-descriptor discussed in section 4.4.3. After running clustering algorithms on the 300 dimensional feature-descriptor relation vectors, the results from Mixture Models based Clustering were the most consistent with the labelled dataset. 13 groups emerged from the high dimensional relation vector data, which can be somewhat visualized with 2 dimensional projections in figures 17 and 18.

The 13 groups within the data can be described as the main 13 features in the data: monitor, screen, color, specs, quality, keyboard, computer, hardware, netbook, price, brightness, viewing angle and some miscellaneous features. From this exploratory clustering analysis one can conclude that even if in a 2 dimensional projection the groups are not completely visible, in the original 300 dimensional vector space all groups within the data are separated groups of opinions. This mixture model based clustering technique can be extremely useful if one would like to create a feature breakdown of the opinions within some textual documents, like in section 4.9.

5.2 Discussion about ideas/techniques that did not worked

After the exploratory data analysis phase of the project, it was noted that the main difficulty is the unknown structure, content and limitations of the word embeddings being in 300 dimensions. There were some ideas towards feature extraction using artificial neural networks, dimensionality reduction or classification models. The artificial neural network idea was scrapped, since it would have been uninterpretable, but the last 2 ideas progressed through towards experimentation.

Dimensionality reduction techniques were applied to the Computer dataset in almost an exploratory data analysis sense, but there was a clear goal of finding features. Repeating the same approach as in section 5.1 with clustering analysis, two different ways the data was analyzed was word vector based and relation vector based dimensionality reduction. Most techniques like Principal Component analysis, Singular Value Decomposition, Factor Analysis, Multidimensional Scaling and t-Distributed Stochastic Neighbour Embedding reduced the high dimensional data significantly. With the word vector based dimensionality reduction analysis, most techniques reduced the 300 dimensional data to about 200 dimensions, with about 90% of variance retained, while the relation vector based dimensionality reduction analysis narrowed down most relation vectors between 70 and 85 dimensions. This was a great achievement, but the only use of these reduced dimensions was the idea of feeding them into classification models, to serve as features extracted from the high dimensional vectors. After doing so, it was noticed that most of the models trained on these features were not as accurate as the models that were trained on the entire 300 dimensions. Models discussed in section 4.4.4 like LASSO, which is also known as penalized regression model, or a Support Vector Machine (SVM) model seemed to be doing a better job at classifying vectors as real or nonreal relation vectors than the previously mentioned models trained on features extracted. One can conclude that the inner mechanism of an SVM model and the model selection that a LASSO model is capable of are more effective than features extracted from dimensionality reduction techniques, at least in the case of high dimensional feature descriptor relation vectors. The only plausible explanation that can be concluded from this specific experiment is that by performing dimensionality reduction there are certain aspects of the high dimensional data that get lost, thus the model trained on such features extracted are not as accurate, since some of the meaning from the original data was lost. Dimensionality reduction ended up being the technique that originally was thought to have a large impact towards identifying relation vectors, but at the end it ended up not being a consistent enough tool to be used to identify relation vectors, just based on the loadings that it outputs.

Another idea was the use of exploratory factor analysis to look at observed variables, like word vectors, word vector lengths, distances, similarity measurements and angles between word vectors to identify some unobserved factor(s) within the word vector space that can explain the relationship between feature vectors and descriptor vectors. Some efforts were made to conduct such experiment, but no findings can be shown towards any factor being identified. Future work can be focused towards a more in-depth look at such a research angle.

A more interesting idea was to look at the relationship between synonyms in the textual data, and try to infer some meaning based on replacing an uncommon word, with itâĂŹs most common synonym. Such technique is called synonym replacement. The total opposite of synonym replacement is synonym expansion, where every wordâĂŹs synonyms are looked up and when doing inference on a word, its most common synonyms could be taken into account in order clarify or improve the contextual meaning of the word. These two ideas seem like a really good approaches in theory, but in practice, when implemented, they could get really out of hand by doing more harm than improvement. In case of the synonym expansion, large numbers of synonyms added to the dictionary of the text could further complicate the analysis, while some synonyms are contextual, thus they do not always improve the inference on a sentence. In this project a judgement call was made to leave out these two techniques from the final product. Future work can be focused on improving the current technique by adding a combination of synonym expansion and synonym replacement techniques to the current work.

Even after these experiments the goal of this research project was the same as before: design a technique that identifies opinions in the textual data. At the end of these experiments only one major step was made: the discovery of relation vectors and their characteristics. The next logical step would be to go from a feature word to all the descriptor words that describe that specific feature. The means of how to do this step was still unknown. A simple approach came up: build a model taken in a feature vector and estimates the relation vector in order to find the descriptor vector, which points to the descriptor words of a specific feature. Really easy idea, which seems theoretically possible, but practically, after a few experiments it seemed impossible. A feature vector represents the vector encoding of a noun, like 'car'. There were not any obvious ways to search around a very high dimensional vector space for vectors that connect the feature vector with some vector encodings of some descriptor words like 'shiny', 'fast' or 'nice'. Nothing really worked in terms of using some technique or algorithm that finds vectors, which represent words that were used to describe what someone's opinion is about something. Without any concrete examples or features of some vector no supervised model could be trained, and an unsupervised model was out of question, thus this simple approach was let go.

A different approach came up, which tried to use length, distance, angle and similarity measurements between word vectors as characteristics in order to predict the relation vector between a feature and a descriptor. Again, probably a great idea, but firstly there needs to be some experimentation, to see if it will work. Three different kinds of experiments were proposed: angle analysis, centroid vector analysis, and feature - descriptor distance analysis. Angle analysis was all about trying to identify any pattern or rules that determine the orientation of a relation vector that connects a feature vector and a descriptor vector. After careful analysis, the only valid conclusion that could be drawn was the observation that in most cases the angle between feature vectors and descriptor vectors in mostly between 75 and 85 degrees. This was mostly a trend, not a concrete fact, so no real hypothesis test could really support this to be a golden rule. Centroid analysis was an experiment where all the feature vectors were treated a big cluster, and all the descriptors were treated as a separate cluster. First the mean vector of both groups was found, then the two mean vectors were connected by a so called centroid relation vector, in order to try to generalize the relation vectors that connect the feature and descriptor clusters. It turned out that this centroid vector was a really inaccurate representation of general relation vector, since it was way off in terms of direction from all the other relation vectors connection a feature and a descriptor vector. Finally the distances between feature and descriptor vectors were analyzed and no meaning was inferred, since the word vectors depend on the word2vec modelâĂŹs training data. Using a different model would give totally different distances between certain vectors, so no progress was made. Without some really strongly correlated variables and stable measurements there was no possible way to estimate parameters for the coordinates of a high dimensional vector, thus this approach was thrown out as well. The final approach, that ended up working properly was the feature - dependency word linkage technique described in section 4.5 .

5.3 Discussion about the final technique

All aspects of the final technique designed have been explained throughout section 4. This section discusses the highlights of some key details that need a more in depth explanation. Firstly there were extensive experimentations and tests run on different combinations of text pre-processing techniques to be used to clean the textual data. Some of the techniques cause too much 'damage' to the original data by alternating it into something more general, that takes away small parts of the overall meaning of the data. Some of these techniques include lemmatization, stemming, phrase detection, stopword removal and conversion of all data to lowercase letters. Lemmatization and stemming are really similar by working towards the same purpose of finding the root a word (i.e 'houses' -> 'house', or 'feeding'-> 'feed'). Of course, there are significant differences between stemming and lemmatization, but both of them are useful in certain scenarios. The main problem that comes up is when cutting down the suffix of some word, the context of the word might become ambiguous. Nonetheless deep learning algorithms like word2vec have a different vector representation for the root of a word, than for some word that has a suffix added to the root. Phrase detection is really useful, whether it is bigram or trigram model architecture, but sometimes ends up damaging the data by making a phrase out of a word and a stopword, or joining a pronoun with some other word together. Some examples of this problem can be the words 'to' and 'the' transformed 'to_the' indicating a new phrase detected or the common phrase detection of 'I_am' within text. Intuitively one would say that this phenomenon is not a problem, but experiences from this research project show that an average parts of speech tagger (POS tagger) does not know what to do with oddly detected phrases like 'I_am' of 'to_the'. Stopword removal and punctuation removal is very useful in some cases, like when prepared the data to be fed into word2vec deep learning model, but very damaging in cases when dependency parsing needs to be executed, and the dependency parser relies on the punctuation and stopwords within the sentences. The last controversial technique is the conversion of all alphabetic text to lowercase letters. This technique is useful for not making a difference between words in regular, lowercase form versus when they are used as first word in the sentence and they are in uppercase, but creates massive semantic damage by all the sudden turning the company 'Apple' into the fruit 'apple', ruining the context and potentially all the meaning that could have been extracted from the data. All in all, compromises have been made and two different text cleaning techniques have been implemented. Their characteristics, also, their similarities and differences can be found in section 4.2, within Table 1.

Moving on to feature extraction, one can note that feature extraction is almost considered a solved problem, since it has been researched for a long time and the accuracy of the current best feature extractor systems are well around the 90th percentile. More in-depth previous works can be found in section 2. For the purposes of this project a frequency and relation based hybrid method's implementation was found online, as an open source project (Bansal, 2014). Of course, this implementation is definitely not the most accurate feature extractor, but it was good enough in order to make the rest of the technique work. Some of its weaknesses are the inability of identifying implicit opinions, also identifying features that are not nouns. The latter one not being a huge issue, since Moghaddam & Ester (2010) states more than 70% of features are explicit noun or noun phrases. Hu & Liu (2004) assume that most features are noun phrases. A point could be made to replace the current feature extractor with a better, more effective and more efficient implementation, if possible, an academically researched and tested implementation of feature extraction.

For word embeddings Google's quite famous 300 dimensional word2vec deep learning model (Mikolov et al., 2013) is used, but for the purposes of this project this model has one noteworthy flaw. Essentially this model is used for looking up pre-trained vector representations for each word from the input documents. These high dimensional vector have been trained on Google's training data, rather than what the input documents are. For sure one can make the argument that during Google's data driven training process the true meaning of the words are encoded into high dimensional vectors. The flaw within this current technique is that the deep learning model was not trained on product specific data. One could argue that the vector representations are not the true vector representations, since the input documents get fed into a static model, rather than a dynamic model that adjusts itself based on the current input documents given. This is a real issue that could not be addressed, but should be addressed in future work. The problem with training a product specific model is that word2vec models need millions, if not billions of words to be trained on. This very large number of training words can not be collected, if only subjective customer reviews about a specific product are considered. This issue of not being able to properly train a product specific model left no other option but to compromise and use one of the best pre-trained static models.

Looking at the process of dependency parsing, the specifics of how the dependency parser creates a list of words that depend on another word was not discussed extensively within section 4.4.3 and 4.5. What really happens 'under the hood' of this algorithms is the detection of words that dependent words based on a dependency tree traversal. A standard breath-first search (BFS) approach was implemented into the dependency parser to traverse the dependency tree created by the parsing models built inside each dependency parser. Firstly, dependency parsing is used within the Word linkage algorithm described in section 4.5, and the sole purpose of the parsing process is to determine a list of words that a morphologically dependent to a feature word. A customized mini-algorithm is implemented within the word linkage process that takes in a feature word, goes inside the dependency tree, it applies the BFS tree traversal algorithm to find the node containing this feature word. It continues by taking a look at the 'family of nodes' around the feature word node. The 'family of nodes' refers to the relationship between a node and another closely connected node, just a parent or child nodes. After long and careful experimentation, two sets of 'node families' have been found to work really well. The first one being the set of 'grandparent', 'parent' nodes, joined with the 'sibling' and 'child' nodes of the node containing the feature word. This specific set of nodes the 'close family" of words dependent to the feature word. The other set of nodes was called the 'larger family of dependent nodes' and it consists of 'parent', 'grandparent' nodes, 'sibling' nodes, 'aunt/uncle' nodes, 'cousin' nodes, 'children' and 'grandchildren' nodes, all relationships being with respect to the feature word. It is difficult to choose in between these two types of dependency node families, since both of them are working towards two separate goals of word linkage: maximizing the number of opinion phrases found in the text, and minimizing the number of non-sense opinion phrases found within the text. The set of nodes called the 'large family of dependency nodes' takes care of maximizing the number of opinion phrases found in the text, while the set of nodes called the 'close family' minimizing the number of non sense opinion phrases found within the text, while still finding quite a lot of opinion phrases. In a perfect word, a compromise between the two sets would be apply to optimize the opinion phrases identified.

5.4 Results

In this section all research questions will be answered and some performance related results will be provided for the technique described in section 4.

The first research question was 'What are the benefits of continuous vector representation of word models for feature-based opinion summarization of customer reviews?' The answer to this question seemed unintuitive, but the concept of a relation vector answers it. Using vector representations of words allows one to capture the contextual properties and details of what is said in some reviews. Word vectors coming out of deep learning models like Word2Vec (Mikolov et al., 2013) encode the 'relationship' and 'meaning' between words in such a way that in high dimensional vector space, it can be assumed that all contextual meaning of reviews could be put together purely based on working with vector representation of words from each review. The proof of this is the concept of relation vectors, which are the difference between a feature vector and a descriptor vector, and 'encode' the relationship between feature and descriptor words (opinion phrases).

The second research question was "Can enough valuable insight be extracted from customer reviews in such a way to group/categorize different opinions that have been said about a product/entity/event/experience related to a company?" One can argue that the feature breakdown of all positive and negative opinions found in customer reviews can be considered enough valuable insight extracted from the customer reviews. One can choose to look at two different groupings of all opinions: either categorize opinions based on what feature they belong to (what feature they describe) or categorize them simply as positive and negative opinions that people have said about a specific product.

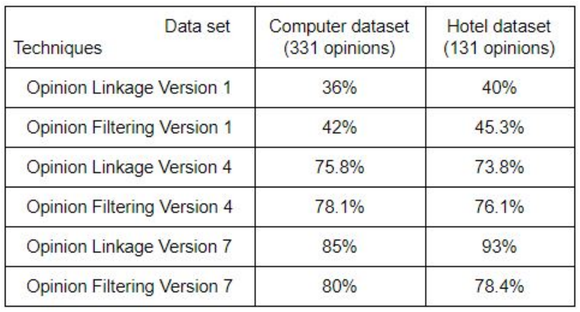
The last research question was "What kind of limitations does a feature based opinion mining system have?" A current major limitation of this technique is that it can not 'truly' find what was said about a feature in the reviews. Given a feature word, the technique can not pick out the words that were used to describe what was said about the feature. Instead, what this technique really does, is find potential words that describe what was said about a feature, then filter out the words that do not describe the feature. Throughout this research project no algorithm was able to predict a descriptor word that was used to describe the feature word. Given only a feature word, no algorithm could just work out what the descriptor word is. Perhaps a different model based learning process could be taken in order to design such an approach. Another major limitation of this technique is that the relation vector classification model uses supervised learning, meaning that it needs labelled training data in order to learn the characteristics and features of relation vectors. Perhaps future work could focus on an unsupervised approach.

This technique was developed and tested on 2 labelled datasets, Computer (Hu & Liu, 2004) and Hotel (Wang et al., 2010; 2011) reviews. The development cycle went through 7 main iterations, each time improving it by learning what goes wrong in the technique. The performance of the technique was measured based on 2 criteria. The first one is called opinion linkage (also called recall), which can be measured as a percentage, and it denotes the percentage of opinion phrases detected (or recalled) from customer reviews by the technique. The second criteria is called opinion filtering, also measured as a percentage, and it represents the accuracy of the relation vector classification models at classifying new vectors as either real or nonreal relation vectors.

At the beginning, in version 1, the accuracy of opinion linkage and filtering was measured to be only about 40% accurate. The technique went through significant changes. These changes resulted in a 70-75% accuracy by the 4th version. Finally, in the 7th version it reached the maximum accuracy of above 80%, which meant that the technique is capable of finding opinions within the customer reviews with an over 80% success rate. A quick overview of performance can be seen in the following table:

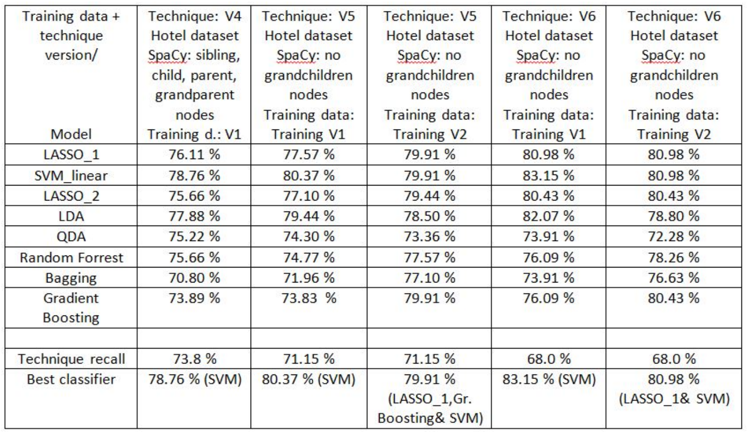
More detailed performance results can be seen in the next 4 tables below.

The above 4 tables show the progression of the technique designed through this research project. The technique version shows the different iterations of the technique, since the technique went through multiple parameter and setting



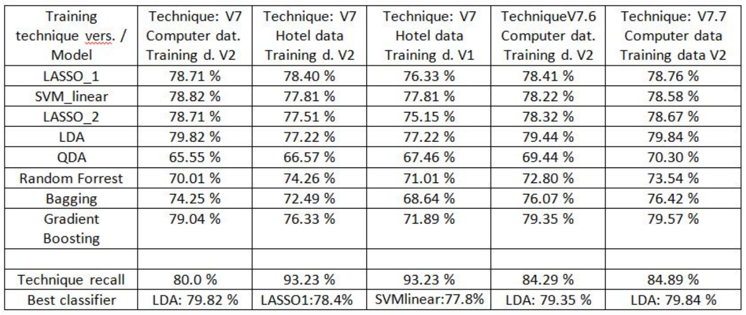
Data set Techniques	Computer dataset (331 opinions)	Hotel dataset (131 opinions)
Opinion Linkage Version 1	36%	40%
Opinion Filtering Version 1	42%	45.3%
Opinion Linkage Version 4	75.8%	73.8%
Opinion Filtering Version 4	78.1%	76.1%
Opinion Linkage Version 7	85%	93%
Opinion Filtering Version 7	80%	78.4%

Table 2: Overview of performance from different versions of the technique.



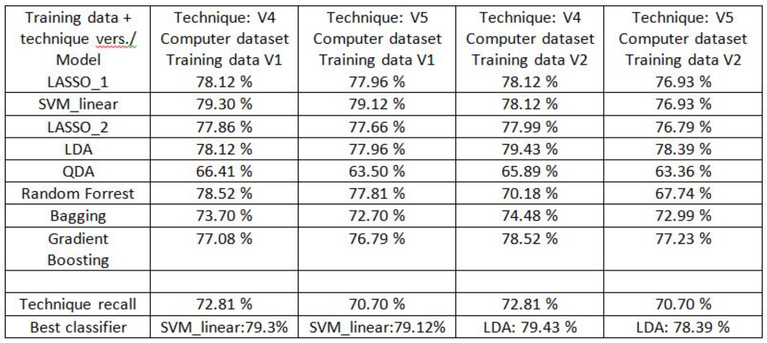
Training data + technique version/ Model	Technique: V4 Hotel dataset SpaCy: sibling, child, parent, grandparent nodes Training d.: V1	Technique: V5 Hotel dataset SpaCy: no grandchildren nodes Training data: Training V1	Technique: V5 Hotel dataset SpaCy: no grandchildren nodes Training data: Training V2	Technique: V6 Hotel dataset SpaCy: no grandchildren nodes Training data: Training V1	Technique: V6 Hotel dataset SpaCy: no grandchildren nodes Training data: Training V2
LASSO_1	76.11 %	77.57 %	79.91 %	80.98 %	80.98 %
SVM_linear	78.76 %	80.37 %	79.91 %	83.15 %	80.98 %
LASSO_2	75.66 %	77.10 %	79.44 %	80.43 %	80.43 %
LDA	77.88 %	79.44 %	78.50 %	82.07 %	78.80 %
QDA	75.22 %	74.30 %	73.36 %	73.91 %	72.28 %
Random Forrest	75.66 %	74.77 %	77.57 %	76.09 %	78.26 %
Bagging	70.80 %	71.96 %	77.10 %	73.91 %	76.63 %
Gradient Boosting	73.89 %	73.83 %	79.91 %	76.09 %	80.43 %
Technique recall	73.8 %	71.15 %	71.15 %	68.0 %	68.0 %
Best classifier	78.76 % (SVM)	80.37 % (SVM)	79.91 % (LASSO_1,Gr. Boosting& SVM)	83.15 % (SVM)	80.98 % (LASSO_1& SVM)

Table 3: Each model's performance accuracies on the Hotel dataset with version 4, 5 and 6 of the technique.



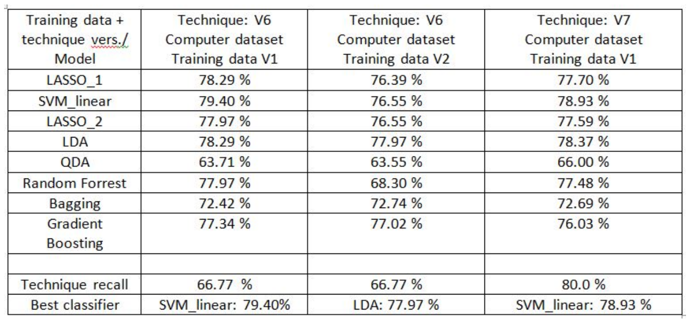
Training technique vers. / Model	Technique: V7 Computer dat. Training d. V2	Technique: V7 Hotel data Training d. V2	Technique: V7 Hotel data Training d. V1	TechniqueV7.6 Computer dat. Training d. V2	Technique: V7.7 Computer data Training data V2
LASSO_1	78.71 %	78.40 %	76.33 %	78.41 %	78.76 %
SVM_linear	78.82 %	77.81 %	77.81 %	78.22 %	78.58 %
LASSO_2	78.71 %	77.51 %	75.15 %	78.32 %	78.67 %
LDA	79.82 %	77.22 %	77.22 %	79.44 %	79.84 %
QDA	65.55 %	66.57 %	67.46 %	69.44 %	70.30 %
Random Forrest	70.01 %	74.26 %	71.01 %	72.80 %	73.54 %
Bagging	74.25 %	72.49 %	68.64 %	76.07 %	76.42 %
Gradient Boosting	79.04 %	76.33 %	71.89 %	79.35 %	79.57 %
Technique recall	80.0 %	93.23 %	93.23 %	84.29 %	84.89 %
Best classifier	LDA: 79.82 %	LASSO1:78.4%	SVMlinear:77.8%	LDA: 79.35 %	LDA: 79.84 %

Table 4: Each model's performance accuracies on the Hotel dataset with version 7 of the technique.



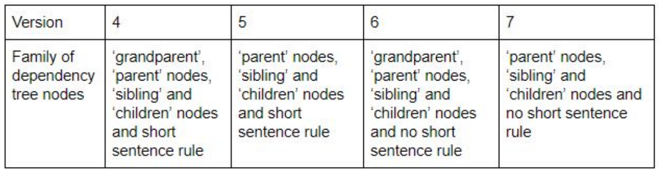
Training data + technique vers./ Model	Technique: V4 Computer dataset Training data V1	Technique: V5 Computer dataset Training data V1	Technique: V4 Computer dataset Training data V2	Technique: V5 Computer dataset Training data V2
LASSO_1	78.12 %	77.96 %	78.12 %	76.93 %
SVM_linear	79.30 %	79.12 %	78.12 %	76.93 %
LASSO_2	77.86 %	77.66 %	77.99 %	76.79 %
LDA	78.12 %	77.96 %	79.43 %	78.39 %
QDA	66.41 %	63.50 %	65.89 %	63.36 %
Random Forrest	78.52 %	77.81 %	70.18 %	67.74 %
Bagging	73.70 %	72.70 %	74.48 %	72.99 %
Gradient Boosting	77.08 %	76.79 %	78.52 %	77.23 %
Technique recall	72.81 %	70.70 %	72.81 %	70.70 %
Best classifier	SVM_linear:79.3%	SVM_linear:79.12%	LDA: 79.43 %	LDA: 78.39 %

Table 5: Each model's performance accuracies on the Computer dataset with version 4 and 5 of the technique.



Training data + technique vers./ Model	Technique: V6 Computer dataset Training data V1	Technique: V6 Computer dataset Training data V2	Technique: V7 Computer dataset Training data V1
LASSO_1	78.29 %	76.39 %	77.70 %
SVM_linear	79.40 %	76.55 %	78.93 %
LASSO_2	77.97 %	76.55 %	77.59 %
LDA	78.29 %	77.97 %	78.37 %
QDA	63.71 %	63.55 %	66.00 %
Random Forrest	77.97 %	68.30 %	77.48 %
Bagging	72.42 %	72.74 %	72.69 %
Gradient Boosting	77.34 %	77.02 %	76.03 %
Technique recall	66.77 %	66.77 %	80.0 %
Best classifier	SVM_linear: 79.40%	LDA: 77.97 %	SVM_linear: 78.93 %

Table 6: Each model's performance accuracies on the Computer dataset with version 6 and 7 of the technique.



Version	4	5	6	7
Family of dependency tree nodes	'grandparent', 'parent' nodes, 'sibling' and 'children' nodes and short sentence rule	'parent' nodes, 'sibling' and 'children' nodes and short sentence rule	'grandparent', 'parent' nodes, 'sibling' and 'children' nodes and no short sentence rule	'parent' nodes, 'sibling' and 'children' nodes and no short sentence rule

Table 7: Dependency nodes included in the word linkage process in each different version.

changes. The classification models have been training on either version 1 or 2 of a training dataset, which differ in generated and labelled non-real relation vectors. The technique's versioning differs in changes made to the dependency parser's dependency tree builder's settings and parameters, while the whole technique was tested on either the Computer (Hu & Liu, 2004) or Hotel (Wang et al., 2010;2011) benchmark dataset. As previously mentioned, there are 2 different kinds of performance measurements: opinion filtering and opinion linkage (or recall). Opinion filtering is highly affected by the classification models, since these models filter out the non-real relation vectors, keeping only the real relation vectors to be converted into valid opinion phrases found within the text. If one looks at the performance of the classification models, almost all classification models perform at an accuracy of 80% at classifying new vectors correctly. Based on the performance consistencies, one can suggest that the Lasso models, Linear Support Vector Machine Models and the LDA models are the best models, with SVM achieving even 83% accuracy on the Hotel dataset with version 6. The dependency parser's settings have a large impact on opinion linkage, since if the parser does not pick the proper dependency words to link a feature to, some opinions might not be found. In the table below one can see the setting details of the parser's dependency tree setup for each version of the technique.

In the above table, all nodes' relationship are measured with respect to the feature word. The short sentence rule refers to a small exception case, where if there are less than 10 words in a sentence, the dependency parser does not build a dependency tree, but rather returns every word in the sentence as a dependent word. This was a good idea to test with, since short sentence structures are very different than long sentence structures. Based on the experimentation with the benchmark datasets, the highest opinion linkage accuracy was achieved by the 7th version, which suggests that larger dependency tree family nodes have a higher chance of finding all dependency words, leading to more opinion phrases being detected.

5.5 Future Works

There is a need for improving the technique on certain sentence structures, that could be called edge cases. The biggest issue is the dependency parser, which suits some sentence structures better than others, thus the performance of the whole technique depends on the structure of the sentences. This technique fails to identify implicit opinions phrases, and if a sentence is syntactically incorrect, the dependency parser will not do a good job identifying dependencies, thus the word linkage algorithm will connect the wrong opinion phrases. The dependency parser does not deal well with multiple opinions listed together, especially when these opinions are separated with the word 'and' (i.e., 'the room was clean, bright and fabulous'). In this example the technique only identifies the 'clean room' opinion phrase, and does not connect 'bright' with 'room' and 'fabulous' with 'room'. The last edge case is the situation when opinion words are too far away from each other, and there is a whole sentence worth of words in between the opinion words (i.e 'The room in the hotel, which was built in 1986 by Richard Harvey, was very nice'). In this example what should be connected is 'room' and 'nice', but they are so far from each other, and in most cases the dependency parser fails to notice them.

Improvements can be made on generating better non-real relation vectors for the relation vector classification model's training data. Other improvements include the use of different dependency parsers, like Redshift (Honnibal, 2013), SyntaxNet (Alberti et al., 2017) or Stanford Parser (Chen & Manning, 2014). Synonym replacement or synonym expansion, or anything related to taking into consideration synonyms of some words would be most likely helpful towards a better interpretation of the text. Unfortunately, the current technique does not include any 'synonym manipulation'. Using a trained, dynamic word embeddings model rather than using Google's pre-trained static model (Mikolov et al., 2013a) would help by using dynamically changing vectors that adjust the meaning of some words and expressions to the data's own context and domain knowledge. Using other word embedding models like Doc2Vec (Le & Mikolov, 2014), Lda2Vec (Moody, 2016), ConceptNet Numberbatch (Speer et al., 2017), Sense2Vec (Trask et al., 2015), FastText (Joulin et al., 2016), Prop2Vec (Avraham & Goldberg, 2017), Topical Word Embeddings (Liu et al., 2015) or many others.

Also included in future works, a semi-supervised learning approach could be taken when training the relation vector classification model, using a model-based classification approach with the multivariate t-distribution, which is available as an R library called teigen (Andrews & McNicholas, 2012; 2014). Lastly, some experimentation could be performed based on Ghanimifard's work on 'illogical opinions'. Ghanimifard (2017) looked into metaphor-literal vectors, which could determine if a word pair or opinion phrase is literally possible in real life, or it is an unrealistic, fictional, illogical opinion (i.e. 'delicious airplane') . Metaphor-literal vector spaces could be considered to be fitted into a future version of the current technique.

6. CONCLUSION, SIGNIFICANCE AND IMPACT

This research project was proven to be a modern approach to the problem of feature based opinion mining by proposing a new technique that solves the problem of opinion phrase detection differently than before. This technique uses a mixture of statistical machine learning models, linguistic deep learning models, natural language processing and text mining to provide a unique combination of algorithms and techniques. Based on the results and performance presented, one can argue that this is a promising technique to automatically detect opinion phrases. Based on the long list of suggested improvements in the future works section, this technique has a long way to go until most edge cases could be covered by the same system of algorithms. Most importantly, this whole research project shows a proof of concept that such a system of algorithms could perform as well as 80% accurately at detecting opinion phrases in subjective customer review data. In terms of significance and impact, both customers and businesses can benefit with the application of such opinion mining system. This technique can provide a summary of all opinions about a specific product, which serves as convenience for customers by giving a 'big picture' view of what previous customers have said. Businesses can benefit from such a system, since business intelligence about customer opinions and satisfaction levels is priceless information. Lastly, the use of this technique could enhance the relationship between businesses and customers by offering full transparency between expectations and actual opinions about products.

References