ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEBINTELLIGENCE

Χρήση Αλγορίθμων Μηχανικής Μάθησης για την Εξόρυξη Διάθεσης (Sentiment Analysis) από Κριτικές Ξενοδοχείων στο Διαδίκτυο

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΑΝΑΓΙΩΤΗ ΣΤΑΛΙΔΗ

Επιβλέπων : Κωσταντίνος Διαμαντάρας
Καθηγητής, ΑΤΕΙΘ

Θεσσαλονίκη, Ιούνιος 2015

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEBINTELLIGENCE

Χρήση Αλγορίθμων Μηχανικής Μάθησης για την Εξόρυξη Διάθεσης (Sentiment Analysis) από Κριτικές Ξενοδοχείων στο Διαδίκτυο

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΑΝΑΓΙΩΤΗ ΣΤΑΛΙΔΗ

Επιβλέπων : Κωσταντίνος Διαμαντάρας
Καθηγητής, ΑΤΕΙΘ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 2 Ιουνίου 2015.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
....	....	....
Κωσταντίνος Διαμαντάρας	Παναγιώτης Αδαμίδης	Μιχάλης Σαλαμπάσης
Καθηγητής Α.Τ.Ε.Ι.Θ.	Καθηγητής Α.Τ.Ε.Ι.Θ.	Αναπληρωτής Καθηγητής Α.Τ.Ε.Ι.Θ.

Θεσσαλονίκη, Ιούνιος 2015

(Υπογραφή)

....

Παναγιώτης Σταλίδης

Μηχανικός Πληροφορικής Α.Τ.Ε.Ι.Θ.

Περίληψη

Η παρούσα διπλωματική εργασία μελετά μεθόδους μηχανικής μάθησης με σκοπό την εξόρυξη συναισθήματος από κριτικές ξενοδοχείων που βρίσκονται στο διαδίκτυο. Μελετήσαμε την συμπεριφορά τεσσάρων αλγορίθμων μηχανικής μάθησης, συγκεκριμένα δύο πιθανοτικούς αλγορίθμους, τον αλγόριθμο Naïve Bayes και τον αλγόριθμο Maximum Entropy και τις Μηχανές Διανυσμάτων Υποστήριξης SVM, με δύο διαφορετικούς πυρήνες, έναν γραμμικό και έναν ακτινικό (RBF). Για την εκπαίδευση των αλγορίθμων μηχανικής μάθησης, μελετήθηκαν τρία μοντέλα εξόρυξης χαρακτηριστικών, ένα μοντέλο λεξικού, ένα μοντέλο εντοπισμού κρυφών χαρακτηριστικών και ένα μοντέλο καταμέτρησης λέξεων. Παρόλο που οι αλγόριθμοι έχουν διαφορετικές προσεγγίσεις στη λύση του προβλήματος που τους ανατίθεται, σημαντικότερο παράγοντα στη διαφοροποίηση των αποτελεσμάτων είχε η μέθοδος εξόρυξης χαρακτηριστικών. Συνδυάζοντας τα χαρακτηριστικά από όλους τις μεθόδους εξόρυξης, καταφέραμε να βελτιώσουμε τα αποτελέσματα όλων των αλγορίθμων μηχανικής μάθησης.

Λέξεις Κλειδιά: Ανάλυση Συναισθήματος, Μηχανική Μάθηση, Ταξινόμηση Κειμένων, Naïve Bayes, Maximum Entropy, SVM, Κριτικές Ξενοδοχείων

Abstract

In the present thesis we tackled the problem of sentiment analysis on hotel reviews found online. Sentiment Analysis is the process of detecting the positive or negative orientation of the writer, in this case of a hotel review, towards the subject of the text excerpt, in this case hotel. We utilized both probabilistic machine learning algorithms like Naïve Bayes and Maximum Entropy, and linear classifiers like Support Vector Machines. The classifiers were investigated on several feature extracting methods. One method was to use a general purpose sentimental lexicon and aggregate the sentiment orientation to the review level. The other method was to detect hidden aspects of the words used in the review and thus detect the hidden aspects discussed in the review. A third method was the Bag-of-Words model, where each word becomes a feature for the classifier. Finally we investigated combining the feature extraction methods and that proved the most successful method.

Keywords: Sentiment Analysis, Machine Learning, Text Classification, Opinion Mining, Naïve Bayes, Maximum Entropy, SVM, Hotel Reviews

Πίνακας περιεχομένων

1

Introduction

1.1 Opinions in Online Hotel Reservation Systems

Opinions are central to almost all human activities and are key influencers of our behaviors. Our beliefs and perceptions of reality, and the choices we make, are, to a considerable degree, conditioned upon how others see and evaluate the world. For this reason, when we need to make a decision we often seek out the opinions of others. This is not only true for individuals but also true for organizations. Opinions and its related concepts such as sentiments, evaluations, attitudes, and emotions are the subjects of study of sentiment analysis and opinion mining. The inception and rapid growth of the field coincide with those of the social media on the Web, e.g., reviews, forum discussions, blogs, micro-blogs, Twitter, and social networks, because for the first time in human history, we have a huge volume of opinionated data recorded in digital forms. Since early 2000, sentiment analysis has grown to be one of the most active research areas in natural language processing. It is also widely studied in data mining, Web mining, and text mining. In fact, it has spread from computer science to management sciences and social sciences due to its importance to business and society as a whole. Tourism is a major source of income for many countries, and affects the economy of both the source and host countries, in some cases being of vital importance. Tourism is travel for recreation, leisure, religious, family or business purposes, usually for a limited duration. It is commonly associated with international travel, but may also refer to travel to another place within the same country. The World Tourism Organization defines tourists as people "traveling to and staying in places outside their usual environment for not more than one consecutive year for leisure, business and other purposes". Tourism has become a popular global leisure activity and can be domestic or international, with international tourism having both incoming and outgoing implications on a country's balance of payments [1]. From 2001 to 2011 hotel revenue from online booking services has risen from 1.4% to 8.4% of the total revenue in the hotel industry [2]. TripAdvisor has incorporated more than 200 million hotel reviews from unique customers [3] while Booking has another 46 million verified reviews [4] to mention only the two top online reservation sites.

1.2 Sentiment Analysis

In order to grasp the sentiments and opinions expressed in this multitude of hotel reviews, an extensive analysis is rudimentary. While having a lot of feedback is imperative for a business analyst to detect the reasons behind the success or failure of a hotel, manually reading this many reviews is impossible. But, a computer, while able to depict text and process volumes of information, needs a proper way to understand natural language and concepts behind words. The first, and most important step, is to find a viable and concise way to represent these concepts, called feature extraction. While a human being can interpret text even though there are spelling or grammatical and syntactical errors, a computer program doesn’t yet have this kind of expertise. And in online review text, a lot of mistakes are present, whether unintentional, or intentional as in the case of lengthened words or emoticons or in general use of slang. Also, a human reader can detect, most of the times, the scope of a negation in a sentence, or the irony present in some text. The human process to understanding text is by knowledge of words and their meaning. This knowledge comes from the study of lexicons, grammar and syntactical rules. Using an all purpose lexicon is a good start, but then you have to take into account the context of the text that one has to understand. Depending on the domain that a word is employed at, the sentiment connotation can be different. Something being “big” could be positive if “big” is what is expected or desired, or negative, if “small” is the norm of the domain, even neutral if it is part of a question. So, one could take upon the cumbersome task of creating a lexicon for each domain. Instead of manually creating this lexicon by hand, why not just let the computer, or rather an algorithm, construct the lexicon based on a set of sample text from the domain that we want to interpret?

Machine learning algorithms have been used both for general natural language processing tasks and for text classification in particular. While extensive work has been done in the field, there is still room for improvement. Not all machine learning algorithms are suitable for the task of text classification. In order to detect the best algorithm to use, one must conduct experiments on the precise domain and conditions that the algorithm is going to be used. In this thesis, we conducted experiments to determine the most appropriate set of features to use and the machine learning algorithm that best learns to separate the good reviews from the bad ones.

1.2.1 Contribution

The contribution of this thesis can be summarized as:

1. We studied techniques that can detect sentiment and opinions in text.
2. We implemented well documented feature extraction methods.
3. We propose the concatenation of features from other methods in order to improve classification results
4. We evaluated the models using well known classification algorithms
5. We conclude that with proper feature selection sentiment analysis is possible

1.3 Outline

In the following chapter we present a small survey of work in the field of Sentiment Analysis with emphasis on online hotel reviews. Then, in the third chapter, we delve in the mathematical background of the machine learning algorithms that we used. The methods for extracting the necessary features for Sentiment Analysis are discussed in chapter four. In chapter five we give details about the experiments we conducted and present the results of these experiments. The interested reader can follow up on the technical details of the setup we used for these experiments in chapter six. Finally, in chapter seven we present our conclusions and formulate our ideas for further work on the field and in chapter eight are the references.

2

Related Work

Sentiment Analysis (SA) or Opinion Mining (OM) is the computational study of people's opinions, attitudes and emotions toward an entity. The entity can represent individuals, events or topics. These topics are most likely to be covered by reviews. The two expressions SA or OM are interchangeable. They express a mutual meaning. However, some researchers stated that OM and SA have slightly different notions [5]. Opinion Mining extracts and analyzes people's opinion about an entity while Sentiment Analysis identifies the sentiment expressed in a text then analyzes it. Therefore, the target of SA is to find opinions, identify the sentiments they express, and then classify their polarity.

Sentiment Analysis can be considered a classification process with three main classification levels: document-level, sentence-level, and aspect-level SA. Document-level SA aims to classify an opinion document as expressing a positive or negative opinion or sentiment. It considers the whole document a basic information unit which would mean that is only talking about one topic. Sentence-level SA aims to classify sentiment expressed in each sentence. The first step is to identify whether the sentence is subjective or objective. If the sentence is subjective, Sentence-level SA will determine whether the sentence expresses positive or negative opinions. It was pointed out by Wilson et al. [6] that sentiment expressions are not necessarily subjective in nature. However, there is no fundamental difference between document and sentence level classifications because sentences are just short documents [7]. Classifying text at the document level or at the sentence level does not provide the necessary detail needed on all aspects of the entity, which is needed in many applications. To obtain these details, we need to go to the aspect level. Aspect-level SA aims to classify the sentiment with respect to the specific aspects of entities. The first step is to identify the entities and their aspects. The opinion holders can give different opinions for different aspects of the same entity like this sentence "The voice quality of this phone is not good, but the battery life is long".

The data sets used in SA are an important issue in this field. The main sources of data are from product reviews. These reviews are important to the business holders as they can take business decisions according to the analysis results of users' opinions about their products. The reviews sources are mainly review sites. SA is not only applied on product reviews but can also be applied on stock markets [8], news articles [9], or even political debates [10]. In political debates for example, we could figure out people's opinions on a certain election candidates or political parties. The election results can also be predicted from political posts. Also used as data sources in the SA process are social network sites and microblogging sites and they are considered a very good source of information because people share and discuss their opinions about a certain topic freely.

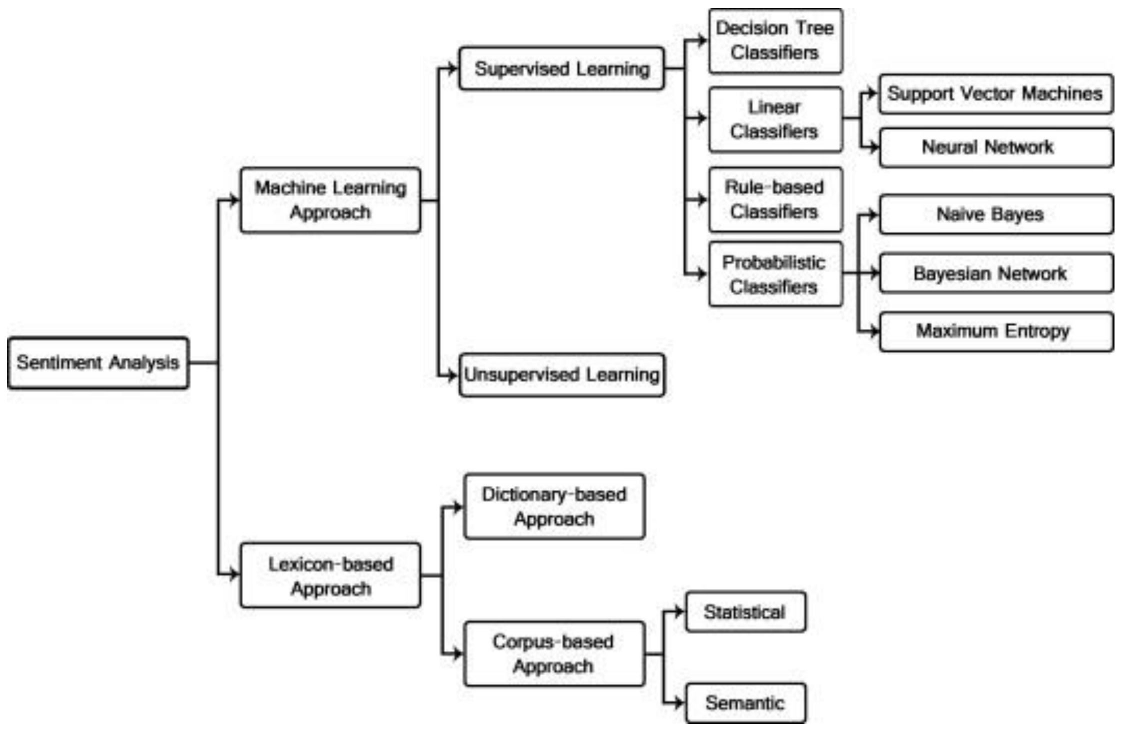


Figure 1 Sentiment Analysis roadmap

There exist two main approaches to the problem of extracting sentiment automatically. The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words or phrases in the document [11]. The text classification approach involves building classifiers from labeled instances of texts or sentences [12], essentially a supervised classification task. The latter approach could also be described as a statistical or machine-learning approach. A roadmap to SA can be seen in Figure 1.

2.1 Lexicon Based Sentiment Analysis

Dictionaries for lexicon-based approaches can be created manually, as is described in [13] and [14] or automatically, using seed words to expand the list of words [15][11][16]. Much of the lexicon-based research has focused on using adjectives as indicators of the semantic orientation of text [15][17][18][19].

First, a list of adjectives and corresponding sentiment orientation (SO) values is compiled into a dictionary. Then, for any given text, all adjectives are extracted and annotated with their SO value, using the dictionary scores. The SO scores are in turn aggregated into a single score for the text. Though lexicon based approaches have not yet managed to yield as accurate results as machine learning techniques, the can generalize their results better when used outside the domain that they were trained. Also, because one of the steps in lexicon based approaches is to annotate text with part of speech tags, the aspects that the sentiment is targeted upon can be handily detected.

2.2 Sentiment Analysis with Machine Learning

One of the many tasks that machine learning algorithms have tackled is the text classification task. In text classification, a number of features are extracted from the text in order to create a vector for each sample. Features can be words in the text, called unigrams, or groups of words, called n-grams or even constructed features, like the number of nouns, verbs or adjectives. After feature selection and transformation the documents can be easily represented in a form that can be used by a ML algorithm. Many text classifiers have been proposed in the literature implementing machine learning techniques like probabilistic models and linear classifiers. They often differ in the approach adopted: decision trees, naive-Bayes, rule induction, neural networks, nearest neighbors, and support vector machines. Although many approaches have been proposed, automated text classification is still a major area of research primarily because the effectiveness of current automated text classifiers is not faultless and still needs improvement.

One of the earliest works which used supervised machine learning methods to solve sentiment classification problem is from Pang et al. [12]. In this paper, authors used three machine learning techniques to classify sentiment of movie review documents. To implement these machine learning techniques on movie review documents, they used the standard bag of features frame work. They test several features to find optimal feature set. Unigrams, bigrams, adjective and position of words were used as features in these techniques. To reduce the number of features, they used only unigrams appearing at least four times in all document corpuses and bigrams occurring at least seven times. The results show that the best performance is achieved when the unigrams are used in SVM classifier. As they show in this paper, better performance is reached by using only presence of feature instead of feature frequency.

In [20] authors augmented bag-of-words classification with a technique which performed shallow parsing to find opinion phrases, classified by orientation and by a taxonomy of attitude types from appraisal theory [21], specified by a hand-constructed attitude lexicon. Text classification was performed using a support vector machine, and the feature vector for each corpus included word frequencies (for the bag-of-words), and the percentage of appraisal groups that was classified at each location in the attitude taxonomy, with particular orientations. They achieved 90.2% accuracy classifying the movie reviews in Pang et al.'s [12] corpus.

Ye et al. [22] incorporated sentiment classification techniques into the domain of destination review. They used three supervised learning algorithms of support vector machine, NB and the character based N-gram model to classify destination reviews. The information gain (IG) was used to selecting feature set. They used the frequency of words to represent a document instead of word presence. They found SVM outperforms the other two classifiers with an accuracy peak at about 86% when the training corpuses contain 700 reviews.

Prabowo and Thelwall [23] took a combined approach to sentiment analysis with a hybrid classifier, applying different classifiers in series, until acceptable results are obtained. If this cannot be achieved with one classifier, the system passes the task onto the next in line, until no more classifiers exist. For this, they use a combination of rule-based classification, supervised learning, and machine learning. For rule-based classification and supervised learning, authors use three different rules from existing research. Two of the used rule sets were also combined with two preexisting induction algorithms, ID3 and RIPPER, to generate two induced rule sets, which were also tested. For machine learning based classification, they used a Support Vector Machine using two pre-classified training sets, positive and negative, and have the SVM create a hyper plane to best separate the two planes. The hybrid classifier was tested on a combination of movie reviews, product reviews and MySpace comments, and yielded anywhere from 72.77% F-measure score to 90% F-measure score, depending on the corpus.

2.3 Sentiment Analysis on Hotel Reviews

Some of the work already done in SA was conducted on online hotel reviews. Datasets ranging from 1000 reviews to more than 120000 have been used. In their work, Gindl et al. [24] tested if there can be a cross domain sentiment lexicon. To test their theory, they used datasets from TripAdvisor and Amazon, and tried training the models using the one dataset and testing the performance on the other. In order to create reference measurements, they trained and tested their model using the TripAdvisor dataset which consisted of 1800 reviews. They report an F-measure of 0.79. A semi-supervised computational method for a two-pass domain-specific sentiment lexicon construction was proposed by Lau and Keung [25]. Their method achieved 0.8263 F-measure on a 103,115 review corpus of TripAdvisor. For Gräbner [26], a lexicon based approach on a TripAdvisor dataset of 1000 reviews achieved 0.61 F-measure. Using a linear kernel SVM Gamon [27] performed sentiment classification on a TripAdvisor dataset. Their method of feature reduction after using a bag-of-words model in conjunction with linguistic statistics yielded 0.67 F-measure and a 0.69 accuracy on binary classification. While developing a method to visualize sentiments about hotels, Bjørkelund et al. [28] used two methods for measuring their success. Utilizing a TripAdvisor dataset, they performed sentiment analysis both with a Naïve Bayes classifier and a lexicon based approach. Naïve Bayes reached 0.89 accuracy while the lexicon based approach achieved 0.80 accuracy.

3

Theoretical Background

.

3.1 Naïve Bayes

The first supervised learning method we introduce is the multinomial Naïve Bayes (NB) model, a probabilistic learning method. Probabilistic approaches make strong assumptions about how the data is generated, and posit a probabilistic model that embodies these assumptions; then they use a collection of labeled training examples to estimate the parameters of the generative model. Classification on new examples is performed with Bayes' rule by selecting the class that is most likely to have generated the example. The naïve Bayes classifier is the simplest of these models, in that it assumes that all attributes of the examples are independent of each other given the context of the class. This is the so-called “naïve Bayes assumption”. While this assumption is clearly false in most real-world tasks, naïve Bayes often performs classification very well. This paradox is explained by the fact that classification estimation is only a function of the sign (in binary cases) of the function estimation; the function approximation can still be poor while classification accuracy remains high [29][30]. Because of the independence assumption, the parameters for each attribute can be learned separately, and this greatly simplifies learning, especially when the number of attributes is large.

There are two event models that are commonly used: the multivariate Bernoulli event model and the multinomial event model. The multinomial NB event model generally outperforms the multivariate one, and has also been found to compare favorably with more specialized event models, as both [31] and [32] conclude.

The probability of a document *d* being in class *c* is computed as

$$P(c|d) \infty \prod_{1 \leq k \leq n_d} P(t_k|c)$$

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class *c*. We interpret $P(t_k|c)$ as a measure of how much evidence t_k contributes that *c* is the correct class. $P(c)$ is the prior probability of a document occurring in class *c*.

If a document's terms do not provide clear evidence for one class versus another, we choose the one that has a higher prior probability. $(t_1, \mathrm{t}_2, \ldots, t_{n_d})$ are the tokens in *d* that are part of the vocabulary we use for classification and n_d is the number of such tokens in *d*. For example, $(t_1, \mathrm{t}_2, \ldots, t_{n_d})$ for the one sentence document "That is an amazing hotel" might be $(amazing, an, that, hotel, is)$, with $n_d = 5$. In text classification, our goal is to find the most likely class for the document which in NB models is the maximum a posteriori (MAP) class c_{map} :

$$C_{map} = arg_{c \in C} max P(c|d) = arg_{c \in C} max P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

In order to avoid floating point underflow while computing all these conditional probabilities, it is better to perform the computations by adding the logarithm of the probabilities. The class with the highest log probability score is still the most probable since:

$$\log xy = \log x + \log x$$

and the logarithm function is monotonic. Thus maximization that is actually done in the implementation used is:

$$c_{max} = arg_{c \in C} \max[\log P(c) + \sum_{1 \leq k \leq n_d} \log \mathrm{P}(\mathrm{t_k}|\mathrm{c})\ ]$$

Each conditional parameter $\log \mathrm{P}(\mathrm{t_k}|\mathrm{c})$ is a weight that indicates how good an indicator t_k is for *c*. Similarly, the prior $\log P(c)$ is a weight that indicates the relative frequency of *c*. More frequent classes are more likely to be the correct class than infrequent classes. The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class, and c_{map} selects the class for which we have the most evidence.

For estimating the parameters $P(c)$ and $\mathrm{P(t_k|c)}$ we must calculate the relative frequency which corresponds to the most likely value of each parameter given the training data. If n_c is the number of documents in class *c* and N is the total number of documents, then

$$P(c) \; = \frac{N_c}{N}$$

We estimate the conditional probability $P(t_k|c)$ as the relative frequency of term t in documents belonging to class c:

$$P(t_k|c) = \frac{T_{ct}}{\sum_{\mathrm{t'}\in\mathrm{V}} \mathrm{T_{ct'}}}$$

where T_{ct} is the number of occurrences of term t in training documents of class c.

The problem with these estimations is that for a term t class c combination that did not occur in the training data, the value is zero. Since, to calculate the conditional probability of a document d belonging to class c, we are multiplying the conditional probabilities for all terms, for every class c where a term t never appeared, the probability will be zero, independent of how strong an evidence other terms might give. In order to mitigate this problem, we can simply add one to all counts of terms, thus modifying the equation to estimate the conditional probability to:

$$P(t_k|c) = \frac{T_{ct} + 1}{\sum_{\mathrm{t'}\in\mathrm{V}}(\mathrm{T_{ct'}} + 1)} = \frac{T_{ct} + 1}{(\sum_{\mathrm{t'}\in\mathrm{V}} \mathrm{T_{ct'}}) + B}$$

where B is the number of terms in the vocabulary.

3.2 Maximum Entropy

Maximum entropy (ME) is a probabilistic distribution estimation technique widely used for a variety of natural language task, such as language modeling, part-of-speech tagging, text segmentation and text classification. The underlying principle of ME is that without external knowledge, one should prefer distributions that are uniform. Constrains on the distribution, derived from labeled training data, inform the model where to deviate from this uniformity. The ME formulation has a unique solution which can be found by the improved iterative scaling algorithm. In text classification, ME estimates the conditional distribution of the class label given a document. A document is represented by a set of word count features. The labeled training data is used to estimate the expected value of these word counts for every class. Improved Iterative Scaling (IIS) finds a text classifier for an exponential form that is consistent with the constraints from the labeled data.

In ME we use the training data to set constraints on the conditional distribution. Each constraint expresses a characteristic of the training data that should also be present in the learned distribution. Any real valued function of the document and the class can be a feature and ME will restrict the model distribution to have the same expected value for this feature as seen in the training data. It is stipulated that the learned conditional distribution $P(c|d)$ must have the property:

$$\frac{1}{|D|}\sum_{d\in D} f_i(d,c(d)) = \sum_{d} P(d)\sum_{c} P(c|d) f_i(d,c)$$

In practice, because we do not know P(d), we use the unlabeled training data as an approximation to the document distribution and thus enforce the constrains:

$$\frac{1}{|D|}\sum_{d\in D} f_i(d,c(d)) = \frac{1}{|D|}\sum_{d\in D}\sum_{c} P(c|d) f_i(d,c)$$

So, the first step is to identify a set of feature functions that will be useful in classification, then for each feature, measure its expected value over the training data and use it as a constraint for the model distribution.

When constraints are estimated in this fashion, it is guaranteed that a unique distribution exists that has maximum entropy [33]. Moreover, it has been shown [34] that the distribution is always of the exponential form:

$$P(c|d) = \frac{1}{Z(d)} \exp\left(\sum_{i} \lambda_i f_i(d,c)\right)$$

where each $f_i(d,c)$ is a feature, λ_i is a parameter to be estimated and $Z(d)$ is the normalization factor to ensure a proper probability and can be calculated as:

$$Z(d) = \sum_{c} \exp\left(\sum_{i} \lambda_i f_i(d,c)\right)$$

When the constraints are estimated from labeled training data, the solution to the ME problem is also the solution to a dual maximum likelihood problem for models of the same exponential form. Additionally, it is guaranteed that the likelihood surface is convex, having a single global maximum and no local maxima. A possible approach for finding the maximum entropy solution would be to guess any initial exponential distribution of the correct form as a starting point and then perform hill climbing in likelihood space. Since there are no local maxima, this will converge to the maximum likelihood solution for exponential models, which will also be the global maximum entropy solution.

3.3 Support Vector Machines

Support vector machines (SVM) are based on the Structural Risk Minimization (SRM) principle [35] from computational learning theory.

3.3.1 Linear Model

The idea of structural risk minimization is to find a hypothesis h for which we can guarantee the lowest true error. The true error of h is the probability that h will make an error on an unseen and randomly selected test example. The following upper bound connects the true error of a hypothesis h with the error of h on the training set and the complexity of h [35]

$$P\left(error(h)\right) \leq train\ error(h) + 2\sqrt{\frac{d\left(\ln\frac{2n}{d}+1\right)-\ln\frac{n}{4}}{n}}$$

The bounds holds with probability at least 1-n. n denotes the number of training examples and d is the VC-Dimension (VCdim) [35], which is a property of the hypothesis space and indicates its expressiveness. The above equation reflects the well known trade-off between the complexity of the hypothesis space and the training error. A simple hypothesis space will probably not contain good approximating functions and will lead to a high training error and thus to a high true error. On the other hand a too rich hypothesis space will lead to a small training error but a high testing error, a situation commonly called “overfitting”. It would seem crucial to pick the hypothesis space with the proper complexity.

In SRM this is done by defining a structure of hypothesis spaces H_i, so that their respective VCdim d_i increases.

$$\forall i{:}\ d_i \leq d_{i+1} \text{ and } H_1 \subset H_2 \subset \ldots \subset H_i \ldots$$

The goal would be to find the index i that minimizes the error.

Let us assume the following linear function

$$h(d) = sign\{w * d + b\} = \begin{cases} +1, if\ w * d + b > 0 \\ -1, else \end{cases}$$

Instead of building the structure based on the number of features using a feature selection strategy, SVM uses a refined structure which acknowledges the fact that most features in text categorization are relevant. If all example vectors d_i are contained in a ball of radius R and it is required that for all examples d_i

$$\left|\vec{w} \cdot \vec{d_i} + b\right| \geq 1, with\ \|\vec{w}\| = A$$

then this set of hyperplane has a VCdim d bounded by

$$d \leq min([R^2A^2], n) + 1$$

It should be noted that the dimensions of these hyperplanes do not necessarily depend on the number of features, but instead depend on the Euclidian length $||\vec{w}||$ of the weight vector $\vec{w}$. This would mean that if our hypothesis has a small weight vector, we might generalize well in high dimensional spaces.

In their basic form SVM finds the hyperplane that separates the training data with the shortest weight vector. This hyperplane separates positive and negative training samples with maximum margin. Finding this hyperplane can be translated into the following optimization problem

$$\text{Minimize } ||\vec{w}||$$

$$\text{So that } \forall i: y_i \left|\vec{w} \cdot \vec{d_i} + b\right| \geq 1$$

Where y_i is +1 if document d_i belongs in the positive class, or -1 if it belongs to the negative class. These constraints require that all training samples are classified correctly.

Since this optimization problem is difficult to handle numerically, Vapnik et al. [36] uses Lagrange multipliers to translate the problem into an equivalent quadratic optimization problem for which efficient algorithms exist that can guarantee to find to global optimum. The result of the optimization process is a set of coefficients a_i^* which can be used to construct the hyperplane:

$$\vec{w} \cdot \vec{d} = (\sum_{i=1}^{n} a_i^* \, y_i \vec{d_i}) \cdot \vec{d} = \sum_{i=1}^{n} a_i^* \, y_i (\vec{d_i} \cdot \vec{d})$$

And

$$b = \frac{1}{2}(\vec{w} \cdot \overrightarrow{d_+} + \vec{w} \cdot \overrightarrow{d_-})$$

We can see that the resulting weight vector of the hyperplane is constructed as a linear combination of the training examples. Only the examples for which the coefficient ai is greater than zero contribute. These vectors are called Support Vectors and they are the training examples which have minimum distance from the hyperplane. To calculate b, two arbitrary support vectors $\overrightarrow{d_+}$ and $\overrightarrow{d_-}$) can be used.

3.3.2 Use of Kernels

To learn non linear hypotheses, SVMs can make use of convolution functions: $K(\overrightarrow{d_1}, \overrightarrow{d_2})$. Depending on the convolution function, SVMs learn polynomial classifiers ($K(\overrightarrow{d_1}, \overrightarrow{d_2}) = (\overrightarrow{d_1} \cdot \overrightarrow{d_2} + 1)^d$), or radial basis function (RBF) classifiers ($K(\overrightarrow{d_1}, \overrightarrow{d_2}) = exp(\gamma(\overrightarrow{d_1} - \overrightarrow{d_2})^2)$), or two layer sigmoid neural nets ($K(\overrightarrow{d_1}, \overrightarrow{d_2}) = \tanh(s(\overrightarrow{d_1} - \overrightarrow{d_2}) \quad + c)$).

Any convolution function must satisfy Mercer's Theorem [35] which means that the inner product of vectors $\overrightarrow{d_1}$ and $\overrightarrow{d_2}$ is computed after they have been mapped into a new feature space by a non linear mapping φ.

$$\varphi(\overrightarrow{d_1}) \cdot \varphi(\overrightarrow{d_2}) = K(\overrightarrow{d_1}, \overrightarrow{d_2})$$

4

Feature Extraction

Selecting relevant features and deciding how to encode them for a learning method can have an enormous impact on the learning method's ability to extract a good model. Much of the interesting work in building a classifier is deciding what features might be relevant, and how we can represent them.

4.1 Text Preprocessing

Reviewers, especially online, have a tendency to use informal language. In order to extract features properly, some steps need to be applied to the raw text.

4.1.1 Tokenization

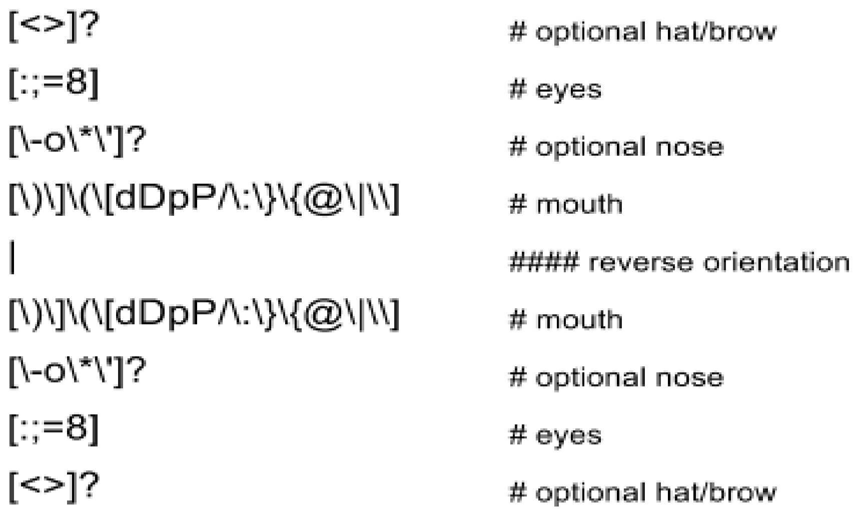
Tokenizing is the process of splitting text into smaller parts, namely tokens and is fundamental to all NLP tasks. There is no single right way to do tokenization. The right algorithm depends on the application. Tokenization is even more important in sentiment analysis than it is in other areas of NLP, because sentiment information is often sparsely and unusually represented - a single cluster of punctuation like “>:-(“ might tell the whole story [37].

The simplest tokenizer that can be used is the white space tokenizer. It simply down cases the string and splits the text on any sequence of whitespace, tab, or newline characters. But the tokenizer used by the Penn Treebank [38] and many other important large-scale corpora for

NLP is the Treebank style tokenizer. Thus, it is a de facto standard. This alone makes it worth considering, since it can facilitate the use of other tools. However, because the tokenization of raw Web text is so bad, later applications are likely to stumble also if used unmodified.

4.1.2 Punctuation

Almost all tokens that involve punctuation are split apart — URLs, Twitter mark-up, phone numbers, dates, email addresses ... Thus, emoticons are collapsed with their component parts, URLs are not constituents, and Twitter markup is lost. Emoticons are extremely common in many forms of social media, and they are reliable carriers of sentiment. In order to preserve as much sentiment expressed as possible, emoticons as well as other punctuation marks should be identified and tokenized separately. As estimated by [39], just the small list of regular expressions in Table 1 can capture 96% of the emoticon tokens occurring on Twitter. Sentiment is also often expressed by the use of multiple consecutive punctuation marks, for example "!!!!!".



```
[<>]?                          # optional hat/brow
[:;=8]                         # eyes
[\-o\*\']?                     # optional nose
[\)\]\(\[dDpP/\:\}\{@\|\\]     # mouth
|                              #### reverse orientation
[\)\]\(\[dDpP/\:\}\{@\|\\]     # mouth
[\-o\*\']?                     # optional nose
[:;=8]                         # eyes
[<>]?                          # optional hat/brow
```

Table 1 Regular expression to detect emoticons

4.1.3 Contractions

Contractions like "can't" contribute their own sentiment, as distinct from co-occurrence of can and a negation, whereas the Treebank style tokenizer splits them into two tokens. Sometimes apostrophes are used but other times they are omitted. In order to have a concise replacement strategy, contractions are replaced by their full text equivalent for example both "can't" and "cant" are replaced by two tokens "can" "not". Since contractions have been dealt with, all remaining single quotes are replaced by double quotes. Co-occurrences of "not" are then processed as other forms of negation.

4.1.4 Negations

Negation is a very common linguistic construction that affects polarity and, therefore, needs to be taken into consideration in sentiment analysis. In SA, unlike in other text classification tasks, a single negation can be the only clue to suggest a negative rather than a positive sentiment. The usual way to incorporate negation modeling into this representation is to add artificial words [39]. For example, if a word x is preceded by a negation word, then rather than considering this as an occurrence of the feature x, a new feature NOT_x is created. Since, as suggested by [12] the scope of negation cannot be properly modeled, every word is replaced until the next punctuation mark.

4.1.5 Lengthening

Lengthening by character repetition is a reliable indicator of heightened emotion. In English, sequences of three or more identical letters in a row are basically unattested in the standard lexicon, so such sequences are very likely to be lengthening. The amount of lengthening is not predictable, and small differences are unlikely to be meaningful. Thus, elongated words can be replaced by a two token sequence consisting of the word correctly spelled and a special token denoting elongation.

4.1.6 Spelling

In the previous steps, the various tokens were processed in order to detect deliberate use of misspelled words. All other misspellings can be corrected by passing the tokens from a spell checker. Any misspelled words detected can be replaced by the most common from the suggestions of the spell checker. Since the spell checker will provide a suggestion even for non-existent words, prudent use should be exercised. One way to go around this would be to check how different the suggested word is from the misspelled one and only use the suggestion if the difference is small.

4.1.7 Sentence Segmentation

Sentence boundary disambiguation (SBD), also known as sentence breaking, is the problem in natural language processing of deciding where sentences begin and end. Often natural language processing tools require their input to be divided into sentences for a number of reasons. However sentence boundary identification is challenging because punctuation marks are often ambiguous. For example, a period may denote an abbreviation, decimal point, an ellipsis, or an email address – not the end of a sentence. About 47% of the periods in the Wall Street Journal corpus denote abbreviations. As well, question marks and exclamation marks may appear in embedded quotations, emoticons, computer code, and slang. Languages like Japanese and Chinese have unambiguous sentence-ending markers.

4.1.8 Stopword Removal

Words that have the same probability to be used in all classes have no real discriminative power and can potentially detriment results of classifiers by making the feature space very large. In addition to common English language words like “a”, “the”, “to”, which have no sentiment conveyed, all words appearing in the majority of documents can be removed.

4.2 Bag of Words Model

The first step in modeling a document into vector space is to create a dictionary of terms present in documents. To do that, you can simply select all terms from the document and convert each one to a dimension in the vector space. One approach to create the document vector could be to check for the existence of a term in the current document and assign a boolean value in the appropriate dimension of the vector. Because this method is like taking the words that make up a document and putting them in bag, then check which words are in the bag, this model is called a Bag of Words (BOW).

In the English language there are some kind of words that are present in almost all of the documents. These words, like “the, is, at, on” have no real discriminative power between documents and therefore can be ignored. Depending on the corpus that is being processed, another set of words specific to the domain could be very common, for example in a hotel review corpus words like “hotel” and “room” are likely to be present in most of the reviews. Though these words are not likely indicators of expressed sentiment about the hotel that is reviewed they might be the target of the sentiment. Depending on the level of SA that is being conducted these terms might be useful for the analysis or not.

Instead of only looking for the existence or absence of a term, one could be interested in how many times each word appears in the document. Of course in a big document the likelihood that a term will appear more than once is greater. So a more useful metric could be the frequency of a term appearing in the document. A single mention of the word “bad” in text could be an indication that the review is negative, but if the reviewer keeps repeating it usually means that the term plays an important role in this document. But does this mean that if the term “bad” appears in a document ten times, it makes the document ten times more “bad”? Probably not. It is common to normalize the Term Frequency (TF) by its logarithm.

Naturally there are terms that only appear in a few documents but in those documents they appear multiple times. In order to take into account this fact, the term frequency can be normalized by the inverse frequency of the term appearing in the document corpus, which is the document frequency. The features extracted by this method are called Term Frequency * Inverse Document Frequency (TF*IDF).

Since many of the classification algorithms benefit from feature values being in the range of 0-1, another transformation to the vector can be its normalization.

The BOW model is very common in Information Retrieval, Sentiment Analysis and text classification in general. It's main drawback is that it can't capture the order that the terms were used in the document. So, two documents that are made up from the same terms but in a different order will have the same vector representation.

In order for the model to capture common phrases like "big apple" which is a nickname for New York and has a very different meaning than a big apple, terms can be consisting of more than one word. When terms consist of two consecutive words they are called bigrams, for three words trigrams and in general n-grams.

4.3 Sentimental Dictionary Model

A sentimental analysis using a dictionary model inevitably makes two assumptions: that individual words have what is referred to as prior polarity, that is, a semantic orientation that is independent of context; and that said semantic orientation can be expressed as a numerical value. In general, the SO of an entire document is the combined effect of the adjectives or relevant words found within, based upon a dictionary of word rankings (scores). The dictionary can be created in different ways: manually, using existing dictionaries such as the General Inquirer, or semi-automatically, making use of resources like WordNet [18][41].

The dictionary may also be produced automatically via association, where the score for each new adjective is calculated using the frequency of the proximity of that adjective with respect to one or more seed words. Seed words are a small set of words with strong negative or positive associations, such as *excellent* or *abysmal*. In principle, a positive adjective should occur more frequently alongside the positive seed words, and thus will obtain a positive score, whereas negative adjectives will occur most often in the vicinity of negative seed words, thus obtaining a negative score. The association is usually calculated following Turney's method for computing mutual information [11][16].

To predict the overall SO of a document, one can use a simple aggregate-and-average method: The individual scores for each adjective in a document are added together and then divided by the total number of adjectives in that document. More advanced methods would take into account adverbs, verbs, even nouns. Although the vast majority of the entries in a

SO dictionary are single words, some dictionaries allow for multi-word entries [19]. Special consideration must be made when the presence of negation or valence is detected.

Valence words can either intensify or down tone the semantic intensity of a neighboring lexical item. Some researchers in sentiment analysis have implemented intensifiers using simple addition and subtraction—that is, if a positive adjective has an SO value of 2, an amplified adjective would have an SO value of 3, and a down toned adjective an SO value of 1. One problem with this kind of approach is that it does not account for the wide range of intensifiers within the same subcategory. Extraordinarily, for instance, is a much stronger amplifier than rather. Another concern is that the amplification of already "loud" items should involve a greater overall increase in intensity when compared to more subdued counterparts (compare truly fantastic with truly okay); in short, intensification should also depend on the item being intensified.

The obvious approach to negation is simply to reverse the polarity of the lexical item next to a negator, changing *good* (+3) into *not good* (−3). This we may refer to as switch negation [42]. There are a number of subtleties related to negation that need to be taken into account, however. One is the fact that there are negators, including *not*, *none*, *nobody*, *never*, and *nothing*, and other words, such as *without*, a verb, or *lack*, a noun, which have an equivalent effect, some of which might occur at a significant distance from the lexical item which they affect; a backwards search is required to find these negators, one that is tailored to the particular part of speech involved. One other interesting aspect of the pragmatics of negation is that negative statements tend to be perceived as more marked than their affirmative counterparts, both pragmatically and psychologically [43], and that would suggest that frequency distribution of negative sentences is smaller than the affirmative ones [44].

Our implemented model makes use of the semi-automatically created sentiment lexicon SentiWordNet [45] which has been applied in different opinion related tasks, i.e. for subjectivity analysis and sentiment analysis with promising results. SentiWordNet extends the WordNet lexicon of synonyms adding sentiment values for each synset in the collection by means of a combination of linguistic and statistic classifiers. To each synset of WordNet, a triple of polarity scores is assigned i.e., a positivity, negativity and objectivity score, with the sum of these scores being always 1.

4.4 Word2Vec Model

In their paper Mikolov et al. [46] propose two novel model architectures for computing continuous vector representations of words from very large data sets, the Continuous Bag-of-Words (CBOW) model and the Skip-Gram model (SG). The CBOW model architecture is similar to the feed-forward Neural Net Language Model (NNLM) proposed in [47]. NNLM consists of input, projection, hidden and output layers. At the input layer, N previous words are encoded using 1-of-V coding, where V is size of the vocabulary. The input layer is then projected to a projection layer P that has dimensionality N by D, using a shared projection matrix. For CBOW the non-linear hidden layer is removed and the projection layer is shared for all words. In order to minimize computational complexity, there is use of hierarchical softmax where the vocabulary is represented as a Huffman binary tree, since Huffman trees assign short binary codes to frequent words, and this reduces the number of output units that need to be evaluated. The target of the system is to predict the use of a word given a window of previous and future words. Though the SG model architecture is very similar, the model instead of predicting the current word based on the context, tries to maximize classification of a word based on another word in the same sentence. More precisely, by using each current word as an input to a log-linear classifier with continuous projection layer, the model predicts words within a certain range before and after the current word. [46] found that increasing the range improves quality of the resulting word vectors, but it also increases the computational complexity. Since the more distant words are usually less related to the current word than those close to it, they give less weight to the distant words by sampling less from those words in their training examples.

The word representations computed using either CBOW or SG, are very interesting because the learned vectors explicitly encode many linguistic regularities and patterns. Somewhat surprisingly, many of these patterns can be represented as linear translations [48].

Following these successful techniques, researchers have tried to extend the models to go beyond word level to achieve phrase-level or sentence-level representations [49][50][51][52][53][54]. While using a sophisticated and complex approach could produce better representation, it is outside the scope of this thesis.

Instead, our model uses a simple approach, the weighted average of all the words in the document. For every word in the dictionary we compute the word representation by using the CBOW model and we construct a dictionary matrix where one dimension is the term dictionary and the other dimension is the hidden aspects of each word. The dot product of a document's TF*IDF vector with the dictionary matrix will produce the document hidden aspects.

5

Evaluation

So far we have presented the mathematics behind the classifiers that we are using and the methods in order to extract the features to train those classifiers. In order to evaluate the various features and to determine which set of features is better suited for each classifier, we run a number of sentiment classification experiments.

5.1 Measurements

Generally, the performance of sentiment classification is evaluated by using four metrics: Accuracy, Precision, Recall and F-measure, though other metrics have been proposed in literature. Accuracy is defined as the percentage of correct classifications.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative}$$

Precision is the percentage of the samples classified in a class that actually belong in this class.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall is the percentage of samples that belong in a class that were classified correctly.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

While accuracy can represent the effectiveness of a model very nicely, if the sample is not balanced, then the accuracy measurement will be skewed in favor of the overrepresented class. Precision is a measure of trust in the system. The higher precision is, the more certain we can be that the decision of the classifier is correct. On the other hand, recall measures the systems effectiveness. If recall is low, then many samples that belong to a class pass unnoticed. A measurement that can encompass the system's performance is F-measure, sometimes called F1-measure or F1-score. It is defined as:

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The above measurements are calculated for every class that the model has to separate and the system is then ranked overall with the mean of the measurements.

5.2 Dataset Description

The applied corpus comprises of customer reviews from TripAdvisor [3], a major web 2.0-platform with focus on travel and vacation services. Customers can book, rank and review hotels, flights and restaurants. The focus of the portal is to filter content, based on rankings that are derived from user ratings. Thus, rankings are split into several categories, like value, rooms, location, cleanliness and sleep quality. Available rating categories are determined by the type of the reviewed object. A rating scale contains five values, ranging from 'terrible' to 'excellent'. These values are further referred to as 1star to 5star. A separate mandatory overall rating summarizes the total customer satisfaction. Finally, the natural language part of the review comprises a title and a text. The title is displayed in quotation marks and users are invited to use concise formulations, like "We loved it and we'll be back!", or "There were things I hated.". The text is of variable size. The used corpus from TripAdvisor is restricted to reviews of hotels located in Crete, Greece, written in English. Each record contains a hotel category, the overall rating, the title and the review text. Furthermore, the entries of the subcategories: value for money, rooms, location, cleanliness, service, and sleep quality are available. These subcategories may contain null values denoting that the user didn't care about that detail. In total, the corpus comprises of over 60.000 reviews. Texts and ratings were automatically extracted from TripAdvisor.

Rating	1star	2star	3star	4star	5star	Total
Number of Reviews	2839	3298	6828	17248	34975	62189

Table 2 Number of Reviews per user Rating

From Table 2, emerges that the number of available ratings increases with the positivity of the overall rating. From the number of reviews per star rating we can assume that the reviewers have a tendency to give high ratings. We will assume that reviews with a rating of 3 stars is a negative review, albeit slightly.

For the conducted sentiment analysis, two sets of reviews were examined. To ease the process of document analysis and lexicon construction as well as to provide the same amount of training data for each class label [55], both datasets were selected as balanced. For the ratings scale experiments, the dataset was constructed consisting of 2.200 reviews randomly taken from each of the 5 classes. The number of reviews chosen is defined from ratings class 1 star, leaving room for randomness. The dataset was further split in training data of 2000 reviews per class and validation data of 200 reviews per class. The other dataset, used for binary sentiment classification, consisted of 5500 reviews from the positive class of ratings 4 through 5 and the same amount of reviews for the negative class of ratings 1 through 3. Number of Reviews per user Rating

5.3 Experimental Setup

From the reviews that were gathered, the review title and review text were processed separately. In Figure 2 the reader can see the steps of NLP that were used on raw text before features could be extracted. Initially, both review titles and review text was preprocessed. The preprocessing steps consist of a tokenization step, where the text is split into single word tokens. Punctuation marks were split into separate tokens. The next step removed all known contractions with the words that were contracted. Any tokens that had more than 3 same consecutive characters were shortened back to their normal form. The same procedure was followed for consecutive punctuation. Special tokens denoting lengthening and intensifying were added. All tokens were passed through a spell check and if appropriate suggestions were available, spelling mistakes were corrected. Tokens, that follow a negation, were appended with a negation mark and stopwords were removed.

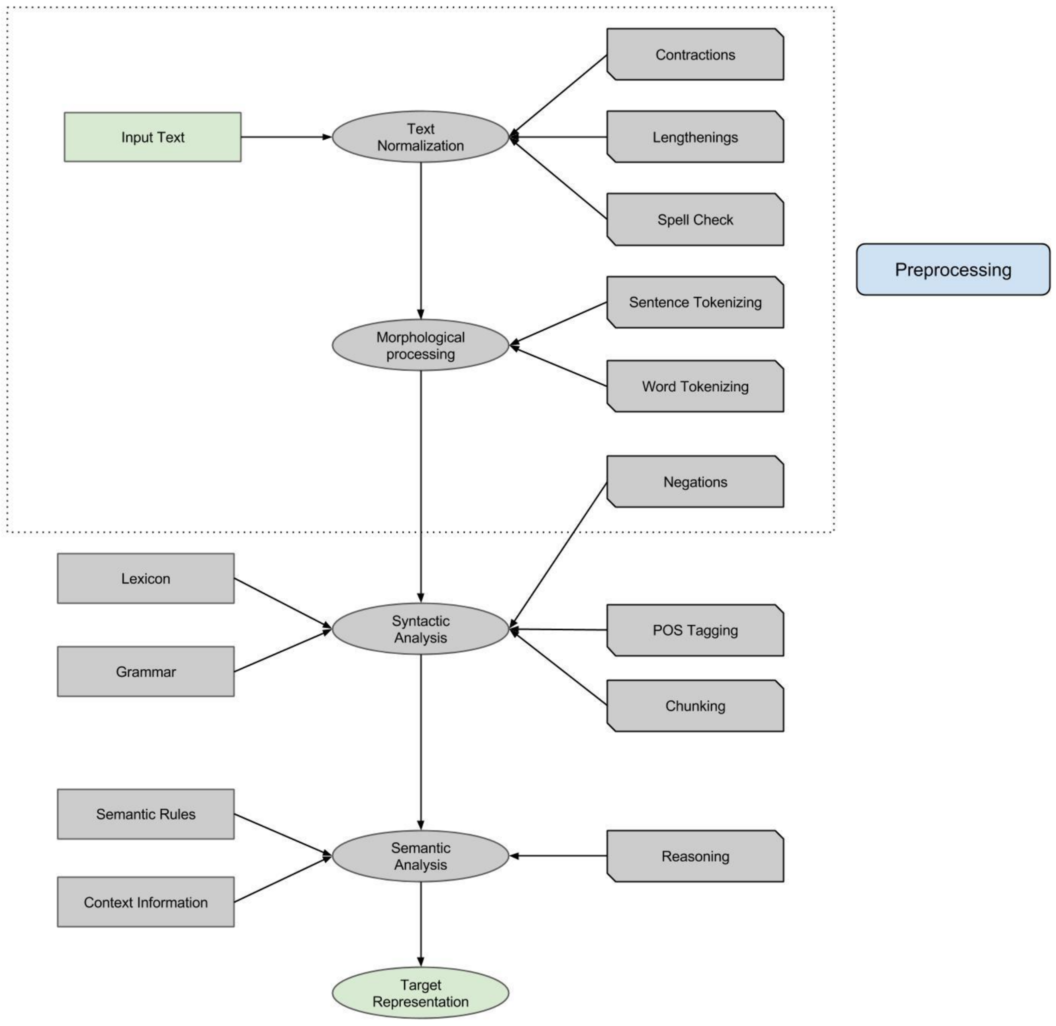


Figure 2 The steps of natural language processing that were used in the text preprocessing stage

The resulting token lists were given to the feature extraction methods. The TFIDF feature extraction method was used to create document vector representations for unigrams, unigrams and bigrams and unigrams, bigrams and trigrams. All possible combinations of weighting and normalization were tested. More specifically, vectors with token counts and frequencies were created. The vectors were normalized by the inverse document frequencies of the words, l1 normalization which normalizes each sample to have a sum of 1 in all features and l2 normalization which introduces a logarithmic normalization. Also, further removal of stopwords was applied by removing words that appear in only a single document to prevent the model from over fitting in the training set and words with a high document frequency as they tend to convey little to none discriminative power. We trained and tested classifiers using only the title vectors, only the review text vectors and by using a concatenation of both title and text vectors for both ratings and binary classification.

The second model that was used to extract features was the Doc2Vec. This model uses the word vector representations created by Word2Vec and aggregates the document representation in the same amount of dimensions. A grid search experiment was conducted in order to find to correct number of hidden dimensions. We also tried to see if the document should be represented by count, frequency or tfidf vectors. For this model also, we extracted measures separately for review title and text and for classification in the 5 ratings scale and the binary classification scale.

Using the SentiWordNet model, we extracted a third set of features. Initially the text was split to sentences. For each word in the sentence we add the positive, neutral and negative sentiment extracted from the SentiWordNet synset lexicon. We also count the tokens that were not found in the dictionary. For each word in the dictionary we added an extra feature with the difference of positive from negative sentiment. If the sum of this value is positive, then we can deduce that the sentence has a positive sentiment, otherwise it has a negative sentiment. For each review we calculate the mean sentiment from the sentences. A binary classification could be completed just by checking the value of the last feature. Reviews with a positive overall sentiment could be classified in the positive class, while reviews with a negative value would be classified in the negative class. We decided to further investigate the model by feeding the vectors to classifiers. This allowed for a ratings scale classification of the reviews by allowing the classifier to find the best splitting point.

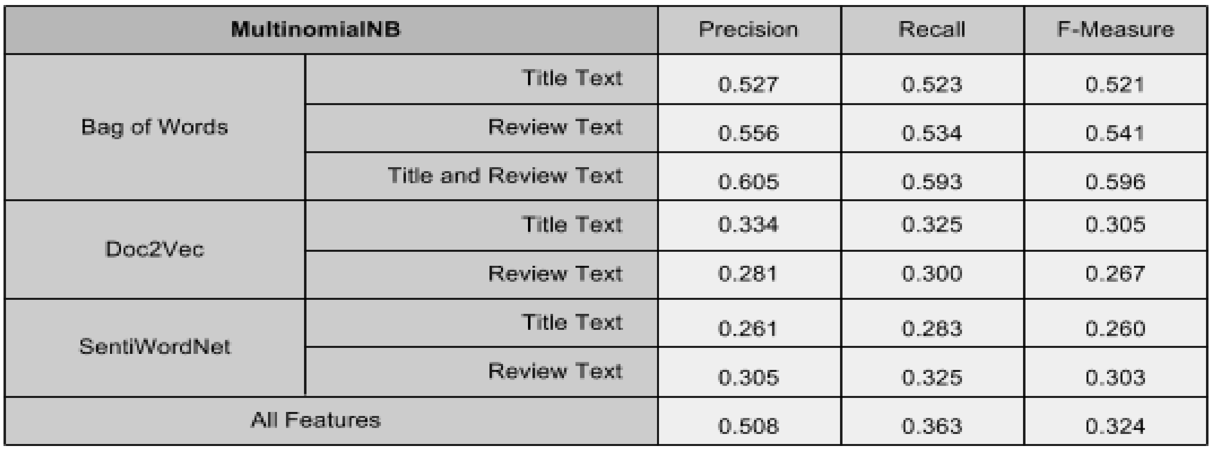
The classifiers that were suggested in the literature as having the best results in the text classification task, are Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM) with a Linear kernel and with an RBF kernel.

Finally, we combined all the features for all three models and measured the classification process. For this experiment, due to time and memory complexity problems, we could not perform a grid search experiment to detect the best parameters. Instead we selected the best parameters from the previous experiments

5.4 Results

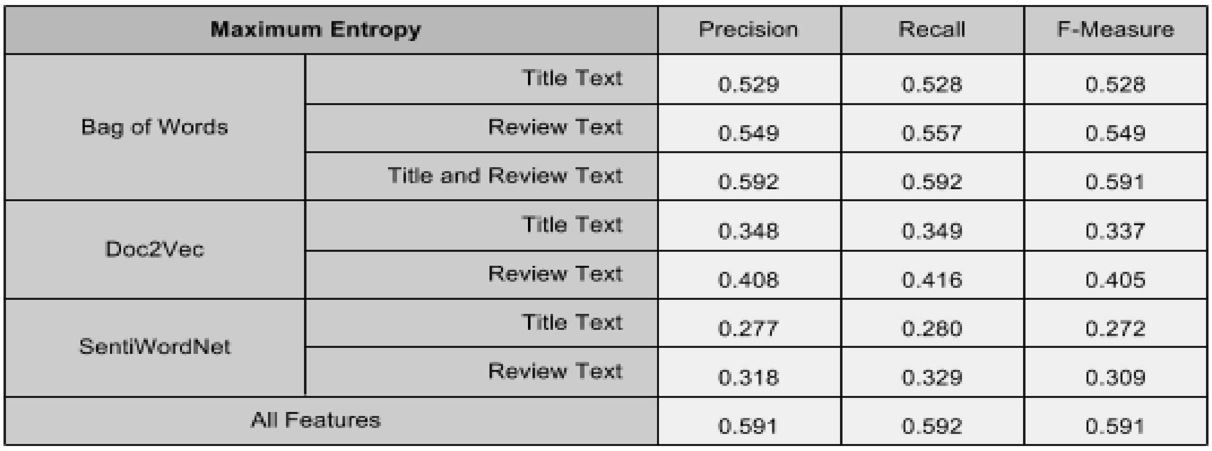
In total, 8 different feature sets were used to evaluate each one of the 4 classifiers that was selected. Each classifier was investigated both on a 1-5 scale and on a binary scale.

On the 1-5 scale classification task, all classifiers had comparable results with an f-measure of close to 0.6 which is almost as good as literature. The best results were achieved by the BOW model of feature extraction with the two other models way behind. While the best classification was achieved by SVM with an RBF kernel, as we can see in Table 6, Maximum Entropy managed to have good results with any set of features that was given to it [Table 4].



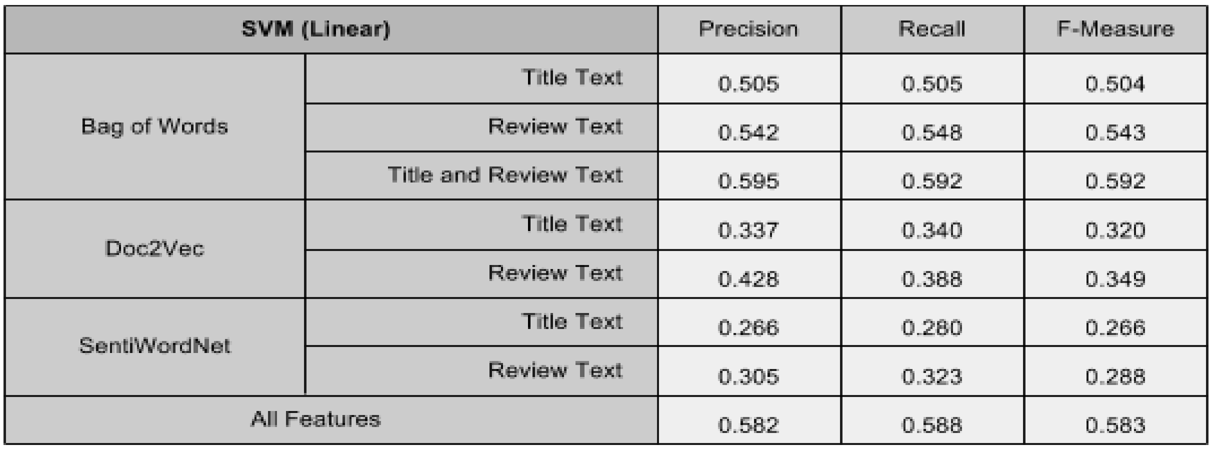
MultinomialNB		Precision	Recall	F-Measure
Bag of Words	Title Text	0.527	0.523	0.521
	Review Text	0.556	0.534	0.541
	Title and Review Text	0.605	0.593	0.596
Doc2Vec	Title Text	0.334	0.325	0.305
	Review Text	0.281	0.300	0.267
SentiWordNet	Title Text	0.261	0.283	0.260
	Review Text	0.305	0.325	0.303
All Features		0.508	0.363	0.324

Table 3 Precision, Recall and F-Measure for MultinomialNB classifier on all feature sets on ratings scale classification



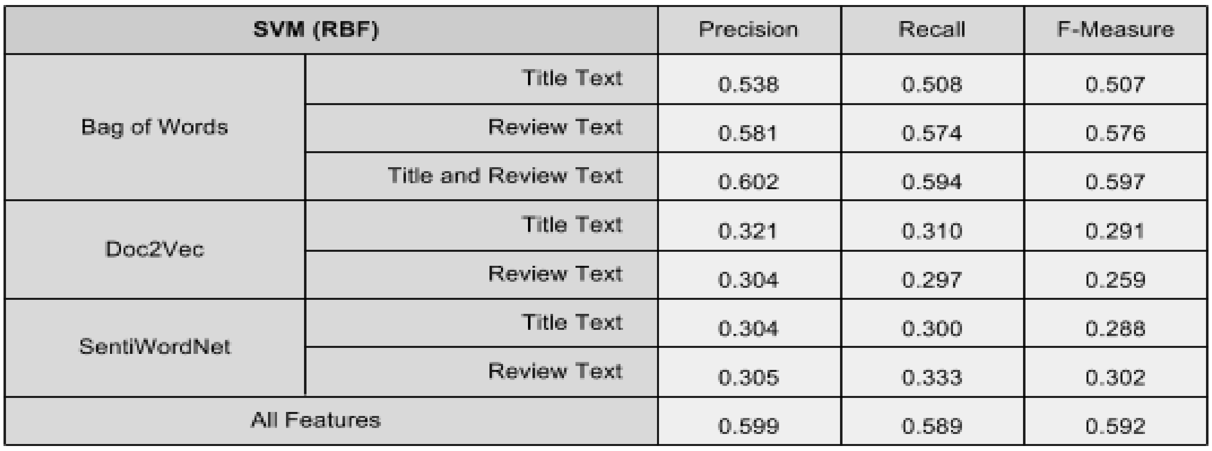
Maximum Entropy		Precision	Recall	F-Measure
Bag of Words	Title Text	0.529	0.528	0.528
	Review Text	0.549	0.557	0.549
	Title and Review Text	0.592	0.592	0.591
Doc2Vec	Title Text	0.348	0.349	0.337
	Review Text	0.408	0.416	0.405
SentiWordNet	Title Text	0.277	0.280	0.272
	Review Text	0.318	0.329	0.309
All Features		0.591	0.592	0.591

Table 4 Precision, Recall and F-Measure for Maximum Entropy classifier on all feature sets on ratings scale classification



SVM (Linear)		Precision	Recall	F-Measure
Bag of Words	Title Text	0.505	0.505	0.504
	Review Text	0.542	0.548	0.543
	Title and Review Text	0.595	0.592	0.592
Doc2Vec	Title Text	0.337	0.340	0.320
	Review Text	0.428	0.388	0.349
SentiWordNet	Title Text	0.266	0.280	0.266
	Review Text	0.305	0.323	0.288
All Features		0.582	0.588	0.583

Table 5 Precision, Recall and F-Measure for SVM classifier with linear kernel on all feature sets on ratings scale classification

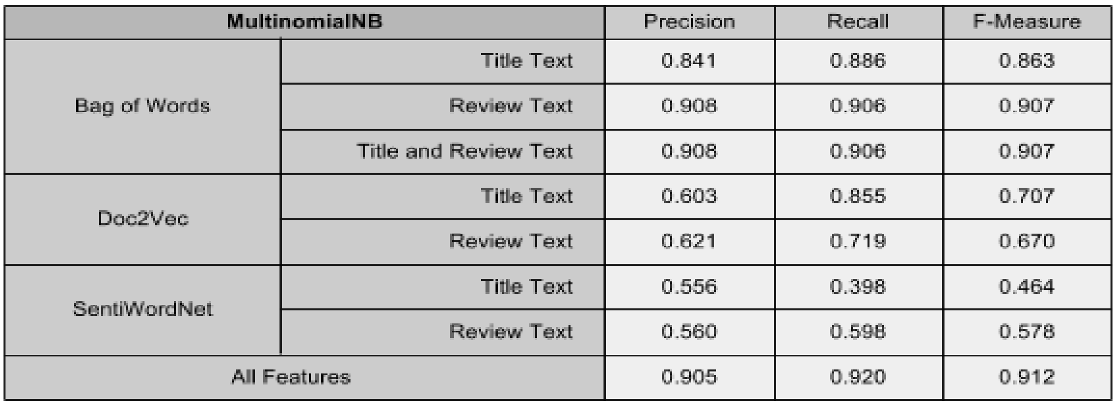


SVM (RBF)		Precision	Recall	F-Measure
Bag of Words	Title Text	0.538	0.508	0.507
	Review Text	0.581	0.574	0.576
	Title and Review Text	0.602	0.594	0.597
Doc2Vec	Title Text	0.321	0.310	0.291
	Review Text	0.304	0.297	0.259
SentiWordNet	Title Text	0.304	0.300	0.288
	Review Text	0.305	0.333	0.302
All Features		0.599	0.589	0.592

Table 6 Precision, Recall and F-Measure for SVM classifier with an RBF kernel on all feature sets on ratings scale classification

In Table 3 we can see that multinomialNB had the worst result when trained with the concatenation of all the features.

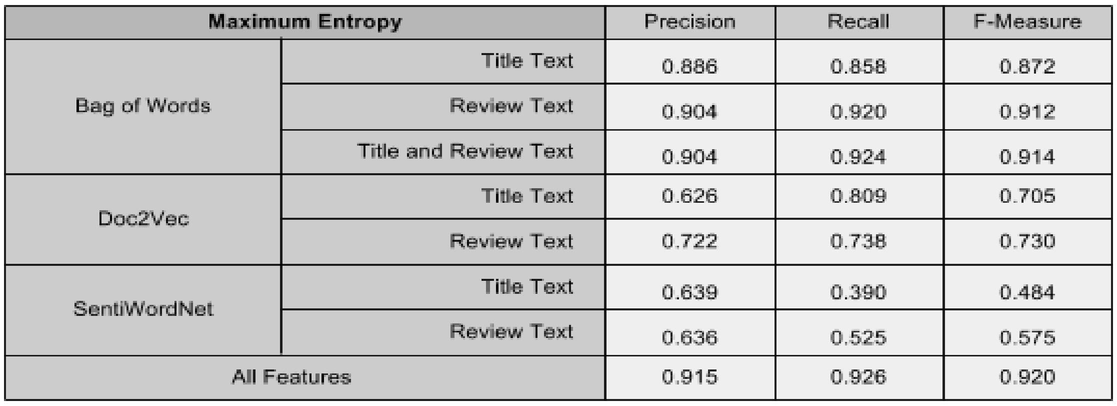
On Table 7 are presented the precision, recall and F-measure for the Multinomial Naïve Bayes classifier per feature set, using the best found parameters on the binary classification task. We can see that NB performed better using the BOW features than either the Doc2Vec or the SentiWordNet features. While using the features from the review text in conjunction with the features from the title, didn't benefit in comparison to using only the features of the text, using the title features alone couldn't come up to par for the BOW model. The SentiWordNet model managed a better performance with the review text features, than the title features. This model, using the title features had a very low recall score (0.398) which would suggest that is very low variance in the feature values. On the other hand, NB was the only classifier where using the Doc2Vec model on title features had a better F-measure than on review text features.



MultinomialNB		Precision	Recall	F-Measure
Bag of Words	Title Text	0.841	0.886	0.863
	Review Text	0.908	0.906	0.907
	Title and Review Text	0.908	0.906	0.907
Doc2Vec	Title Text	0.603	0.855	0.707
	Review Text	0.621	0.719	0.670
SentiWordNet	Title Text	0.556	0.398	0.464
	Review Text	0.560	0.598	0.578
All Features		0.905	0.920	0.912

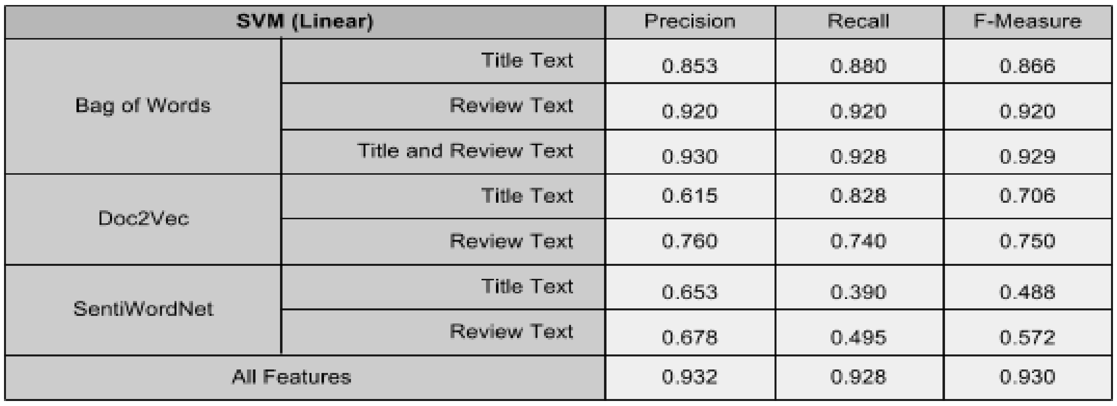
Table 7 Precision, Recall and F-Measure for MultinomialNB classifier on all feature sets on binary classification

By comparing the results in Table 7 through Table 10, all classifiers achieved their best when all features were concatenated. The best features were contributed by the BOW model, while the worst were contributed by SentiWordNet.



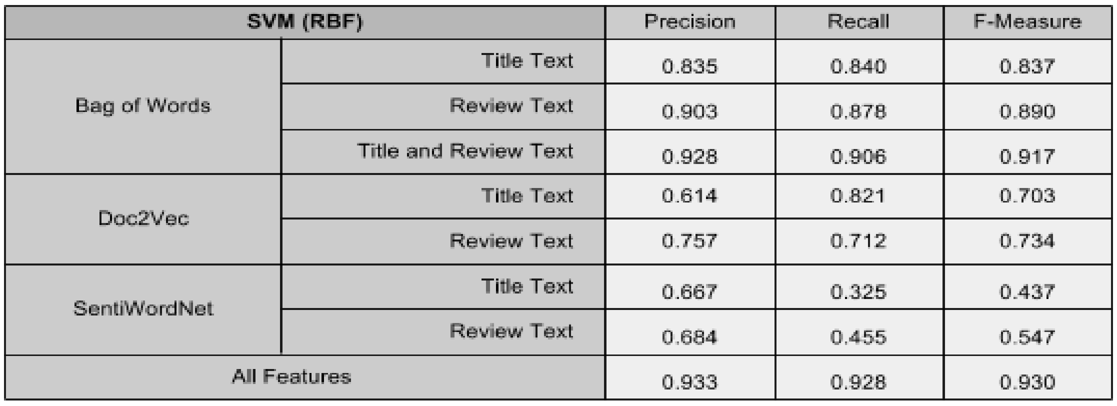
Maximum Entropy		Precision	Recall	F-Measure
Bag of Words	Title Text	0.886	0.858	0.872
	Review Text	0.904	0.920	0.912
	Title and Review Text	0.904	0.924	0.914
Doc2Vec	Title Text	0.626	0.809	0.705
	Review Text	0.722	0.738	0.730
SentiWordNet	Title Text	0.639	0.390	0.484
	Review Text	0.636	0.525	0.575
All Features		0.915	0.926	0.920

Table 8 Precision, Recall and F-Measure for Maximum Entropy classifier on all feature sets on binary classification



SVM (Linear)		Precision	Recall	F-Measure
Bag of Words	Title Text	0.853	0.880	0.866
	Review Text	0.920	0.920	0.920
	Title and Review Text	0.930	0.928	0.929
Doc2Vec	Title Text	0.615	0.828	0.706
	Review Text	0.760	0.740	0.750
SentiWordNet	Title Text	0.653	0.390	0.488
	Review Text	0.678	0.495	0.572
All Features		0.932	0.928	0.930

Table 9 Precision, Recall and F-Measure for SVM classifier with linear kernel on all feature sets on binary classification



SVM (RBF)		Precision	Recall	F-Measure
Bag of Words	Title Text	0.835	0.840	0.837
	Review Text	0.903	0.878	0.890
	Title and Review Text	0.928	0.906	0.917
Doc2Vec	Title Text	0.614	0.821	0.703
	Review Text	0.757	0.712	0.734
SentiWordNet	Title Text	0.667	0.325	0.437
	Review Text	0.684	0.455	0.547
All Features		0.933	0.928	0.930

Table 10 Precision, Recall and F-Measure for SVM classifier with an RBF kernel on all feature sets on binary classification

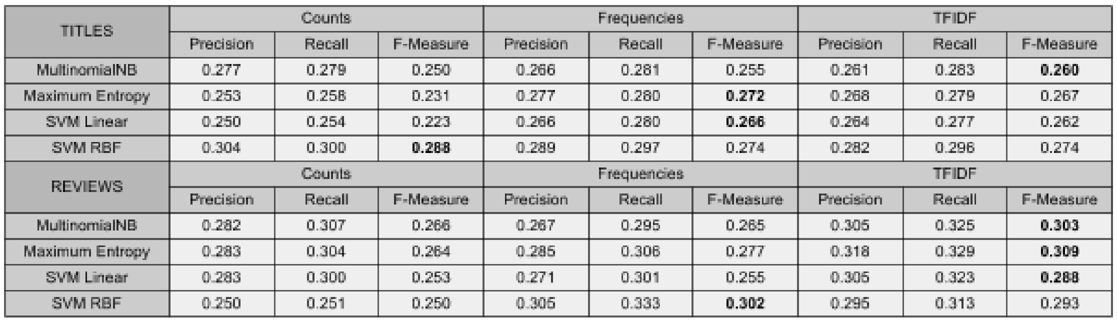
The best overall performance was achieved by Support Vector Machines using a linear kernel.

5.4.1 SentiWordNet Model

The model we implemented using SentiWordNet, calculates a positive, objective and negative score for each token encountered in the corpus, by averaging the values of all synsets in

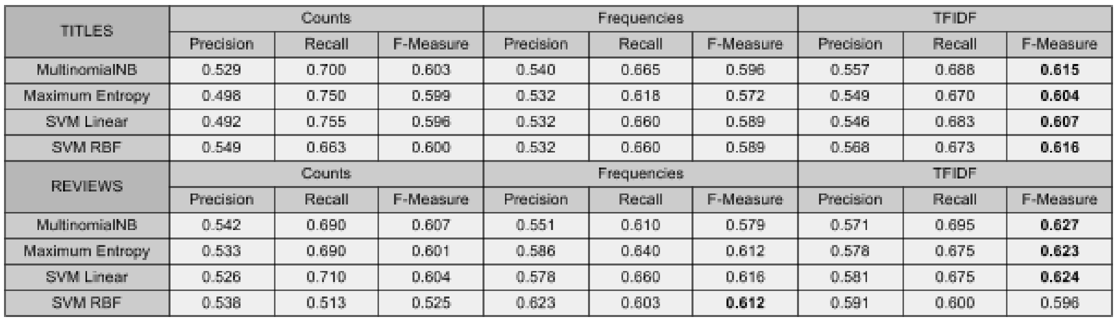
SentiWordNet that have the same text representation. Tokens that are used after a negation have their polarity reversed. In order to anticipate unknown tokens, a fourth value was added, denoting unknown sentiment, and a fifth where we calculated the difference between positive and negative sentiment. Since a token would not pass it's sentiment in another sentence, we first split each review in sentences. For each sentence we form a vector as in a BOW model. We aggregated the sentence sentiment by calculating the dot product of the sentiment matrix with the sentence vector. In order to aggregate at the review level, we calculated the mean of each feature for the sentences used. This process adds five features for every review, a positive sentiment value, an objective value, a negative sentiment value, an unknown word binary value and the difference between positive and sentiment value.

In Table 11 we present the precision, recall and f-measure achieved by all four of the classifiers selected. Each classifier was trained and tested using a count vector, a term frequency vector and a TFIDF vector. Although the sentiment in every sentence should not be influenced by the frequency each token appears in the corpus, it is interesting to note that all classifiers had very good results with the TFIDF vectors, especially on the review text. When the results were converted from a scale of 1-5 into a positive-negative scale, all classifiers in all feature sets except SVM-RBF yield the best result for TFIDF vectors as we can see in Table 12.



TITLES	Counts			Frequencies			TFIDF		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
MultinomialNB	0.277	0.279	0.250	0.266	0.281	0.255	0.261	0.283	**0.260**
Maximum Entropy	0.253	0.258	0.231	0.277	0.280	**0.272**	0.268	0.279	0.267
SVM Linear	0.250	0.254	0.223	0.266	0.280	**0.266**	0.264	0.277	0.262
SVM RBF	0.304	0.300	**0.288**	0.289	0.297	0.274	0.282	0.296	0.274
REVIEWS	Counts			Frequencies			TFIDF		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
MultinomialNB	0.282	0.307	0.266	0.267	0.295	0.265	0.305	0.325	**0.303**
Maximum Entropy	0.283	0.304	0.264	0.285	0.306	0.277	0.318	0.329	**0.309**
SVM Linear	0.283	0.300	0.253	0.271	0.301	0.255	0.305	0.323	**0.288**
SVM RBF	0.250	0.251	0.250	0.305	0.333	**0.302**	0.295	0.313	0.293

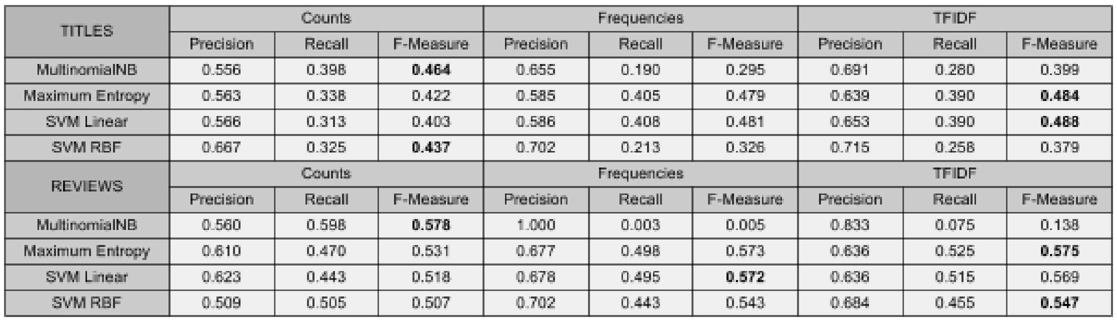
Table 11 Precision, Recall and F-Measure by all classifiers on a ratings scale



TITLES	Counts			Frequencies			TFIDF		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
MultinomialNB	0.529	0.700	0.603	0.540	0.665	0.596	0.557	0.688	**0.615**
Maximum Entropy	0.498	0.750	0.599	0.532	0.618	0.572	0.549	0.670	**0.604**
SVM Linear	0.492	0.755	0.596	0.532	0.660	0.589	0.546	0.683	**0.607**
SVM RBF	0.549	0.663	0.600	0.532	0.660	0.589	0.568	0.673	**0.616**
REVIEWS	Counts			Frequencies			TFIDF		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
MultinomialNB	0.542	0.690	0.607	0.551	0.610	0.579	0.571	0.695	**0.627**
Maximum Entropy	0.533	0.690	0.601	0.586	0.640	0.612	0.578	0.675	**0.623**
SVM Linear	0.526	0.710	0.604	0.578	0.660	0.616	0.581	0.675	**0.624**
SVM RBF	0.538	0.513	0.525	0.623	0.603	**0.612**	0.591	0.600	0.596

Table 12 Precision, Recall and F-Measure by all classifiers on a ratings scale converted to binary scale

The same experiments were conducted with the same setup but the review target classification was converted to a positive-negative scale. As we can see in Table 13, the results achieved by this method are very low.



TITLES	Counts			Frequencies			TFIDF		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
MultinomialNB	0.556	0.398	**0.464**	0.655	0.190	0.295	0.691	0.280	0.399
Maximum Entropy	0.563	0.338	0.422	0.585	0.405	0.479	0.639	0.390	**0.484**
SVM Linear	0.566	0.313	0.403	0.586	0.408	0.481	0.653	0.390	**0.488**
SVM RBF	0.667	0.325	**0.437**	0.702	0.213	0.326	0.715	0.258	0.379
REVIEWS	Counts			Frequencies			TFIDF		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
MultinomialNB	0.560	0.598	**0.578**	1.000	0.003	0.005	0.833	0.075	0.138
Maximum Entropy	0.610	0.470	0.531	0.677	0.498	0.573	0.636	0.525	**0.575**
SVM Linear	0.623	0.443	0.518	0.678	0.495	**0.572**	0.636	0.515	0.569
SVM RBF	0.509	0.505	0.507	0.702	0.443	0.543	0.684	0.455	**0.547**

Table 13 Precision, Recall and F-Measure by all classifiers on a binary scale classification

As already explained, the model implemented is very simple and that would explain the very low results compared to the other models. The main reason for its implementation is to test if these five features can help one of the other models to improve.

5.4.2 Word2Vec Model

It is our understanding that the dimensions of the vectors representing the dictionary words, can be perceived as hidden aspects of each word's sentiment. In our model we create a dictionary for each word used in the corpus. In order to train the CBOW model, a sequence of sentences is expected, so the reviews were split into sentences before they were fed to the word2vec model. For each token in the review corpus, a vector was calculated from word2vec creating a matrix with dimensions number of unique tokens by number of aspects. We then normalize each aspect in the range of 0 to 1. For each review we create the term frequency vector which of course has a dimension of number of unique tokens. By calculating the dot product of the vector with the matrix, we have a vector representing the percentage of each hidden aspect in the review.

All classifiers were investigated using a grid search to detect the optimal number of hidden aspects. The optimal number of hidden aspects is the number of aspects that achieve the best f-measure score in classifying the aggregated document vectors. In Table 14, the optimal number of hidden aspects for every classifier are presented, together with the f-measure score that they achieved in a 5 class scale, while in Table 15 the optimal number of hidden aspects are presented for a binary scale classification.

TITLES	Without Lemmatization		With Lemmatization	
	#Attributes	F-Measure	#Attributes	F-Measure
MultinomialNB	65	**0.305**	15	0.297
Maximum Entropy	55	**0.337**	75	0.327
SVM Linear	40	**0.329**	130	0.326
SVM RBF	65	**0.337**	145	0.336
REVIEWS	Without Lemmatization		With Lemmatization	
	#Attributes	F-Measure	#Attributes	F-Measure
MultinomialNB	70	0.266	90	**0.267**
Maximum Entropy	70	0.404	50	**0.405**
SVM Linear	90	**0.401**	55	0.394
SVM RBF	100	0.424	135	**0.427**

Table 14 Best F-Measure for each classifier and number of hidden attributes for each classifier on a five class scale

TITLES	Without Lemmatization		With Lemmatization	
	#Attributes	F-Measure	#Attributes	F-Measure
Naive Bayes	8	0.707	11	0.704
Max Entropy	140	0.705	5	0.702
SVM Linear	43	0.702	45	0.706
SVM RBF	61	0.703	3	0.703
REVIEWS	Without Lemmatization		With Lemmatization	
	#Attributes	F-Measure	#Attributes	F-Measure
Naive Bayes	24	0.670	23	0.668
Max Entropy	84	0.722	61	0.730
SVM Linear	94	0.741	132	0.750
SVM RBF	94	0.734	150	0.753

Table 15 Best F-Measure for each classifier and number of hidden attributes for each classifier on a binary scale

As we can see, on title features, all classifiers achieved better classification using non-lemmatized words, while lemmatization seem the way to go when the whole review is given to the feature extractor when predicting 5 classes. When predicting only positive and negative reviews, lemmatization seems to only affect the number of hidden aspects that achieve the best result. In general, this model of extracting features was inferior to the BOW model, which would suggest that a lot of sentiment information is lost either while extracting the word vectors, either when aggregating the document vectors.

MultinomialNB performed the worst from all classifiers, especially on the 5 classes task, while the others were more or less on the same level. That would suggest that perhaps another variation of Naïve Bayes might be better suited for this set of tasks.

In Table 16 through 18, we have plotted the precision, recall and f-measure achieved for every classifier over the number of hidden aspects used for vectors, for vectors using the review text and title and for the 5 class scale and binary classification tasks.

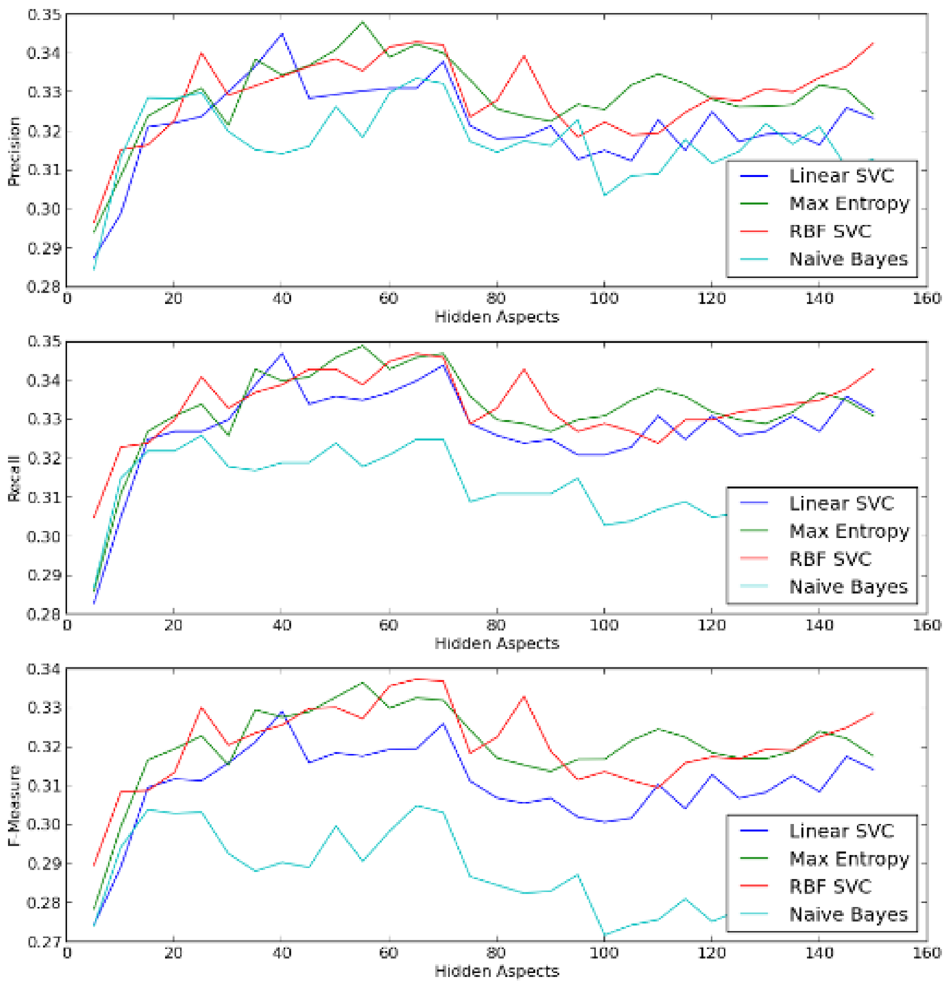


Table 16 Precision, Recall and F-Measure over number of hidden aspects when trained from review titles on a five ratings scale

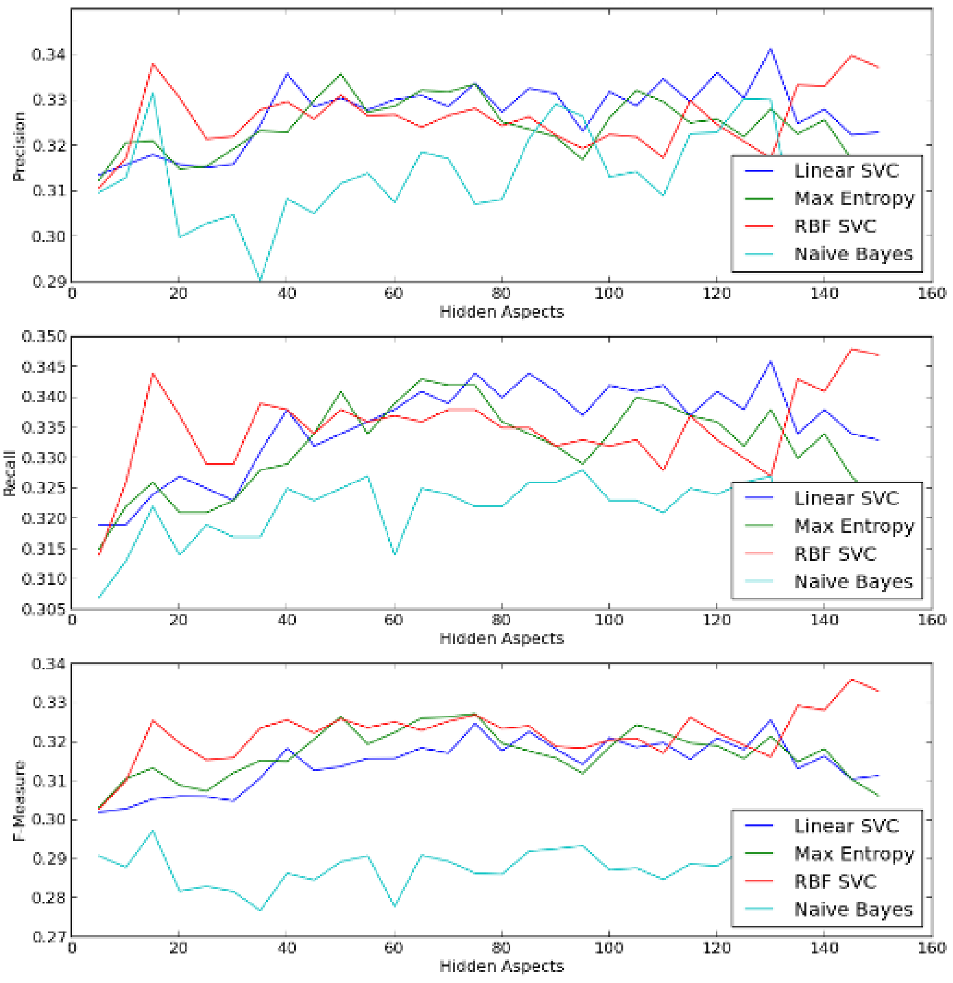


Table 17 Precision, Recall and F-Measure over number of hidden aspects when trained from lemmatized review titles on a five ratings scale

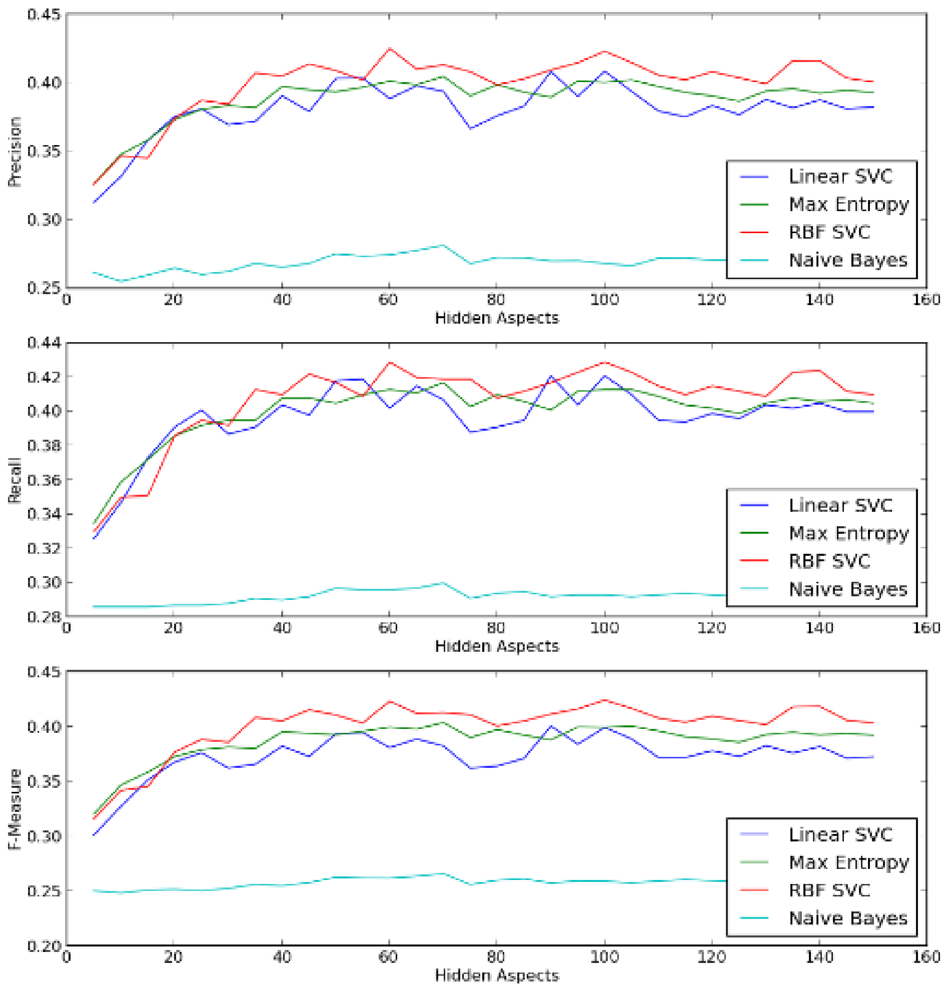


Table 18 Precision, Recall and F-Measure over number of hidden aspects when trained from review text on a five ratings scale

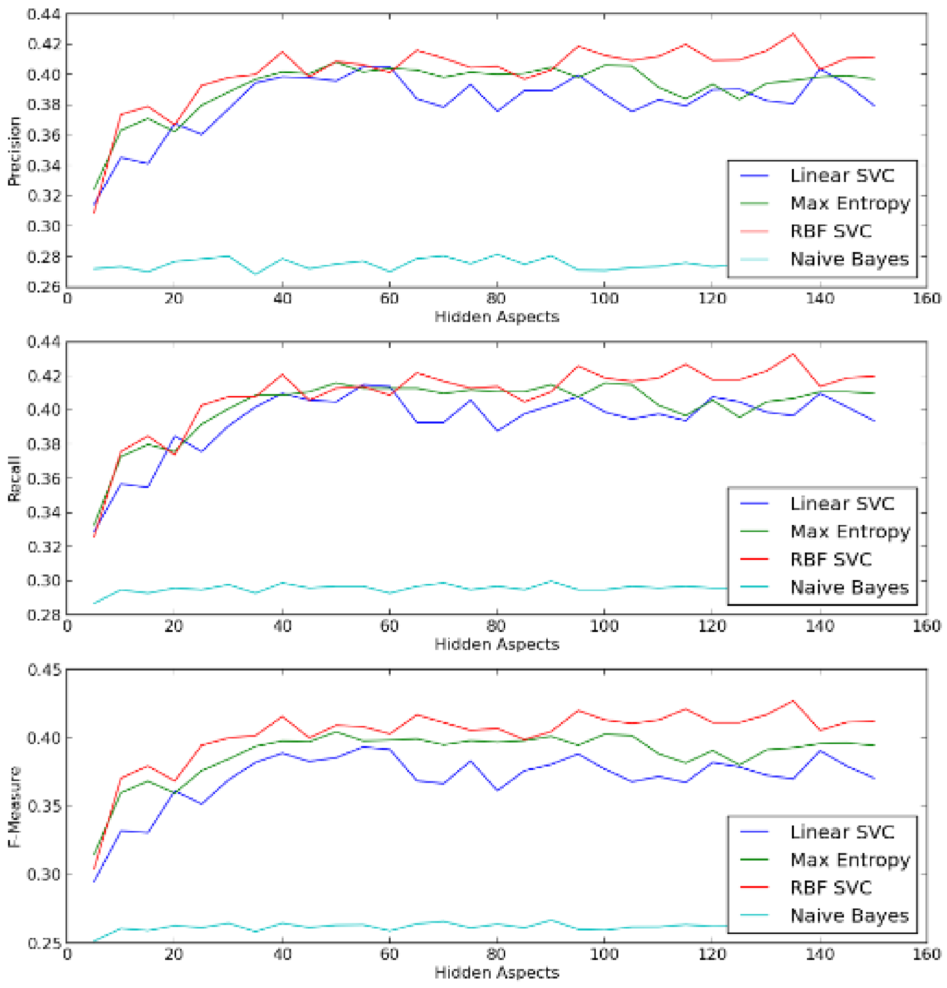


Table 19 Precision, Recall and F-Measure over number of hidden aspects when trained from lemmatized review text on a five ratings scale

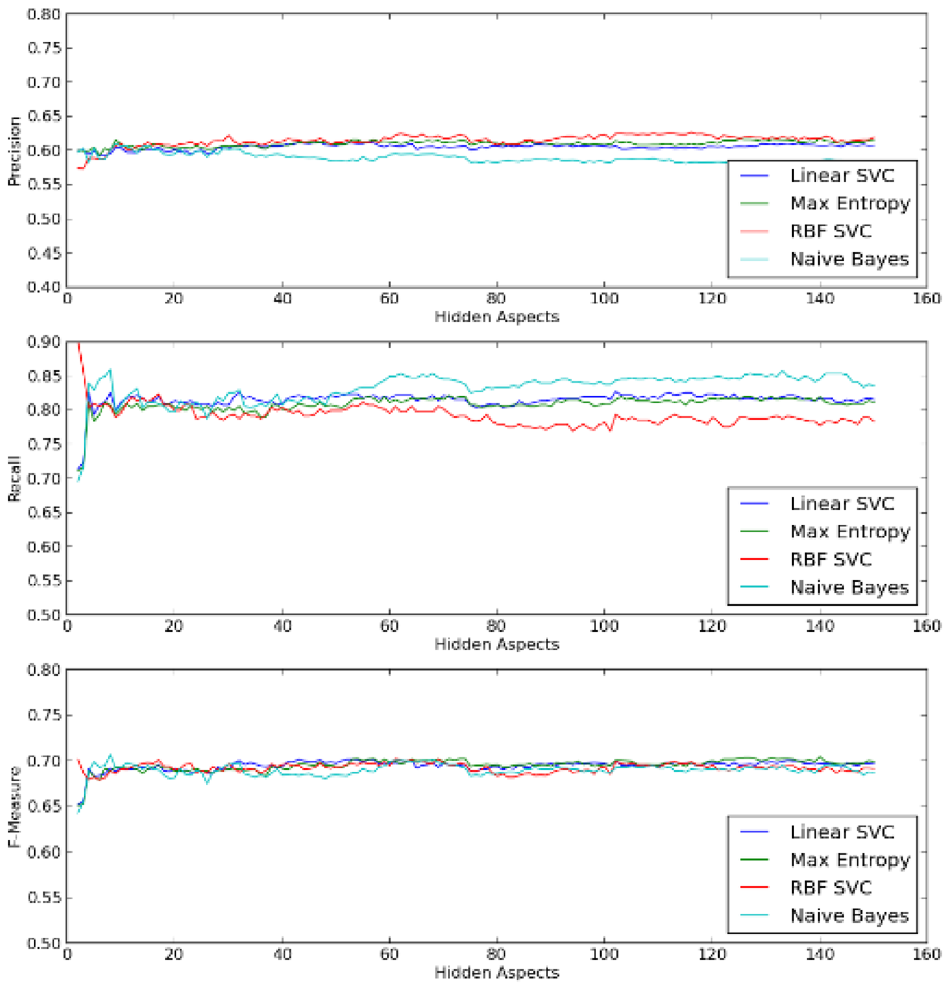


Table 20 Precision, Recall and F-Measure over number of hidden aspects when trained from review titles on a binary scale

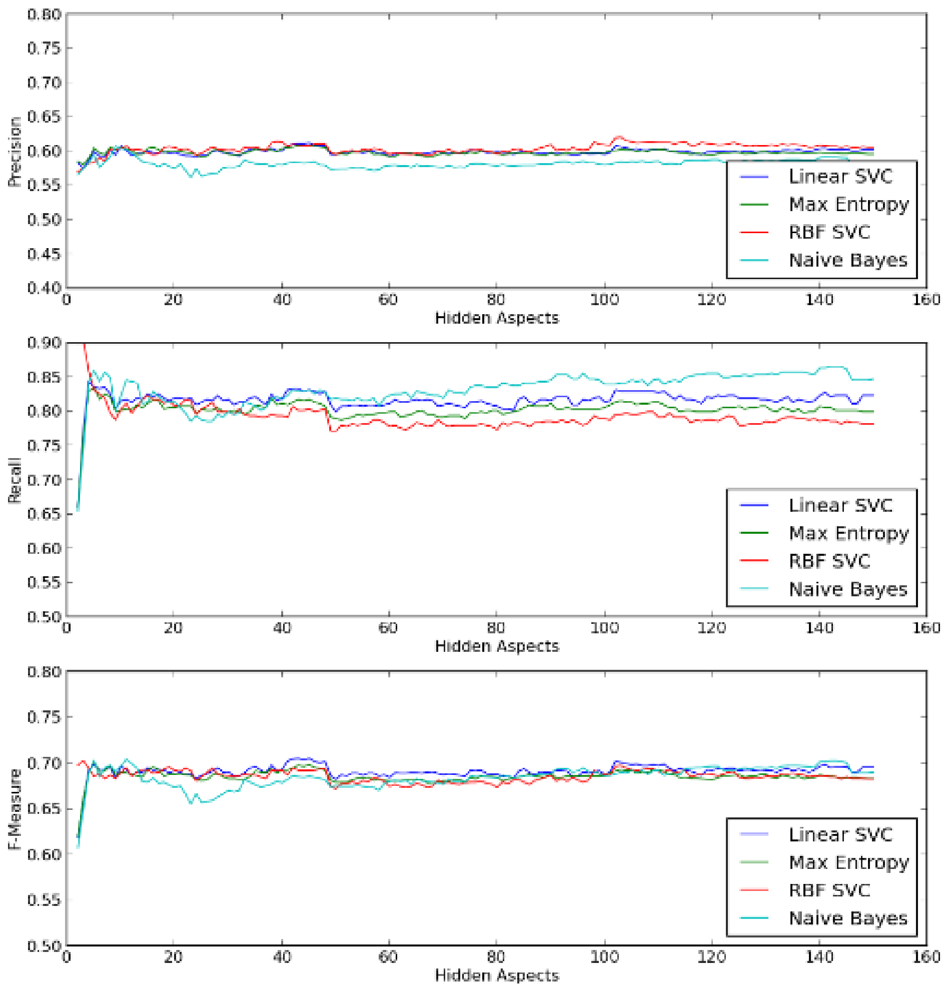


Table 21 Precision, Recall and F-Measure over number of hidden aspects when trained from lemmatized review titles on a binary scale

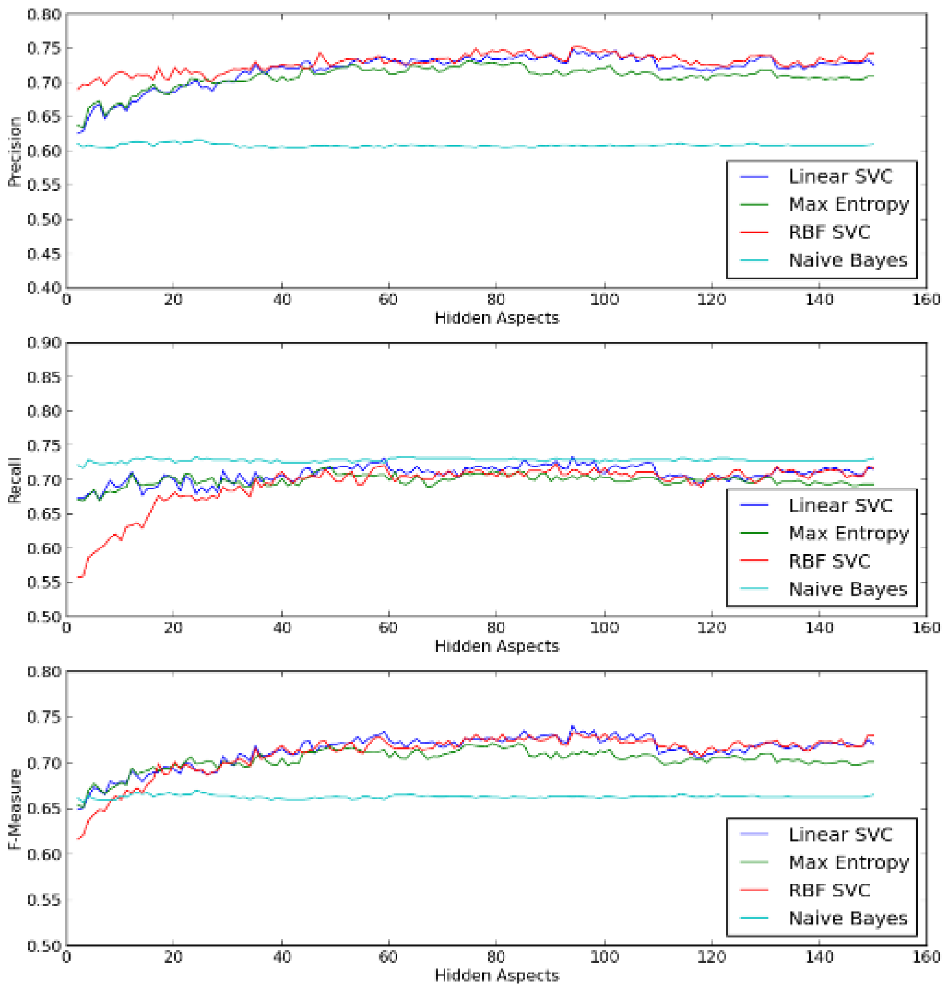


Table 22 Precision, Recall and F-Measure over number of hidden aspects when trained from review text on a binary scale

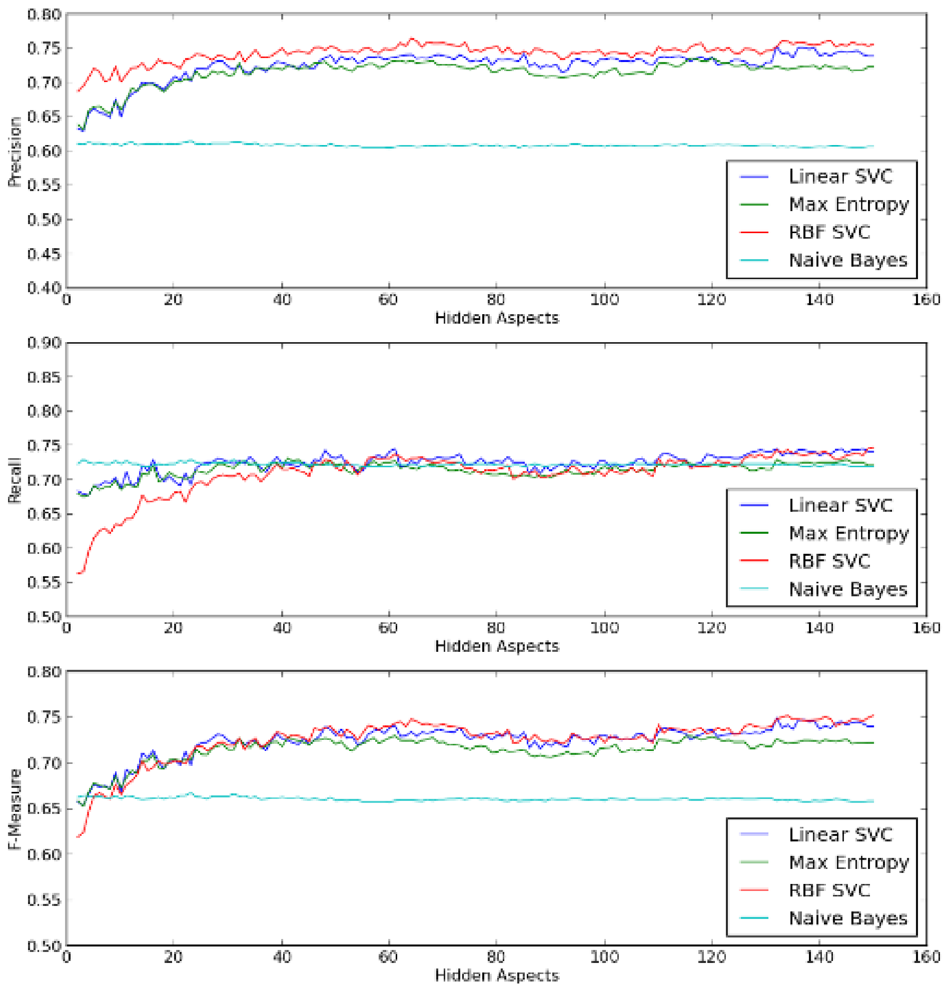


Table 23 Precision, Recall and F-Measure over number of hidden aspects when trained from lemmatized review text on a binary scale

5.4.3 Bag of Words Model

The BOW model achieved in general the best classification results when used on its own. We performed an extensive grid search over all feature extraction parameters for each of the classifiers. All of the classifiers performed their best on both the 5 class and the binary classification tasks when both the title and review text features were extracted. On Table 24 we present the precision, recall and f-measure for the parameters with the best f-measure for each of the tasks over each of the feature sets used with MultinomialNB.

The equivalent information is presented in Table 25 for Maximum Entropy, in Table 26 for SVM with a linear kernel and in Table 27 for SVM with an RBF kernel.

MultinomialNB	PRECISION	RECALL	F-MEASURE
Title Features, ratings classification	0.527	0.523	0.521
Title Features, ratings classification as binary	0.754	0.843	0.796
Title Features, binary classification	0.841	0.886	0.863
Review Features, ratings classification	0.556	0.534	0.541
Review Features, ratings classification as binary	0.887	0.785	0.833
Review Features, binary classification	0.908	0.906	0.907
Title AND Review Features, ratings classification	0.605	0.593	0.596
Title AND Review Features, ratings classification as binary	0.858	0.860	0.859
Title AND Review Features, binary classification	0.908	0.906	0.907

Table 24 Best Precision, Recall and F-Measure per feature set and classification scale for MultinomialNB

Maximum Entropy	PRECISION	RECALL	F-MEASURE
Title Features, ratings classification	0.529	0.528	0.528
Title Features, ratings classification as binary	0.819	0.813	0.816
Title Features, binary classification	0.886	0.858	0.872
Review Features, ratings classification	0.549	0.557	0.549
Review Features, ratings classification as binary	0.848	0.868	0.858
Review Features, binary classification	0.904	0.920	0.912
Title AND Review Features, ratings classification	0.592	0.592	0.591
Title AND Review Features, ratings classification as binary	0.863	0.868	0.865
Title AND Review Features, binary classification	0.904	0.924	0.914

Table 25 Best Precision, Recall and F-Measure per feature set and classification scale for Maximum Entropy

SVM(Linear)	PRECISION	RECALL	F-MEASURE
Title Features, ratings classification	0.505	0.505	0.504
Title Features, ratings classification as binary	0.803	0.828	0.815
Title Features, binary classification	0.853	0.880	0.866
Review Features, ratings classification	0.542	0.548	0.543
Review Features, ratings classification as binary	0.831	0.870	0.850
Review Features, binary classification	0.920	0.920	0.920
Title AND Review Features, ratings classification	0.595	0.592	0.592
Title AND Review Features, ratings classification as binary	0.867	0.882	0.875
Title AND Review Features, binary classification	0.930	0.928	0.929

Table 26 Best Precision, Recall and F-Measure per feature set and classification scale for SVM with linear kernel

SVM(RBF)	PRECISION	RECALL	F-MEASURE
Title Features, ratings classification	0.538	0.508	0.507
Title Features, ratings classification as binary	0.781	0.813	0.797
Title Features, binary classification	0.835	0.840	0.837
Review Features, ratings classification	0.581	0.574	0.576
Review Features, ratings classification as binary	0.872	0.855	0.864
Review Features, binary classification	0.903	0.878	0.890
Title AND Review Features, ratings classification	0.602	0.594	0.597
Title AND Review Features, ratings classification as binary	0.896	0.865	0.880
Title AND Review Features, binary classification	0.928	0.906	0.917

Table 27 Best Precision, Recall and F-Measure per feature set and classification scale for SVM with an RBF kernel

N-GRAMS	unigrams	(1,1)
	unigrams and bigrams	(1,2)
	unigrams, bigrams and trigrams	(1,3)
NORMALIZATION	no normalization	
	sum of features is 1	l1
	logarithm of feature	l2
USE OF IDF	use of inverse document frequency	TRUE
		FALSE
BINARY	count of terms	TRUE
	appearance of term	FALSE
MAX DF	maximum percentage of document a term can appear in to be included	0.2
		0.5
		0.8
MIN DF	minimum number of documents a term must appear in to be included	1
		2

Table 28 Parameters used for feature extraction in BOW model

For each classifier, the best parameters of feature extraction were different. The parameters that were used for this model to extract features and how they are denoted in the result tables, are presented in Table 28

The 10 best and 10 worst parameter combinations for each classifier on 5 class classification and binary classification for title vectors and review text vectors, are presented in Table 29 through Table 44.

For both the probabilistic classifiers, on title vectors for 5 class classification, best performance is achieved when using unigrams to construct TF*IDF vectors, with a logarithmic normalization. Terms that appear in more than 20% of the documents, don’t have any important impact on the classification process. Also, both classifiers have the worst results when bigrams and trigrams are added to the vectors and there is no normalization.

SVM’s with either kernel, also perform their best using unigrams, on title vectors for 5 class classification. Normalization of the vectors is important for SVM’s too, but other parameters don’t seem to affect the process.

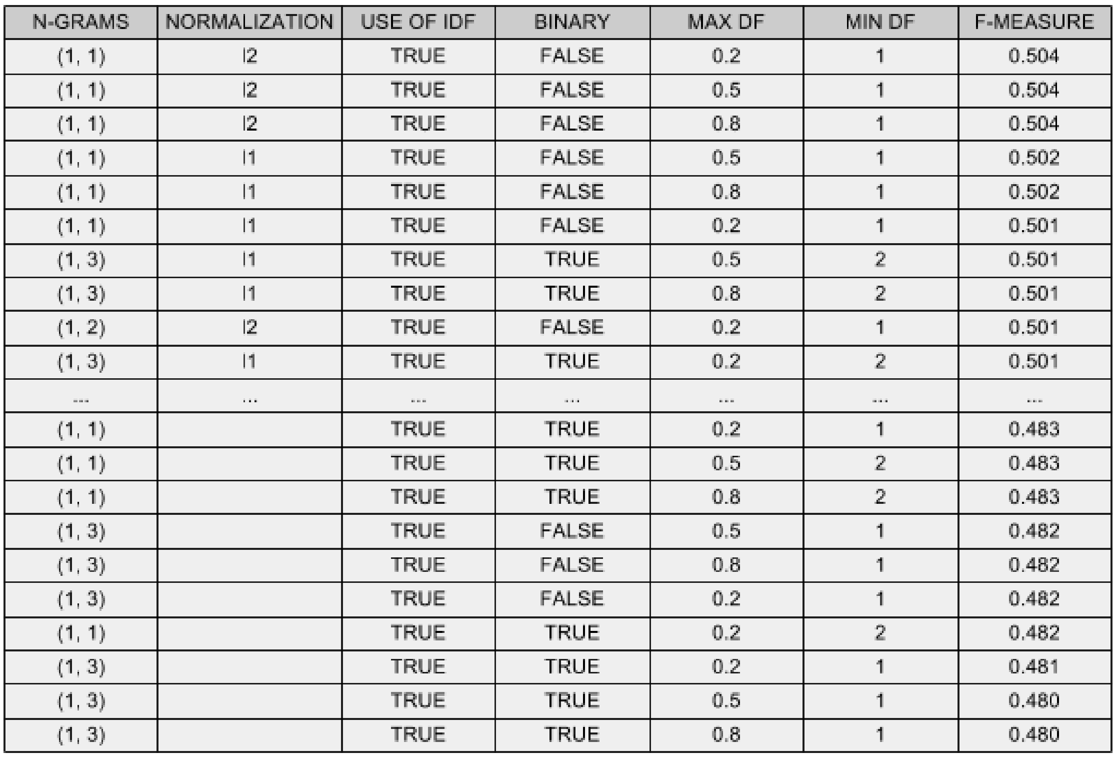
When classifying using the review text, vectors containing the existence information of unigrams and bigrams logarithmically normalized, had the best performance for all classifiers. L1 normalization without using IDF gave the worst results for the two probabilistic classifiers and for SVM with a linear kernel.

SVM with an RBF kernel using unigrams and bigrams TF*IDF vectors had the best f-measure overall when the vectors were logarithmically normalized and the worst f-measure overall when the vectors were not normalized, which would suggest that normalization is the most important factor for good classification results.

The parameters that, were beneficial or detrimental for the classifiers, on the task of binary classification, are different than on the task of 5 ratings classification.

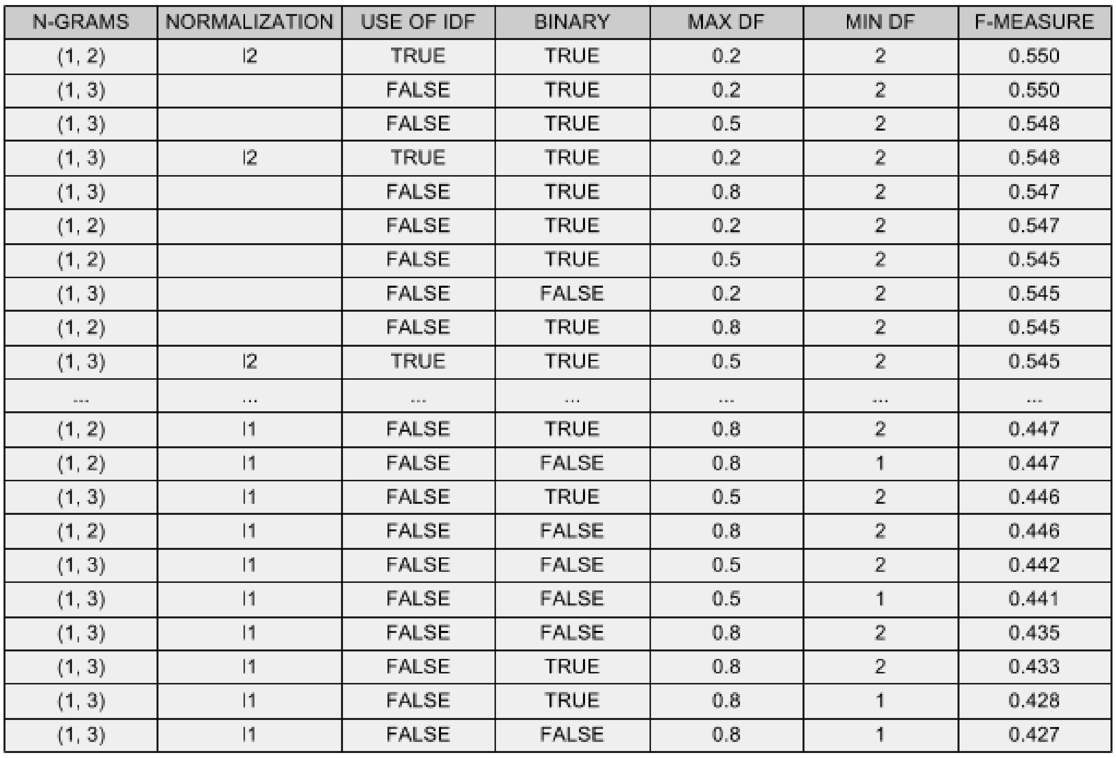
For multinomialNB, using counts of bigrams appearing in titles produces better results and l1 normalization of the vectors is more detrimental than not normalizing them. Vectors of review text, do not benefit as much from logarithmic normalization, while normalizing to 1 has an even more detrimental effect.

For Maximum Entropy, while not normalizing title vectors was detrimental in 5 class classification, for binary classification it is beneficial and instead of the best performance being achieved by logarithmically normalized review text vectors, not normalizing them performs better on binary classification.



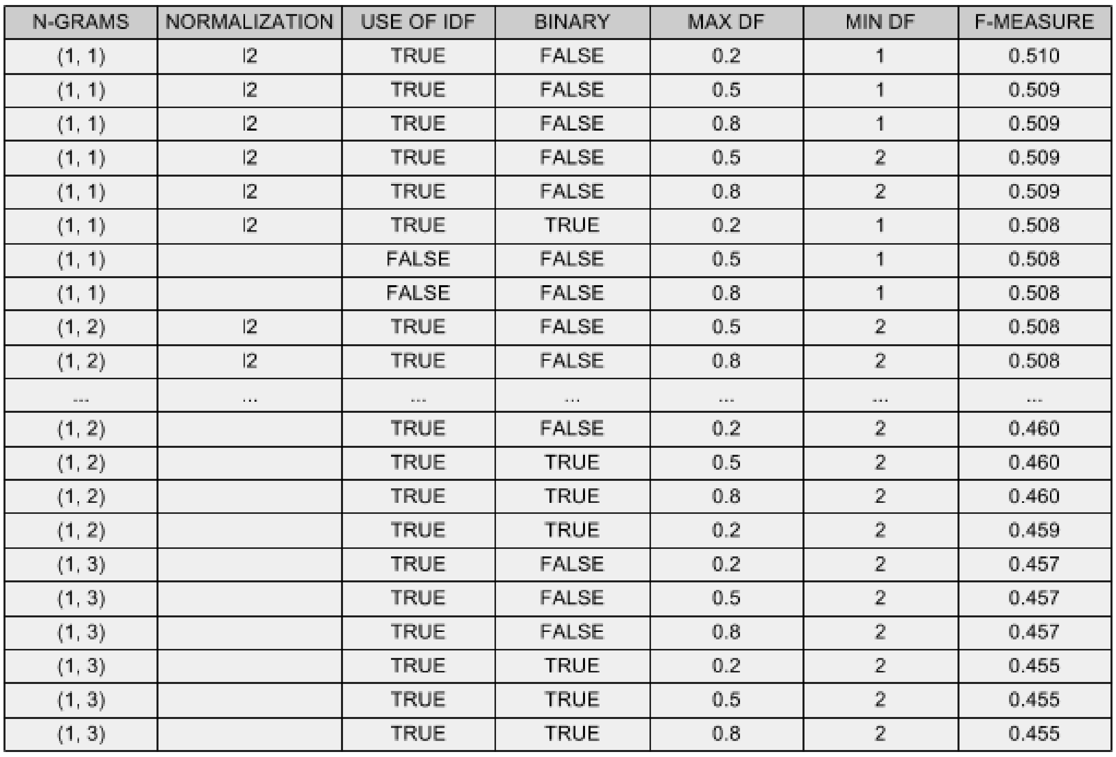
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 1)	l2	TRUE	FALSE	0.2	1	0.504
(1, 1)	l2	TRUE	FALSE	0.5	1	0.504
(1, 1)	l2	TRUE	FALSE	0.8	1	0.504
(1, 1)	l1	TRUE	FALSE	0.5	1	0.502
(1, 1)	l1	TRUE	FALSE	0.8	1	0.502
(1, 1)	l1	TRUE	FALSE	0.2	1	0.501
(1, 3)	l1	TRUE	TRUE	0.5	2	0.501
(1, 3)	l1	TRUE	TRUE	0.8	2	0.501
(1, 2)	l2	TRUE	FALSE	0.2	1	0.501
(1, 3)	l1	TRUE	TRUE	0.2	2	0.501
...	...	...	...	...	...	...
(1, 1)		TRUE	TRUE	0.2	1	0.483
(1, 1)		TRUE	TRUE	0.5	2	0.483
(1, 1)		TRUE	TRUE	0.8	2	0.483
(1, 3)		TRUE	FALSE	0.5	1	0.482
(1, 3)		TRUE	FALSE	0.8	1	0.482
(1, 3)		TRUE	FALSE	0.2	1	0.482
(1, 1)		TRUE	TRUE	0.2	2	0.482
(1, 3)		TRUE	TRUE	0.2	1	0.481
(1, 3)		TRUE	TRUE	0.5	1	0.480
(1, 3)		TRUE	TRUE	0.8	1	0.480

Table 29 Best and worst parameters for feature extraction from titles in a BOW model for MultinomialNB in a 5 class scale



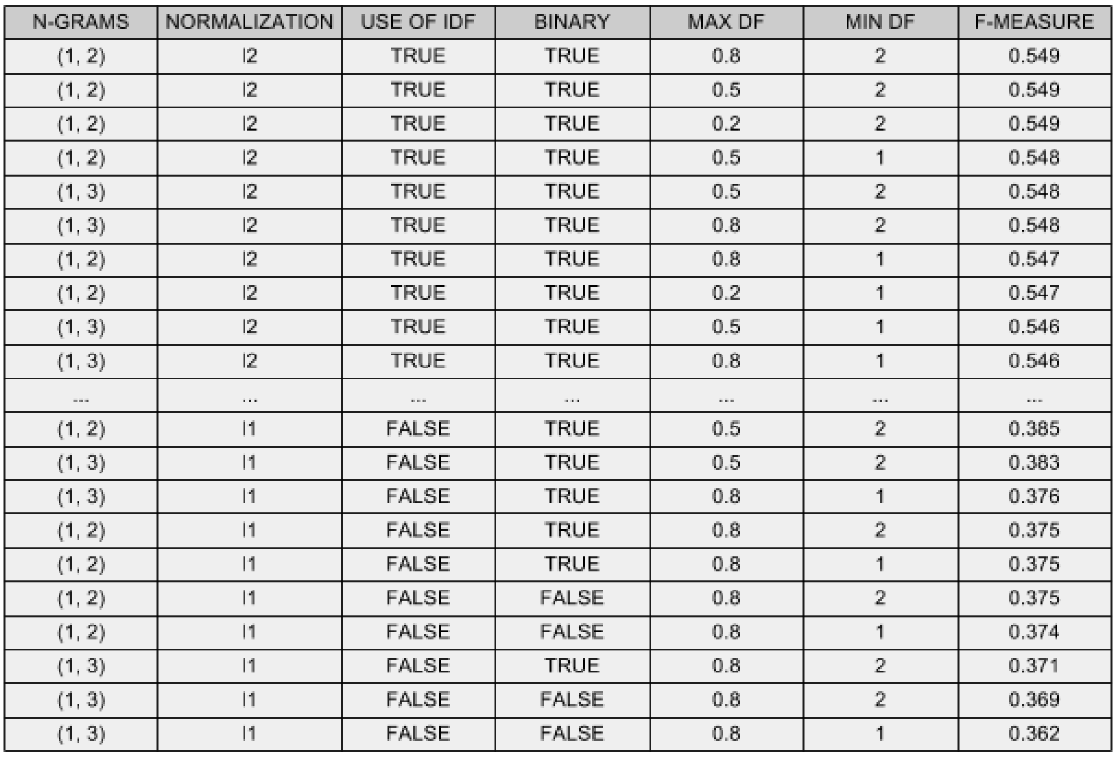
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 2)	l2	TRUE	TRUE	0.2	2	0.550
(1, 3)		FALSE	TRUE	0.2	2	0.550
(1, 3)		FALSE	TRUE	0.5	2	0.548
(1, 3)	l2	TRUE	TRUE	0.2	2	0.548
(1, 3)		FALSE	TRUE	0.8	2	0.547
(1, 2)		FALSE	TRUE	0.2	2	0.547
(1, 2)		FALSE	TRUE	0.5	2	0.545
(1, 3)		FALSE	FALSE	0.2	2	0.545
(1, 2)		FALSE	TRUE	0.8	2	0.545
(1, 3)	l2	TRUE	TRUE	0.5	2	0.545
...	...	...	...	...	...	...
(1, 2)	l1	FALSE	TRUE	0.8	2	0.447
(1, 2)	l1	FALSE	FALSE	0.8	1	0.447
(1, 3)	l1	FALSE	TRUE	0.5	2	0.446
(1, 2)	l1	FALSE	FALSE	0.8	2	0.446
(1, 3)	l1	FALSE	FALSE	0.5	2	0.442
(1, 3)	l1	FALSE	FALSE	0.5	1	0.441
(1, 3)	l1	FALSE	FALSE	0.8	2	0.435
(1, 3)	l1	FALSE	TRUE	0.8	2	0.433
(1, 3)	l1	FALSE	TRUE	0.8	1	0.428
(1, 3)	l1	FALSE	FALSE	0.8	1	0.427

Table 30 Best and worst parameters for feature extraction from review text in a BOW model for MultinomialNB in a 5 class scale



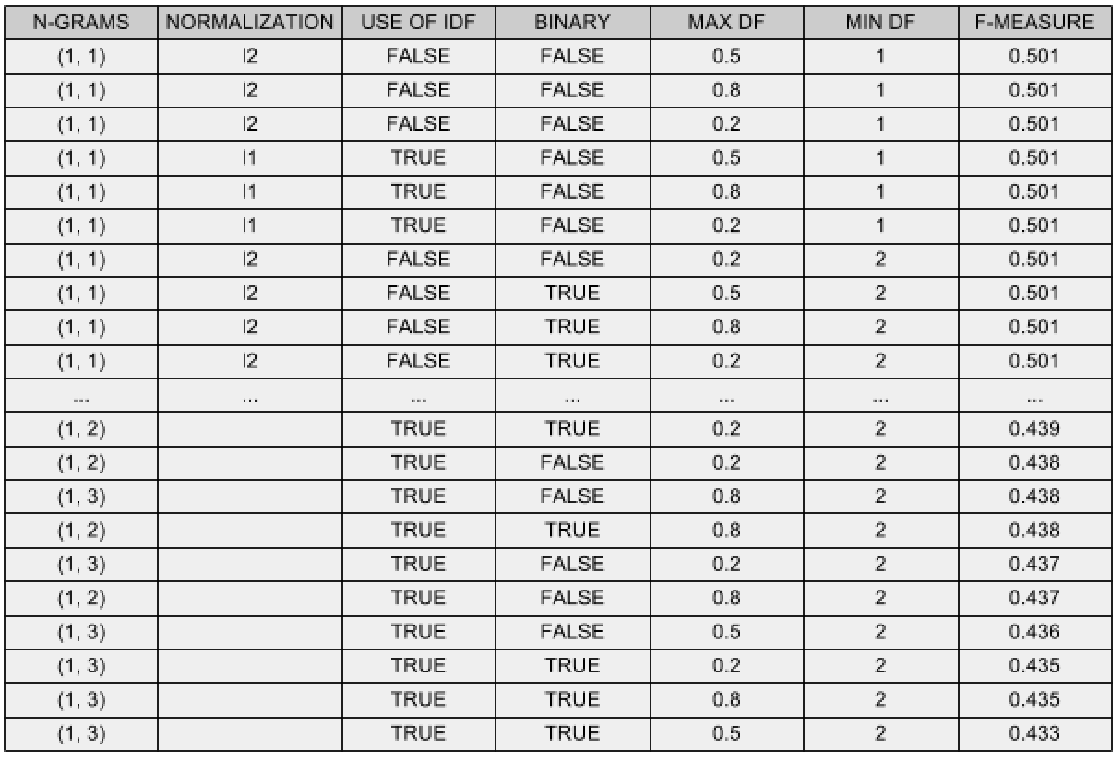
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 1)	l2	TRUE	FALSE	0.2	1	0.510
(1, 1)	l2	TRUE	FALSE	0.5	1	0.509
(1, 1)	l2	TRUE	FALSE	0.8	1	0.509
(1, 1)	l2	TRUE	FALSE	0.5	2	0.509
(1, 1)	l2	TRUE	FALSE	0.8	2	0.509
(1, 1)	l2	TRUE	TRUE	0.2	1	0.508
(1, 1)		FALSE	FALSE	0.5	1	0.508
(1, 1)		FALSE	FALSE	0.8	1	0.508
(1, 2)	l2	TRUE	FALSE	0.5	2	0.508
(1, 2)	l2	TRUE	FALSE	0.8	2	0.508
...	...	...	...	...	...	...
(1, 2)		TRUE	FALSE	0.2	2	0.460
(1, 2)		TRUE	TRUE	0.5	2	0.460
(1, 2)		TRUE	TRUE	0.8	2	0.460
(1, 2)		TRUE	TRUE	0.2	2	0.459
(1, 3)		TRUE	FALSE	0.2	2	0.457
(1, 3)		TRUE	FALSE	0.5	2	0.457
(1, 3)		TRUE	FALSE	0.8	2	0.457
(1, 3)		TRUE	TRUE	0.2	2	0.455
(1, 3)		TRUE	TRUE	0.5	2	0.455
(1, 3)		TRUE	TRUE	0.8	2	0.455

Table 31 Best and worst parameters for feature extraction from titles in a BOW model for Maximum Entropy in a 5 class scale



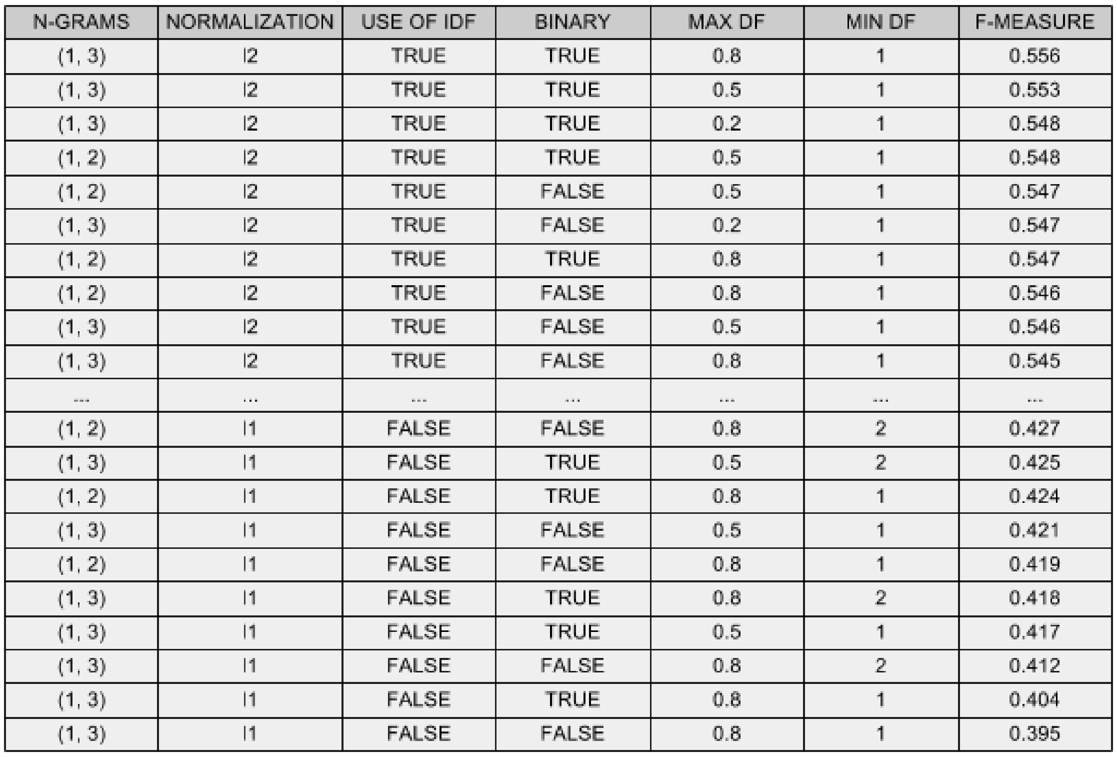
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 2)	l2	TRUE	TRUE	0.8	2	0.549
(1, 2)	l2	TRUE	TRUE	0.5	2	0.549
(1, 2)	l2	TRUE	TRUE	0.2	2	0.549
(1, 2)	l2	TRUE	TRUE	0.5	1	0.548
(1, 3)	l2	TRUE	TRUE	0.5	2	0.548
(1, 3)	l2	TRUE	TRUE	0.8	2	0.548
(1, 2)	l2	TRUE	TRUE	0.8	1	0.547
(1, 2)	l2	TRUE	TRUE	0.2	1	0.547
(1, 3)	l2	TRUE	TRUE	0.5	1	0.546
(1, 3)	l2	TRUE	TRUE	0.8	1	0.546
...	...	...	...	...	...	...
(1, 2)	l1	FALSE	TRUE	0.5	2	0.385
(1, 3)	l1	FALSE	TRUE	0.5	2	0.383
(1, 3)	l1	FALSE	TRUE	0.8	1	0.376
(1, 2)	l1	FALSE	TRUE	0.8	2	0.375
(1, 2)	l1	FALSE	TRUE	0.8	1	0.375
(1, 2)	l1	FALSE	FALSE	0.8	2	0.375
(1, 2)	l1	FALSE	FALSE	0.8	1	0.374
(1, 3)	l1	FALSE	TRUE	0.8	2	0.371
(1, 3)	l1	FALSE	FALSE	0.8	2	0.369
(1, 3)	l1	FALSE	FALSE	0.8	1	0.362

Table 32 Best and worst parameters for feature extraction from review text in a BOW model for Maximum Entropy in a 5 class scale



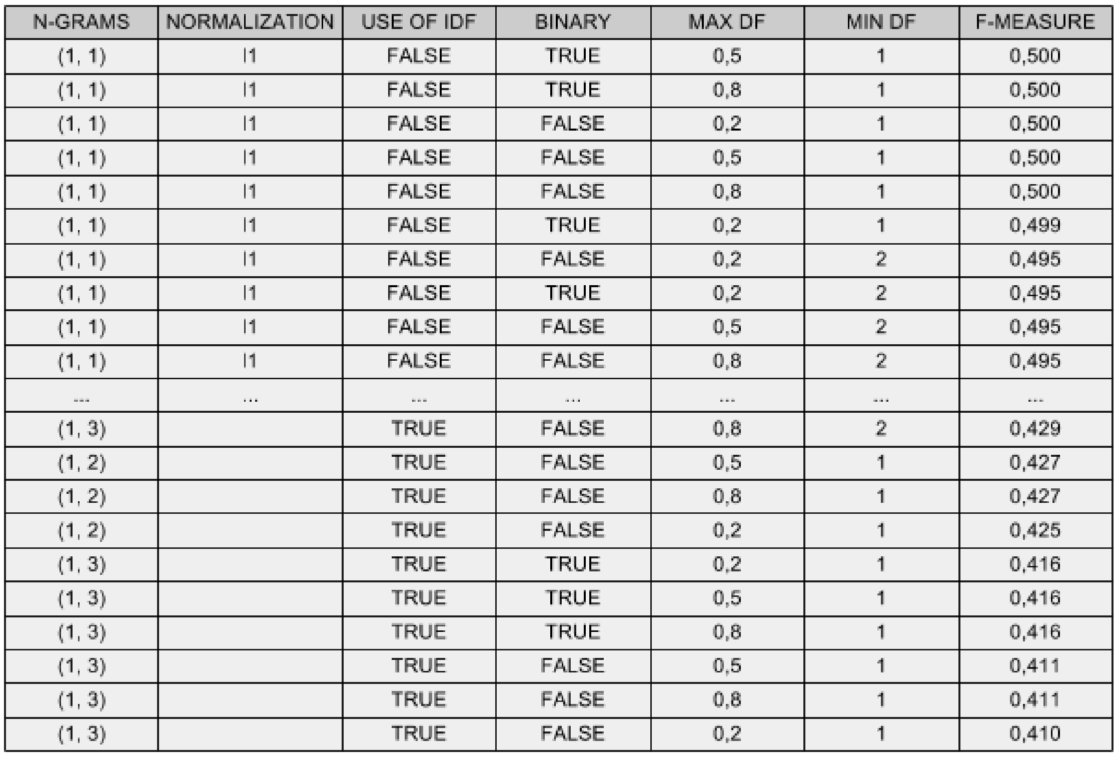
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 1)	l2	FALSE	FALSE	0.5	1	0.501
(1, 1)	l2	FALSE	FALSE	0.8	1	0.501
(1, 1)	l2	FALSE	FALSE	0.2	1	0.501
(1, 1)	l1	TRUE	FALSE	0.5	1	0.501
(1, 1)	l1	TRUE	FALSE	0.8	1	0.501
(1, 1)	l1	TRUE	FALSE	0.2	1	0.501
(1, 1)	l2	FALSE	FALSE	0.2	2	0.501
(1, 1)	l2	FALSE	TRUE	0.5	2	0.501
(1, 1)	l2	FALSE	TRUE	0.8	2	0.501
(1, 1)	l2	FALSE	TRUE	0.2	2	0.501
...	...	...	...	...	...	...
(1, 2)		TRUE	TRUE	0.2	2	0.439
(1, 2)		TRUE	FALSE	0.2	2	0.438
(1, 3)		TRUE	FALSE	0.8	2	0.438
(1, 2)		TRUE	TRUE	0.8	2	0.438
(1, 3)		TRUE	FALSE	0.2	2	0.437
(1, 2)		TRUE	FALSE	0.8	2	0.437
(1, 3)		TRUE	FALSE	0.5	2	0.436
(1, 3)		TRUE	TRUE	0.2	2	0.435
(1, 3)		TRUE	TRUE	0.8	2	0.435
(1, 3)		TRUE	TRUE	0.5	2	0.433

Table 33 Best and worst parameters for feature extraction from titles in a BOW model for SVM with a linear kernel in a 5 class scale



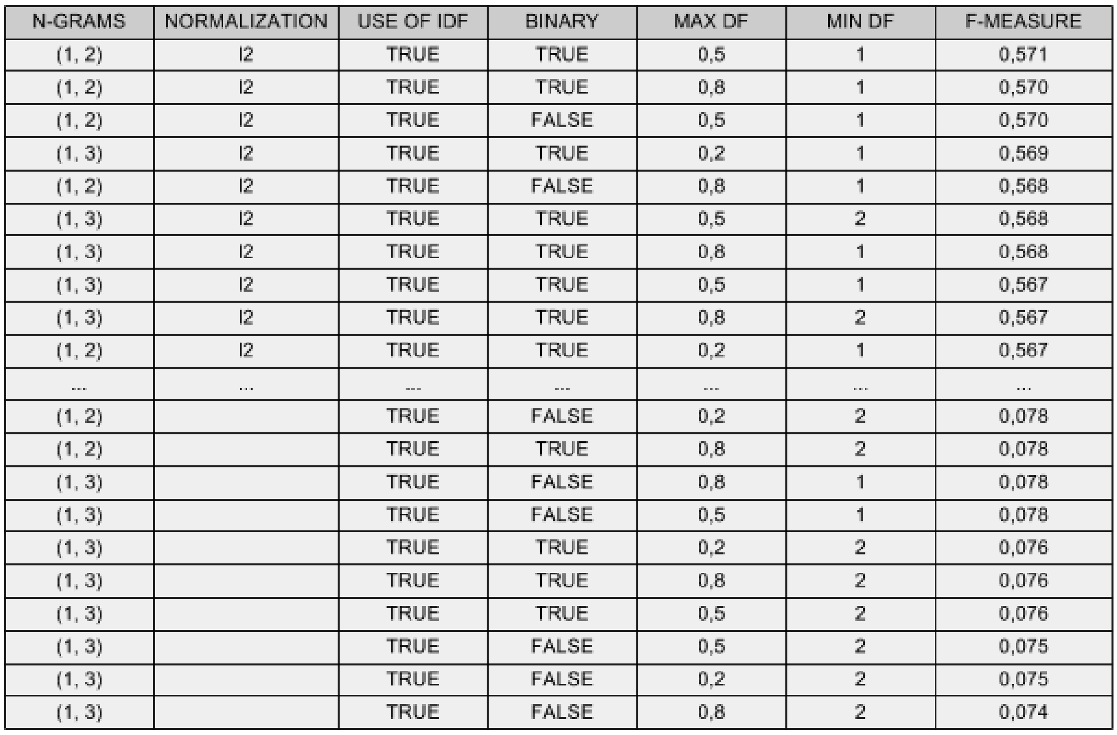
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 3)	l2	TRUE	TRUE	0.8	1	0.556
(1, 3)	l2	TRUE	TRUE	0.5	1	0.553
(1, 3)	l2	TRUE	TRUE	0.2	1	0.548
(1, 2)	l2	TRUE	TRUE	0.5	1	0.548
(1, 2)	l2	TRUE	FALSE	0.5	1	0.547
(1, 3)	l2	TRUE	FALSE	0.2	1	0.547
(1, 2)	l2	TRUE	TRUE	0.8	1	0.547
(1, 2)	l2	TRUE	FALSE	0.8	1	0.546
(1, 3)	l2	TRUE	FALSE	0.5	1	0.546
(1, 3)	l2	TRUE	FALSE	0.8	1	0.545
...	...	...	...	...	...	...
(1, 2)	l1	FALSE	FALSE	0.8	2	0.427
(1, 3)	l1	FALSE	TRUE	0.5	2	0.425
(1, 2)	l1	FALSE	TRUE	0.8	1	0.424
(1, 3)	l1	FALSE	FALSE	0.5	1	0.421
(1, 2)	l1	FALSE	FALSE	0.8	1	0.419
(1, 3)	l1	FALSE	TRUE	0.8	2	0.418
(1, 3)	l1	FALSE	TRUE	0.5	1	0.417
(1, 3)	l1	FALSE	FALSE	0.8	2	0.412
(1, 3)	l1	FALSE	TRUE	0.8	1	0.404
(1, 3)	l1	FALSE	FALSE	0.8	1	0.395

Table 34 Best and worst parameters for feature extraction from review text in a BOW model for SVM with a linear kernel in a 5 class scale



N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 1)	l1	FALSE	TRUE	0,5	1	0,500
(1, 1)	l1	FALSE	TRUE	0,8	1	0,500
(1, 1)	l1	FALSE	FALSE	0,2	1	0,500
(1, 1)	l1	FALSE	FALSE	0,5	1	0,500
(1, 1)	l1	FALSE	FALSE	0,8	1	0,500
(1, 1)	l1	FALSE	TRUE	0,2	1	0,499
(1, 1)	l1	FALSE	FALSE	0,2	2	0,495
(1, 1)	l1	FALSE	TRUE	0,2	2	0,495
(1, 1)	l1	FALSE	FALSE	0,5	2	0,495
(1, 1)	l1	FALSE	FALSE	0,8	2	0,495
...	...	...	...	...	...	...
(1, 3)		TRUE	FALSE	0,8	2	0,429
(1, 2)		TRUE	FALSE	0,5	1	0,427
(1, 2)		TRUE	FALSE	0,8	1	0,427
(1, 2)		TRUE	FALSE	0,2	1	0,425
(1, 3)		TRUE	TRUE	0,2	1	0,416
(1, 3)		TRUE	TRUE	0,5	1	0,416
(1, 3)		TRUE	TRUE	0,8	1	0,416
(1, 3)		TRUE	FALSE	0,5	1	0,411
(1, 3)		TRUE	FALSE	0,8	1	0,411
(1, 3)		TRUE	FALSE	0,2	1	0,410

Table 35 Best and worst parameters for feature extraction from titles in a BOW model for SVM with an RBF kernel in a 5 class scale



N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 2)	l2	TRUE	TRUE	0,5	1	0,571
(1, 2)	l2	TRUE	TRUE	0,8	1	0,570
(1, 2)	l2	TRUE	FALSE	0,5	1	0,570
(1, 3)	l2	TRUE	TRUE	0,2	1	0,569
(1, 2)	l2	TRUE	FALSE	0,8	1	0,568
(1, 3)	l2	TRUE	TRUE	0,5	2	0,568
(1, 3)	l2	TRUE	TRUE	0,8	1	0,568
(1, 3)	l2	TRUE	TRUE	0,5	1	0,567
(1, 3)	l2	TRUE	TRUE	0,8	2	0,567
(1, 2)	l2	TRUE	TRUE	0,2	1	0,567
...	...	...	...	...	...	...
(1, 2)		TRUE	FALSE	0,2	2	0,078
(1, 2)		TRUE	TRUE	0,8	2	0,078
(1, 3)		TRUE	FALSE	0,8	1	0,078
(1, 3)		TRUE	FALSE	0,5	1	0,078
(1, 3)		TRUE	TRUE	0,2	2	0,076
(1, 3)		TRUE	TRUE	0,8	2	0,076
(1, 3)		TRUE	TRUE	0,5	2	0,076
(1, 3)		TRUE	FALSE	0,5	2	0,075
(1, 3)		TRUE	FALSE	0,2	2	0,075
(1, 3)		TRUE	FALSE	0,8	2	0,074

Table 36 Best and worst parameters for feature extraction from review text in a BOW model for SVM with an RBF kernel in a 5 class scale

N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 2)	l2	TRUE	TRUE	0.2	1	0.868
(1, 2)	l2	TRUE	TRUE	0.5	1	0.868
(1, 2)	l2	TRUE	TRUE	0.8	1	0.868
(1, 3)	l2	TRUE	TRUE	0.2	1	0.867
(1, 3)	l2	TRUE	TRUE	0.5	1	0.867
(1, 3)	l2	TRUE	TRUE	0.8	1	0.867
(1, 2)	l2	TRUE	FALSE	0.2	1	0.866
(1, 3)		FALSE	TRUE	0.2	1	0.866
(1, 2)	l2	TRUE	FALSE	0.5	1	0.866
(1, 2)	l2	TRUE	FALSE	0.8	1	0.866
...	...	...	...	...	...	...
(1, 2)	l1	FALSE	FALSE	0.8	1	0.847
(1, 2)	l1	FALSE	FALSE	0.2	2	0.847
(1, 2)	l1	FALSE	FALSE	0.5	2	0.846
(1, 2)	l1	FALSE	FALSE	0.8	2	0.846
(1, 3)	l1	FALSE	FALSE	0.2	2	0.844
(1, 3)	l1	FALSE	FALSE	0.2	1	0.843
(1, 3)	l1	FALSE	FALSE	0.5	2	0.843
(1, 3)	l1	FALSE	FALSE	0.8	2	0.843
(1, 3)	l1	FALSE	FALSE	0.5	1	0.842
(1, 3)	l1	FALSE	FALSE	0.8	1	0.842

Table 37 Best and worst parameters for feature extraction from titles in a BOW model for MultinomialNB in a binary scale

N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 3)		TRUE	TRUE	0.2	1	0.901
(1, 3)		TRUE	TRUE	0.5	1	0.901
(1, 3)		TRUE	TRUE	0.8	1	0.901
(1, 3)		TRUE	FALSE	0.2	1	0.901
(1, 3)		TRUE	FALSE	0.8	1	0.899
(1, 3)		TRUE	FALSE	0.5	1	0.899
(1, 2)		TRUE	FALSE	0.2	1	0.898
(1, 2)		TRUE	FALSE	0.2	2	0.898
(1, 2)		TRUE	FALSE	0.5	1	0.897
(1, 2)		TRUE	TRUE	0.2	2	0.897
...	...	...	...	...	...	...
(1, 1)	l1	FALSE	FALSE	0.8	2	0.820
(1, 2)	l2	FALSE	FALSE	0.8	1	0.820
(1, 3)	l2	FALSE	FALSE	0.5	1	0.812
(1, 3)	l1	FALSE	FALSE	0.5	2	0.812
(1, 3)	l1	FALSE	FALSE	0.5	1	0.811
(1, 2)	l1	FALSE	FALSE	0.8	1	0.805
(1, 2)	l1	FALSE	FALSE	0.8	2	0.804
(1, 3)	l2	FALSE	FALSE	0.8	1	0.802
(1, 3)	l1	FALSE	FALSE	0.8	2	0.794
(1, 3)	l1	FALSE	FALSE	0.8	1	0.790

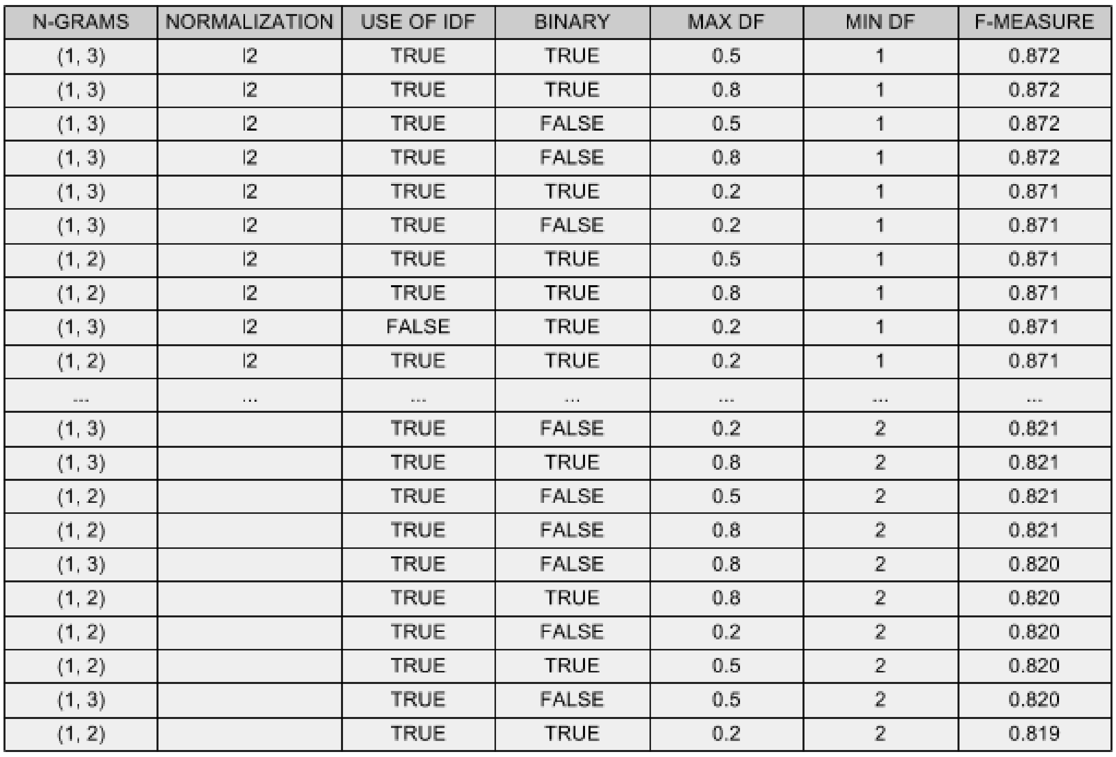
Table 38 Best and worst parameters for feature extraction from review text in a BOW model for MultinomialNB in a binary scale

N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 2)		FALSE	FALSE	0.5	1	0.866
(1, 2)		FALSE	FALSE	0.8	1	0.866
(1, 2)		FALSE	TRUE	0.5	1	0.865
(1, 2)		FALSE	TRUE	0.8	1	0.865
(1, 3)		FALSE	TRUE	0.5	1	0.865
(1, 3)		FALSE	TRUE	0.8	1	0.865
(1, 2)		FALSE	TRUE	0.2	1	0.865
(1, 2)		FALSE	FALSE	0.2	1	0.865
(1, 2)		TRUE	FALSE	0.5	1	0.864
(1, 2)		TRUE	FALSE	0.8	1	0.864
...	...	...	...	...	...	...
(1, 3)	l1	FALSE	FALSE	0.5	2	0.823
(1, 3)	l1	FALSE	FALSE	0.8	2	0.823
(1, 2)	l1	FALSE	FALSE	0.5	1	0.820
(1, 2)	l1	FALSE	FALSE	0.8	1	0.820
(1, 3)	l1	FALSE	TRUE	0.2	1	0.818
(1, 3)	l1	FALSE	TRUE	0.5	1	0.816
(1, 3)	l1	FALSE	TRUE	0.8	1	0.816
(1, 3)	l1	FALSE	FALSE	0.2	1	0.815
(1, 3)	l1	FALSE	FALSE	0.5	1	0.812
(1, 3)	l1	FALSE	FALSE	0.8	1	0.812

Table 39 Best and worst parameters for feature extraction from titles in a BOW model for Maximum Entropy in a binary scale

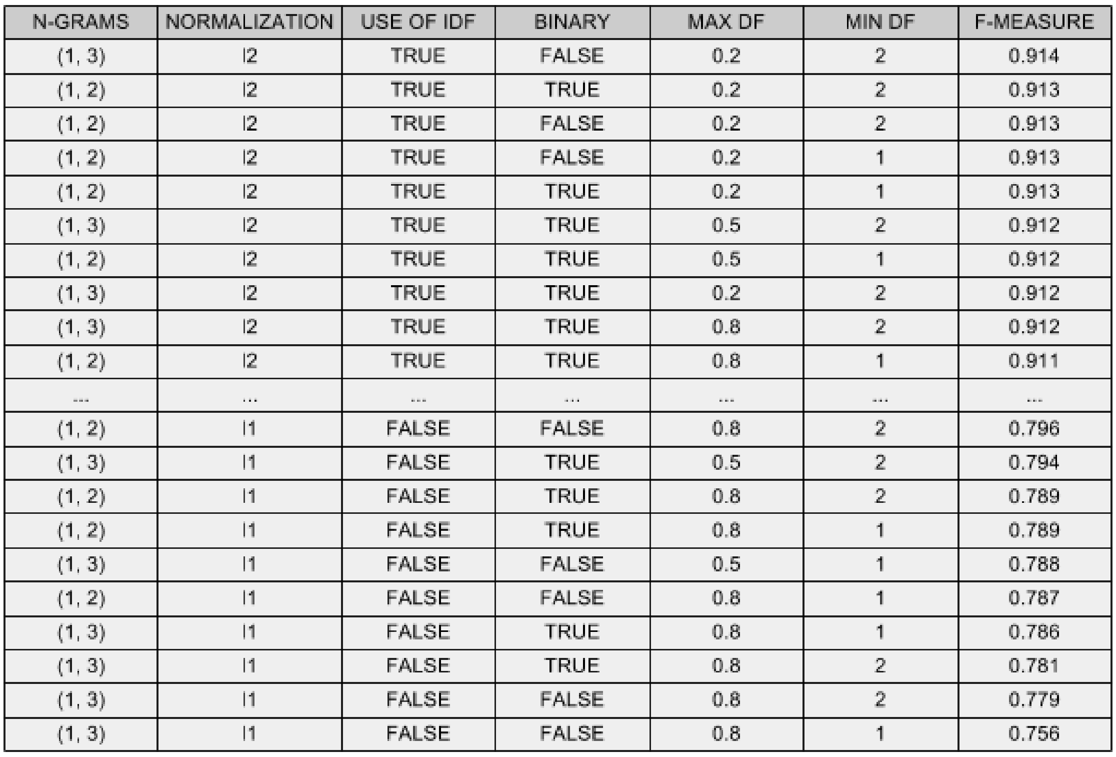
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 3)		TRUE	FALSE	0.2	2	0.910
(1, 2)		TRUE	TRUE	0.5	2	0.909
(1, 2)		TRUE	FALSE	0.2	1	0.909
(1, 2)		TRUE	TRUE	0.8	2	0.909
(1, 3)		TRUE	FALSE	0.2	1	0.909
(1, 3)		TRUE	TRUE	0.2	2	0.909
(1, 2)		TRUE	TRUE	0.2	2	0.908
(1, 2)		TRUE	TRUE	0.2	1	0.908
(1, 2)		TRUE	TRUE	0.5	1	0.908
(1, 2)		TRUE	FALSE	0.2	2	0.908
...	...	...	...	...	...	...
(1, 1)	l1	FALSE	FALSE	0.8	1	0.766
(1, 1)	l1	FALSE	TRUE	0.8	2	0.755
(1, 2)	l1	FALSE	TRUE	0.8	1	0.754
(1, 1)	l1	FALSE	TRUE	0.8	1	0.754
(1, 3)	l1	FALSE	TRUE	0.8	2	0.752
(1, 2)	l1	FALSE	FALSE	0.8	2	0.750
(1, 2)	l1	FALSE	TRUE	0.8	2	0.749
(1, 2)	l1	FALSE	FALSE	0.8	1	0.746
(1, 3)	l1	FALSE	FALSE	0.8	2	0.742
(1, 3)	l1	FALSE	FALSE	0.8	1	0.727

Table 40 Best and worst parameters for feature extraction from review text in a BOW model for Maximum Entropy in a binary scale



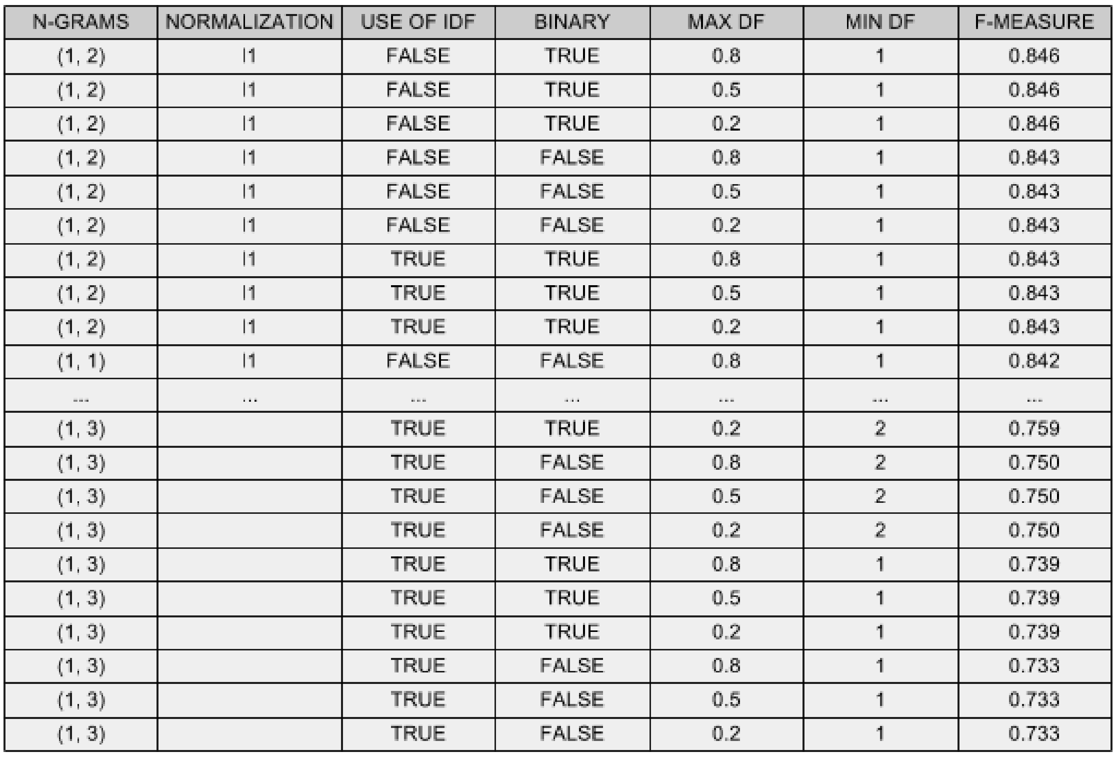
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 3)	l2	TRUE	TRUE	0.5	1	0.872
(1, 3)	l2	TRUE	TRUE	0.8	1	0.872
(1, 3)	l2	TRUE	FALSE	0.5	1	0.872
(1, 3)	l2	TRUE	FALSE	0.8	1	0.872
(1, 3)	l2	TRUE	TRUE	0.2	1	0.871
(1, 3)	l2	TRUE	FALSE	0.2	1	0.871
(1, 2)	l2	TRUE	TRUE	0.5	1	0.871
(1, 2)	l2	TRUE	TRUE	0.8	1	0.871
(1, 3)	l2	FALSE	TRUE	0.2	1	0.871
(1, 2)	l2	TRUE	TRUE	0.2	1	0.871
...	...	...	...	...	...	...
(1, 3)		TRUE	FALSE	0.2	2	0.821
(1, 3)		TRUE	TRUE	0.8	2	0.821
(1, 2)		TRUE	FALSE	0.5	2	0.821
(1, 2)		TRUE	FALSE	0.8	2	0.821
(1, 3)		TRUE	FALSE	0.8	2	0.820
(1, 2)		TRUE	TRUE	0.8	2	0.820
(1, 2)		TRUE	FALSE	0.2	2	0.820
(1, 2)		TRUE	TRUE	0.5	2	0.820
(1, 3)		TRUE	FALSE	0.5	2	0.820
(1, 2)		TRUE	TRUE	0.2	2	0.819

Table 41 Best and worst parameters for feature extraction from titles in a BOW model for SVM with a linear kernel in a binary scale



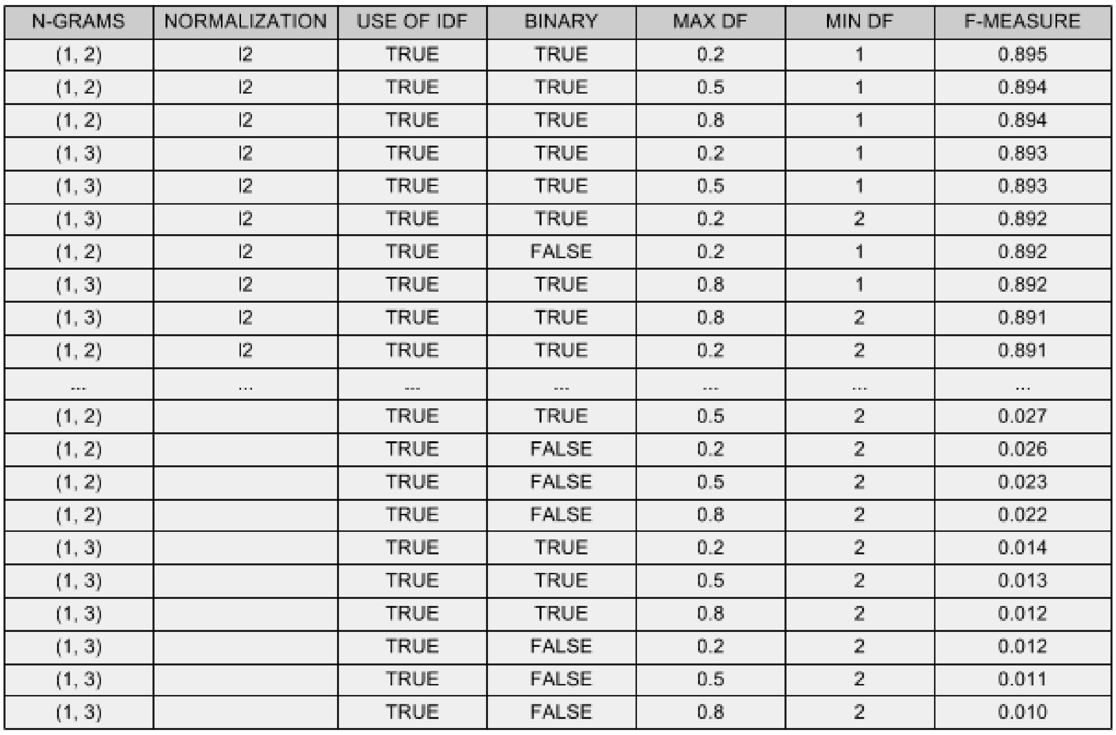
N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 3)	l2	TRUE	FALSE	0.2	2	0.914
(1, 2)	l2	TRUE	TRUE	0.2	2	0.913
(1, 2)	l2	TRUE	FALSE	0.2	2	0.913
(1, 2)	l2	TRUE	FALSE	0.2	1	0.913
(1, 2)	l2	TRUE	TRUE	0.2	1	0.913
(1, 3)	l2	TRUE	TRUE	0.5	2	0.912
(1, 2)	l2	TRUE	TRUE	0.5	1	0.912
(1, 3)	l2	TRUE	TRUE	0.2	2	0.912
(1, 3)	l2	TRUE	TRUE	0.8	2	0.912
(1, 2)	l2	TRUE	TRUE	0.8	1	0.911
...	...	...	...	...	...	...
(1, 2)	l1	FALSE	FALSE	0.8	2	0.796
(1, 3)	l1	FALSE	TRUE	0.5	2	0.794
(1, 2)	l1	FALSE	TRUE	0.8	2	0.789
(1, 2)	l1	FALSE	TRUE	0.8	1	0.789
(1, 3)	l1	FALSE	FALSE	0.5	1	0.788
(1, 2)	l1	FALSE	FALSE	0.8	1	0.787
(1, 3)	l1	FALSE	TRUE	0.8	1	0.786
(1, 3)	l1	FALSE	TRUE	0.8	2	0.781
(1, 3)	l1	FALSE	FALSE	0.8	2	0.779
(1, 3)	l1	FALSE	FALSE	0.8	1	0.756

Table 42 Best and worst parameters for feature extraction from review text in a BOW model for SVM with a linear kernel in a binary scale



N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 2)	l1	FALSE	TRUE	0.8	1	0.846
(1, 2)	l1	FALSE	TRUE	0.5	1	0.846
(1, 2)	l1	FALSE	TRUE	0.2	1	0.846
(1, 2)	l1	FALSE	FALSE	0.8	1	0.843
(1, 2)	l1	FALSE	FALSE	0.5	1	0.843
(1, 2)	l1	FALSE	FALSE	0.2	1	0.843
(1, 2)	l1	TRUE	TRUE	0.8	1	0.843
(1, 2)	l1	TRUE	TRUE	0.5	1	0.843
(1, 2)	l1	TRUE	TRUE	0.2	1	0.843
(1, 1)	l1	FALSE	FALSE	0.8	1	0.842
...	...	...	...	...	...	...
(1, 3)		TRUE	TRUE	0.2	2	0.759
(1, 3)		TRUE	FALSE	0.8	2	0.750
(1, 3)		TRUE	FALSE	0.5	2	0.750
(1, 3)		TRUE	FALSE	0.2	2	0.750
(1, 3)		TRUE	TRUE	0.8	1	0.739
(1, 3)		TRUE	TRUE	0.5	1	0.739
(1, 3)		TRUE	TRUE	0.2	1	0.739
(1, 3)		TRUE	FALSE	0.8	1	0.733
(1, 3)		TRUE	FALSE	0.5	1	0.733
(1, 3)		TRUE	FALSE	0.2	1	0.733

Table 43 Best and worst parameters for feature extraction from titles in a BOW model for SVM with an RBF kernel in a binary scale



N-GRAMS	NORMALIZATION	USE OF IDF	BINARY	MAX DF	MIN DF	F-MEASURE
(1, 2)	l2	TRUE	TRUE	0.2	1	0.895
(1, 2)	l2	TRUE	TRUE	0.5	1	0.894
(1, 2)	l2	TRUE	TRUE	0.8	1	0.894
(1, 3)	l2	TRUE	TRUE	0.2	1	0.893
(1, 3)	l2	TRUE	TRUE	0.5	1	0.893
(1, 3)	l2	TRUE	TRUE	0.2	2	0.892
(1, 2)	l2	TRUE	FALSE	0.2	1	0.892
(1, 3)	l2	TRUE	TRUE	0.8	1	0.892
(1, 3)	l2	TRUE	TRUE	0.8	2	0.891
(1, 2)	l2	TRUE	TRUE	0.2	2	0.891
...	...	...	...	...	...	...
(1, 2)		TRUE	TRUE	0.5	2	0.027
(1, 2)		TRUE	FALSE	0.2	2	0.026
(1, 2)		TRUE	FALSE	0.5	2	0.023
(1, 2)		TRUE	FALSE	0.8	2	0.022
(1, 3)		TRUE	TRUE	0.2	2	0.014
(1, 3)		TRUE	TRUE	0.5	2	0.013
(1, 3)		TRUE	TRUE	0.8	2	0.012
(1, 3)		TRUE	FALSE	0.2	2	0.012
(1, 3)		TRUE	FALSE	0.5	2	0.011
(1, 3)		TRUE	FALSE	0.8	2	0.010

Table 44 Best and worst parameters for feature extraction from review text in a BOW model for SVM with an RBF kernel in a binary scale

SVM classifiers still need the same normalizations to achieve their best binary classification, but extra information from bigrams and trigrams, even from title vectors, seems beneficial.

Combining the vectors of review titles and text benefited all classifiers. The extraction process of the vectors for each classifier was performed with the best achieving parameters for the particular classifier. The classifier that most benefitted from the combined vectors was SVM with an RBF kernel.

5.4.4 Mixed Model

The fact that our classifiers benefited from the use of features extracted separately from the review title and text, urged us to try combining the BOW features with the features extracted with the other models. For each classifier, the best performing features from each model were selected and combined. As we have already seen in Table 7 through Table 10 all classifiers managed better f-measure scores when trained and tested with the combined features of all the models. The problem with the mixed method is that all the feature extraction models must be trained separately, which is a time-consuming process, and all the models must be kept in memory, which demands a lot of memory, since multiple copies of the dictionaries have to be stored.

5.5 Synopsis

Throughout this thesis we have shown that classifying text extracted from online hotel reviews is a feasible process. While lexicon based approaches have been reported to perform well, our SentiWordNet model didn’t manage to perform very well. We consider the possibility that the method that was employed to aggregate sentiment to the document level might have been inappropriate. Also, our model could have extracted more features by using part-of-speech tags. On the other hand, document level aggregation of feature vectors extracted with Word2Vec, in out Doc2Vec model, performed well enough and up to par with models encountered in the literature about sentiment analysis on hotel reviews. We were pleasantly surprised with the performance of the BOW model; not only did it perform the best out of the other models, the f-measure score that was achieved appears too good to be true. By combining the features of all the models that were used, we managed to benefit all classifiers.

6

Technical Details

In order to help the reader understand a little bit more the tasks that we have undertaken, we will present a few details on the implementation steps that were completed. While sentiment analysis is not very demanding on time constraints, all text classification tasks can take o very long time. A modern system has o lot of computational power and vast amounts of RAM, but many processes a time consuming and text representation can become cumbersome.

6.1 Implementation Details

Sentiment analysis is one of the hot topics in machine learning nowadays. A lot of tools are developed from and for researchers to help perform all manner of natural language processing. Also, many machine learning algorithms are implemented for almost all programming environments. Our first task was to detect as many of them as possible and decide which ones were better suited for us.

6.1.1 Platform and Programming Tools

We decided that the environment that better suited us is the Python programming language. Python is a simple yet powerful programming language with excellent functionality for processing linguistic data. Python has a shallow learning curve, its syntax and semantics are transparent, and it has good string-handling functionality. As an interpreted language, Python facilitates interactive exploration. It is an object-oriented language and permits data and methods to be encapsulated and re-used easily. As a dynamic language, Python permits attributes to be added to objects on the fly, and permits variables to be typed dynamically, facilitating rapid development. Python comes with an extensive standard library, including components for graphical programming, numerical processing, and web connectivity. Python is heavily used in industry, scientific research, and education around the world and is often praised for the way it facilitates productivity, quality, and maintainability of software. There is a multitude of libraries for natural language processing and machine learning. The PyCharm programming environment was used for coding and running the scripts. The library that was used for preprocessing the text was NLTK 3.0 for tokenization and sentence segmentation.

NLTK defines an infrastructure that can be used to build NLP programs in Python. It provides basic classes for representing data relevant to natural language processing; standard interfaces for performing tasks such as part-of-speech tagging, syntactic parsing, and text classification; and standard implementations for each task which can be combined to solve complex problems. NLTK comes with extensive documentation. The website provides API documentation that covers every module, class and function in the toolkit, specifying parameters and giving examples of usage. More concise and to the point is the online book. Following the book, one can learn both basic Python programming and basic Natural Language Processing.

NTLK offers bindings to the WordNet and SentiWordNet lexicons which were used for creating a sentiment aware dictionary. Spell checking was implemented with Enchant, using the en-us lexicon. The data were stored in Numpy and Scipy Sparse arrays. The classification algorithms that we used were from Scikit’s Sklearn library. Graphs were drawn using Matplotlib. For downloading the data from TripAdvisor, we implemented a system using urllib2 and BeautifulSoup.

The hardware that we had at our disposal was a two core personal computer with 8GB of RAM. On that we had a Linux Mint 17 installation using an xfce graphical environment.

6.1.2 Data Representation

One of the biggest challenges for our system was the memory footprint of the models. All of the models need to maintain a dictionary of all tokens encountered during the training stage. The spell check preprocessing stage maintains a dictionary with all proposed replacements. The TF*IDF model maintains counts of terms per document in order to calculate IDF. The sentimental lexicon model is of course a dictionary where sentiment information about all tokens is stored. The Doc2Vec also maintains a dictionary with the representation of hidden aspects for every term encountered in the corpus. The classifiers also maintain information about the features. Naïve bayes would store prior probabilities about all features in the corpus.

The number of features extracted from the 10000 review corpus reached 13000 unique unigrams, and 675000 bigrams and trigrams that appear at most in 20% of the documents. Most of the values in these huge vectors are comprised of 0s. In order to make the models viable for a machine with only 8GB of RAM, sparse representations of data were chosen, where only the non zero values in an array are stored. Because of the overhead, only arrays with a degree of sparseness more than 70% can conserve memory this way.

7

Conclusions

To recapitulate, this thesis investigated the usage of machine learning algorithms on the task of extracting the sentiment orientation of hotel reviews found online.

7.1 Synopsis and Conclusions

We implemented different models of feature extraction and investigated the potency of information that features extracted by them have on classifying text reviews. We performed grid search experiments in defining the parameters of these feature extraction methods in order for them to yield the best possible features for the task at hand. We compared classification results of 4 algorithms on the feature extraction methods. With the exception of multinomial Naïve Bayes being a poor fit for the SentiWordNet model, the classifiers had comparable results on the same features. The differentiating factor of the classification process was the quality of the features inputted to the algorithms. The f-measure of the binary classification task, surpassed 90% and therefore we conclude that binary classification of text reviews based on the general sentiment transferred is feasible. On the task of classifying the reviews in a 1-5 scale, f-measure barely reached 60%, therefore the feature extracting methods have room for improvement.

7.2 Future Work

The dataset that we had to work with also included information about predefined aspects of the hotel experience that the visitors could grade, for example the satisfaction about the hotel’s location, or the satisfaction from the hotel staff. It is our intention to further investigate on models that can encapsulate the specificity of this information and create a system that can detect sentiment orientation on a pre aspect basis.

8

References

[1] ‘Tourism’, *Wikipedia*. Wikipedia, 31-May-2015.

[2] ‘OTA’s share of hotel industry revenue in the U.S. 2004-2011 | Statistic.’ [Online]. Available: http://www.statista.com/statistics/217676/united-states-market-share-of-online-travel-agencies. [Accessed: 31-May-2015].

[3] ‘Fact Sheet’, *TripAdvisor.com*. [Online]. Available: http://www.tripadvisor.com/PressCenter-c4-Fact_Sheet.html. [Accessed: 31-May-2015].

[4] ‘Booking’s Best 2015 - Planet Earth's #1 Accommodation’, *Booking.com*. [Online]. Available: http://www.thebookingtruth.com. [Accessed: 25-Apr-2015].

[5] M. Tsytsarau and T. Palpanas, ‘Survey on mining subjective data on the web’, *Data Mining and Knowledge Discovery*, vol. 24, no. 3, pp. 478–514, May 2012.

[6] T. Wilson, J. Wiebe, and P. Hoffmann, ‘Recognizing contextual polarity in phrase-level sentiment analysis’, *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT ’05*, 2005.

[7] B. Liu, ‘Opinion Mining and Sentiment Analysis’, *Web Data Mining*, pp. 459–526, Jan. 2011.

[8] M. Hagenau, M. Liebmann, and D. Neumann, ‘Automated news reading: Stock price prediction based on financial news using context-capturing features’, *Decision Support Systems*, vol. 55, no. 3, pp. 685–697, 2013.

[9] T. Xu, Q. Peng, and Y. Cheng, ‘Identifying the semantic orientation of terms using S-HAL for sentiment analysis’, *Knowledge-Based Systems*, vol. 35, pp. 279–289, 2012.

[10] I. Maks and P. Vossen, ‘A lexicon model for deep sentiment analysis and opinion mining applications’, *Decision Support Systems*, vol. 53, no. 4, pp. 680–688, 2012.

[11] P. D. Turney, ‘Thumbs up or thumbs down?’, *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02*, 2001.

[12] B. Pang, L. Lee, and S. Vaithyanathan, ‘Thumbs up?’, *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP ’02*, 2002.

[13] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, ‘Lexicon-Based Methods for Sentiment Analysis’, *Computational Linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[14] R. M. Tong, ‘An Operational System for Detecting and Tracking Opinions in On-line Discussion’, in *Proceedings of the Workshop on Operational Text Classification (OTC)*, 2001.
[15] V. Hatzivassiloglou and K. R. McKeown, ‘Predicting the semantic orientation of adjectives’, *Proceedings of the 35th annual meeting on Association for Computational Linguistics -*, 1997.
[16] P. D. Turney and M. L. Littman, ‘Measuring praise and criticism’, *ACM Transactions on Information Systems*, vol. 21, no. 4, pp. 315–346, 2003.
[17] J. Wiebe, ‘Learning subjective adjectives from corpora’, in *AAAI/IAAI*, 2000.
[18] M. Hu and B. Liu, ‘Mining and summarizing customer reviews’, *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’04*, 2004.
[19] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, ‘Lexicon-Based Methods for Sentiment Analysis’, *Computational Linguistics*, vol. 37, no. 2, pp. 267–307, 2011.
[20] C. Whitelaw, N. Garg, and S. Argamon, ‘Using appraisal groups for sentiment analysis’, *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM ’05*, 2005.
[21] J. R. Martin, P. R. R. White, and R. R. P. White, *The language of evaluation: appraisal in English*. New York: Palgrave Macmillan, 2005.
[22] Q. Ye, Z. Zhang, and R. Law, ‘Sentiment classification of online reviews to travel destinations by supervised machine learning approaches’, *Expert Systems with Applications*, vol. 36, no. 3, pp. 6527–6535, 2009.
[23] R. Prabowo and M. Thelwall, ‘Sentiment analysis: A combined approach’, *Journal of Informetrics*, vol. 3, no. 2, pp. 143–157, 2009.
[24] A. Weichselbraun, S. Gindl, and A. Scharl, ‘Extracting and Grounding Contextualized Sentiment Lexicons’, *IEEE Intelligent Systems*, vol. 28, no. 2, pp. 39–46, 2013.
[25] R. Y. K. Lau, C. L. Lai, P. B. Bruza, and K. F. Wong, ‘Leveraging web 2.0 data for scalable semi-supervised learning of domain-specific sentiment lexicons’, *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM ’11*, 2011.
[26] D. Gräbner, M. Zanker, G. Fliedl, and M. Fuchs, ‘Classification of Customer Reviews based on Sentiment Analysis’, *Information and Communication Technologies in Tourism 2012*, pp. 460–470, 2012.
[27] M. Gamon, ‘Sentiment classification on customer feedback data’, *Proceedings of the 20th international conference on Computational Linguistics - COLING ’04*, 2004.
[28] E. Bjørkelund, T. H. Burnett, and K. Nørvåg, ‘A study of opinion mining and visualization of hotel reviews’, *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services - IIWAS ’12*, 2012.
[29] N. Friedman, D. Geiger, and M. Goldszmidt, ‘Bayesian network classifiers’, *Machine Learning*, vol. 29, no. 2/3, pp. 131–163, 1997.
[30] P. Domingos and M. Pazzani, ‘On the optimality of the simple Bayesian classifier under zero-one loss’, *Machine Learning*, vol. 29, no. 2–3, pp. 103–130, 1997.
[31] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, ‘Multinomial Naive Bayes for Text Categorization Revisited’, *AI 2004: Advances in Artificial Intelligence*, pp. 488–499, Jan. 2005.
[32] A. McCallum and K. Nigam, ‘A comparison of event models for naive bayes text classification’, in *AAAI-98 workshop on learning for text categorization*, 1998, vol. 752, pp. 41–48.
[33] K. Nigam, J. Lafferty, and A. McCallum, ‘Using maximum entropy for text classification’, in *IJCAI-99 workshop on machine learning for information filtering*, 1999, vol. 1, pp. 61–67.
[34] S. Della Pietra and J. Lafferty, ‘Inducing features of random fields’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, 1997.
[35] V. N. Vapnik, ‘The Nature of Statistical Learning Theory’, 1995.

[36] C. Cortes and V. Vapnik, ‘Support-vector networks’, *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[37] C. Potts, ‘Sentiment Symposium Tutorial: Tokenizing’, Nov-2011. [Online]. Available: http://sentiment.christopherpotts.net/tokenizing.html. [Accessed: 31-May-2015].

[38] M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger, ‘The Penn Treebank’, Proceedings of the workshop on Human Language Technology - HLT ’94, 1994.

[39] C. Johnson, P. Shukla, and S. Shukla, ‘On Classifying the Political Sentiment of Tweets’, *cs. utexas. edu*, 2012.

[40] M. Wiegand, A. Balahur, B. Roth, D. Klakow, and A. Montoyo, ‘A survey on the role of negation in sentiment analysis’, in *Proceedings of the workshop on negation and speculation in natural language processing*, 2010, pp. 60–68.

[41] A. Esuli and F. Sebastiani, ‘SENTIWORDNET: A high-coverage lexical resource for opinion mining’, *Evaluation*, pp. 1–26, 2007.

[42] R. Sauri, ‘A factuality profiler for eventualities in text’, ProQuest, 2008.

[43] C. E. Osgood and M. M. Richards, ‘From Yang and Yin to and or but’, *Language*, vol. 49, no. 2, 1973.

[44] J. Boucher and C. E. Osgood, ‘The pollyanna hypothesis’, *Journal of Verbal Learning and Verbal Behavior*, vol. 8, no. 1, pp. 1–8, 1969.

[45] ‘SentiWordNet’, *SentiWordNet*. [Online]. Available: http://sentiwordnet.isti.cnr.it. [Accessed: 04-Sep-2014].

[46] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ‘Efficient estimation of word representations in vector space’, *arXiv preprint arXiv:1301.3781*, 2013.

[47] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, ‘A neural probabilistic language model’, *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[48] T. Mikolov, W. Yih, and G. Zweig, ‘Linguistic Regularities in Continuous Space Word Representations.’, in *HLT-NAACL*, 2013, pp. 746–751.

[49] F. Fallucchi and F. M. Zanzotto, ‘Transitivity in semantic relation learning’, *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering(NLPKE-2010)*, 2010.

[50] J. Mitchell and M. Lapata, ‘Composition in Distributional Models of Semantics’, *Cognitive Science*, vol. 34, no. 8, pp. 1388–1429, 2010.

[51] E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni, ‘Multi-step regression learning for compositional distributional semantics’, *arXiv preprint arXiv:1301.6939*, 2013.

[52] A. Yessenalina and C. Cardie, ‘Compositional matrix-space models for sentiment analysis’, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 172–182.

[53] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, ‘Parsing with compositional vector grammars’, in *Proceedings of the ACL conference*, 2013.

[54] Q. V. Le and T. Mikolov, ‘Distributed representations of sentences and documents’, *arXiv preprint arXiv:1405.4053*, 2014.

[55] B. Pang and L. Lee, *Opinion mining and sentiment analysis*. Hanover, MA: Now Publishers, 2008.

[56] K. Fragos, Y. Maistros, and C. Skourlas, ‘A Weighted Maximum Entropy Language Model for Text Classification.’, in *NLUCS*, 2005, pp. 55–67.