GALATASARAY UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

MULTICLASS ANALYSIS OF AUTOMATIC TEXT CLASSIFICATION TECHNIQUES

Semuel FRANKO

June 2018

MULTICLASS ANALYSIS OF AUTOMATIC TEXT CLASSIFICATION TECHNIQUES
(OTOMATİK METİN SINIFLANDIRMA TEKNİKLERİNİN ÇOK SINIFLI ANALİZİ)

by

SEMUEL FRANKO, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

June 2018

This is to certify that the thesis entitled

MULTICLASS ANALYSIS OF AUTOMATIC TEXT CLASSIFICATION TECHNIQUES

prepared by **SEMUEL FRANKO** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering at the Galatasaray University** is approved by the,

Examining Committee:

Assist. Prof. Dr. İsmail Burak PARLAK (Supervisor)
Department of Computer Engineering
Galatasaray University

Assist. Prof. Dr. Murat AKIN
Department of Computer Engineering
Galatasaray University

Assist. Prof. Dr. İlker ÜSTOĞLU
Department of Control and Automation Engineering
Yildiz Technical University

Date:

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor İ. Burak Parlak for the continuous guidance and efforts throughout my Master study and my thesis research. His guidance helped me in all phases of the research and writing of this thesis.

I would like to thank my parents for supporting me in every step of my life and making everything look easier.

I am also grateful to my professors and friends in Master's program for supporting me.

This research has been financially supported by the Galatasaray University Scientific Research Projects (BAP) under the grant agreement number 18.401.001.

June 2018

Semuel Franko

TABLE OF CONTENTS

LIST OF SYMBOLS

CV	: Cross Validation
HTML	: Hypertext Markup Language
JSON	: JavaScript Object Notation
NB	: Naive Bayes
NLP	: Natural Language Processing
NLTK	: Natural Language Toolkit
ROC	: Receiver Operating Characteristic
SEPLN	: Spanish Society for Natural Language Processing
SVM	: Support Vector Machines
TFIDF	: Term Frequency-Inverse Document Frequency
XML	: Extensible Markup Language

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Text classification and clustering; are one of the most popular areas of research in natural language processing applications. These areas offer different possibilities to the researchers for determining the metrics that can measure corpora dynamics in the automatic text analysis applications. It is observed that English-based systems developed for the text analysis applications were not studied extensively for Spanish, which is the second most spoken language. In particular, it seems that the number of studies on multi-class text classification is very small compared to English language. The purpose of this work is to develop classifiers with machine learning methods on a corpus that can be used for Spanish text classification and to perform comparative analysis over different parameters. It is also aimed to calculate the optimum performance effects by measuring the critical parameter values in the methods by applying sensitivity analysis. Spanish corpus was created by preparing a set of 10 different topics from texts of electronic newspapers and magazines. The indexing was achieved according to the topics where the pre-processing steps were completed for the machine learning methods. Naive Bayes, Decision Trees, Maximum Entropy and Decision Support Vector Machines are used. The basic parameters affecting the performance of the classifiers were examined and analyzed. The results of more than 1800 tests indicate that the methods can successfully classify the topics. Sensitivity analysis improves the accuracy of the classifier from 2% to 16%. The methods that yield the best performance have reached an accuracy of 89%, 88% and 87%, respectively. In addition to the accuracy, precision and recall of the test results, the computation time has been integrated to the analysis where the classifier models have been computed.

Keywords : natural language processing, machine learning, text classification, spanish language, sensitivity analysis

RÉSUMÉ

Classification de texte et regroupement; est l'un des domaines de recherche les plus populaires dans les applications de traitement du langage naturel. Ces deux domaines offrent des possibilités différentes pour les chercheurs, avec des métriques déterminées pour mesurer la dynamique des corpus dans les applications sous la rubrique de l'analyse automatique de texte. Le but de ce travail est de développer des classificateurs avec des méthodes d'apprentissage automatique sur une compilation qui peut être utilisée pour la classification des textes en espagnol et d'effectuer une analyse comparative sur différents paramètres. Il vise également à calculer les effets de performance optimaux en mesurant les valeurs de paramètres critiques dans les méthodes en appliquant une analyse de sensibilité. La compilation du corpus espagnol est effectuée en préparant une base de 10 sujets différents dans des journaux électroniques et des textes dans les magazines. Les étapes de prétraitement préliminaires, ils sont complétés pour l'apprentissage automatique en ayant indexé selon les sujets. Les méthodes suivants; Naive Bayes, les arbres de décision, l'entropie maximale et les machines vectorielles d'aide à la décision sont utilisés. Les paramètres de base affectant la performance des classificateurs ont été examinés et analysés sur des effets les plus influents. Les résultats de plus de 1800 tests indiquent que les méthodes peuvent classer le sujet avec succès. L'analyse de sensibilité a amélioré la précision du classificateur de 2% à 16%. Les méthodes qui donnent les meilleures performances ont atteint une précision de 89%, 88% et 87%, respectivement. En plus de la précision, de l'exactitude et de l'exactitude des résultats des tests, on a également examiné le temps de calcul pour le traitement requis à fin de préparer le modèle de classificateur et le classificateur optimal.

Mots Clés : traitement du langage naturel, apprentissage automatique, classification de texte, langue espagnol, analyse de sensibilité

ÖZET

Metin sınıflandırma ve kümeleme; doğal dil işleme uygulamaları içerisindeki en popüler araştırma alanlarındandır. Bu iki alan, otomatik metin analizi başlığındaki uygulamalarda derlem dinamiklerini ölçecek metriklerin belirlenmesinde araştırmacılara farklı olanaklar sunmaktadır. Metin analizi uygulamalarında geliştirilen İngilizce tabanlı sistemlerin en çok konuşulan ikinci dil olarak gösterilen İspanyolca için yeteri kadar incelenmediği gözlemlenmektedir. Özellikle, çok sınıflı metin sınıflandırması konusunda yapılan çalışma sayılarının İngilizce ile karşılaştırıldığında oldukça az olduğu görülmektedir. Bu çalışmanın amacı, İspanyolca metin sınıflandırma için kullanılabilecek bir derlem üzerinde, makine öğrenmesi yöntemleri ile sınıflandırıcılar geliştirmek ve farklı parametreler üzerinden karşılaştırmalı analizini gerçekleştirmektir. Bunun yanında duyarlılık analizi uygulanarak yöntemler içerisindeki kritik parametre değerleri ölçülerek optimum performans etkilerinin hesaplanması da amaçlanmıştır. 10 farklı konudan oluşan bir derlem hazırlanarak oluşturulan İspanyolca derlem içerisinde; elektronik gazete ve dergilerdeki metinler, gerekli ön işlem adımları uygulandıktan sonra konularına göre dizinlenerek makine öğrenmesi için hazırlanmıştır. Naive Bayes, Karar Ağaçları (Decision Trees), Maksimum Entropi ve Karar Destek Vektör Makineleri kullanılmıştır. Sınıflandırıcıların performansa etki eden temel parametreleri incelenmiş ve en çok etki edenler üzerinde analiz yapılmıştır. Yapılan 1800'den fazla testin sonuçları ilgili metotların başarıyla konu sınıflandırma yapabildiğini göstermektedir. Duyarlılık analizi sınıflandırıcının doğruluk değerinde %2 ile %16 arasında iyileşme sağlamaktadır. En iyi performansı veren metotlar konu tahmin konusunda %89, %88 ve %87 gibi bir doğruluğa ulaşmaktadır. Test sonuçları doğruluk, kesinlik ve anmanın yanında, sınıflandırıcı modeli hazırlanması için gerekli işlem süresi yönünden de incelenerek optimum sınıflandırıcı için yorum yapılmıştır.

Anahtar Kelimeler : doğal dil işleme, makine öğrenmesi, metin sınıflandırma, ispanyolca dili, duyarlılık analizi

1. INTRODUCTION

Over last decades, the digitalization of textual information has revealed several tasks related to the available online knowledge. The clustering, classification, knowledge extraction and mining which represent different steps of text processing need robust information models due to irregularities and non standard noise in digitized texts. In natural language processing (NLP), text classification is considered as a challenging task due to the complexity of knowledge representation. Text processing is highly correlated with the information characterized by the entities or the tokens represented by language models. These models are considered as generic that can be applied to natural languages.

In this thesis, text analysis refers to cutting edge classification of multiclass documents. The multiclass analysis points out the process of label assignment to a document. Documents arising from different sources like ; news reports, blogs, emails and other online documents would contain similar topics. In these cases, an appropriate label set should be initialized in order to form a correct extraction and management of knowledge representation.

In text classification, several methods are in use through the behavior of document nature. The collection stage of a corpus is the first step for text based studies. A relevant representation of the corpus is crucial in order to set a scalable and language independent algorithms. Thus, machine learning methods are preferred in highly flexible corpus to understand the essential features from the prior sets and to guess posterior knowledge carrying out the dynamic representation of the document classification (Srivastava and Sahami, 2009).

Multiclass document classification is a cutting-edge problem of machine learning where supervised techniques are applied to identify the target class among predefined categories to which the considered text will be assigned (Berry and Castellanos, 2008). There are two important stages of this procedure ; training and testing.

In this thesis, the Spanish corpus has been created by preparing a set of 10 different topics from texts of electronic newspapers and magazines. The observation has been initialized by the setup of relevant datasets by downloading and parsing HTML contents from online documents. After the download of HTML content from online newspaper and magazines, these documents have been parsed to extract text and category data. Furthermore, the documents have been filtered out in the preprocessing stage to analyze raw text data and to normalize them. (tokenization, removing stop words and non alphabetic characters, stemming). The language models such as document vector model or hashing functions have been defined to prepare classifiers. The feature selection has been performed to initialize the document classifiers. The ultimate part was the assignment and the evaluation process in order to reveal the performance of classification results. The aim of this thesis would be summarized as the benchmark analysis of popular machine learning techniques for a multiclass corpus. The optimized classifiers are designed for language models derived from digitized texts by using Python language and related toolboxes such natural language toolbox : NLTK and machine learning toolbox : Scikit. Naive Bayes, Decision Tree, Maximum Entropy and Support Vector Machines have been integrated to the language models. The optimized classifiers have been compared by the classification metrics like accuracy, precision, recall and F-score. During the development of the classifiers, sensitivity analysis has been achieved where the classifiers were examined for their most significant parameters that improve the performance. After multiple tests to obtain the best parameters the optimum classifiers have been set. Also, computation time scores have been recorded and discussed in the comparative analysis of the classifiers.

Chapter 1, presents a brief introduction about the study. Chapter 2 reviews existing studies from the literature. Chapter 3 is devoted to the background of the classifier methodologies that were used. Chapter 4 presents how to create digital documents, obtain a corpus and normalize the texts. Chapter 5 comprises the evaluation procedures which include language features for splitting the documents and evaluation metrics of the results. Chapter 6 covers the experiments and results for each type of classifier and a comparative analysis. Finally, Chapter 7 concludes this thesis by discussing the outcomes.

2. LITERATURE REVIEW

In this section, the survey about the recent studies has been performed from the viewpoint of text classification through Spanish language. Robust text classification is a challenging task for Internet based document analysis. Even if reliable routines are available for English language, automatic text classification is still a cutting edge in Spanish world wide web. (Wu et al., 2008) characterized SVM, Naive Bayes, Decision Trees and Maximum Entropy measures among top 10 data mining algorithms. Saez et al.studied the effects of different levels and types of noise in multi-class datasets. (Pla and Hurtado, 2017) observed multi-class problems for the identification of Iberian languages through social media texts. Multiclass text classification in English draws the challenge behind the heterogeneous effect in corpus through the conceptual match.

As one of these successful methods, Nave Bayes is popular in text classification due to its computational efficiency and relatively good predictive performance. In recent years, there are many studies about the Naive Bayes classifier applied in text classification ((Frank and Bouckaert, 2006) ; (Kim et al., 2006) ; (Lewis, 1998) ; (McCallum and Nigam, 1998) ; (Mladenic and Grobelnik, 1999) ;(Mladenić and Grobelnik, 2003)). For text classification, a major problem is the high dimensionality of the feature space. It is very often that a text domain has several tens of thousands of features. Most of these features are not relevant and beneficial for text classification task. Even some noise features may sharply reduce the classification accuracy. Furthermore, a high number of features can slow down the classification process or even make some classifiers inapplicable.

The analysis of social media especially Twitter messages becomes a popular tool to measure linguistic properties and to classify tweets to different topics (Vilares et al., 2015). In their feature models different approaches have been used such as, Unigrams and Bigrams of words, lemmas, Psychometric properties and Part of speech tags. For the classier, they have used Sequential Minimal Optimization technique which is a variant of

Support Vector Machines. Their results show that relating NLP-extracted features add complementary knowledge over pure lexical models. So it becomes possible to outperform them on standard classification metrics. Morevover, graph based techniques have been used to classify topics in Spanish tweets (Cordobés et al., 2014). In their study, they have proposed a system where very short text classification is possible by using vector classification model. The basic principle is that every piece of text can be represented as a graph. For each topic, a preclassified graph has been constructed. In case of new tweets, its graph is generated and compared with reference graphs where the following text attributes are used, PageRank, HITS, Graph Density and modifications. In both studies ((Cordobés et al., 2014), (Vilares et al., 2015)), TASS 2013 general corpus of tweets of SEPLN (Spanish Society for Natural Language Processing) has been used Moreover, the binary maximum entropy classifiers for sentiment analysis and topic classification of Spanish written tweets have been implemented by (Batista and Ribeiro, 2013). Their system gave better results on topic classification then the sentiment analysis. The maximum entropy classifier has been used for both approaches. In terms of polarity, six values, None, Negative and Negative Plus, Positive and Positive Plus, Neutral have been set. For the topic classification 10 different sets have been defined and for their best combination, test set gave 64.9% and 63.4% accuracy for topic classification and sentiment analysis respectively.

The multilingual studies reveal the performance of language models according to different parameters. The conventional classification approaches that have been proved to be effective for English text were also found effective in Spanish texts in the study of (Anta et al., 2013). In the same study, they have focused on the processing of Spanish tweets. Although they have used stemmers and lemmatizers, n-grams, word types, negations, valence shifters and different methods, none of them made a clear difference in their algorithms. Also, they have underlined the complexity of tweets due to their short structures and the lack of content. 58% accuracy for topic classification and 42% accuracy for sentiment analysis have been reached with the best parameter sets. The opinion classification has been studied by (Martínez-Cámara et al., 2011) used machine learning techniques. They have used the data of the website Muchocine which has summary and reviews of the movies. Opinions have been rated between 1 and 5. Support Vector Machines (SVR), Nave Bayes, Bayesian Logistic Regression (BRR), K Nearest Neighbors and C4.5 Decision Trees have been used. In the test step, best results have been obtained when both of the summary and reviews were integrated in the implementation.

Naive Bayes method is considered as a popular technique in NLP problems. (Escudero et al., 2000) has studied the word sense disambiguation problem where the appropriate meaning is chosen for a given word. They have used Naive Bayes and Exemplar-based classification techniques. Regarding final results, Naive Bayes with standard parameters had given the general accuracy of 67.1%. The exemplar-based classifier was modified to two variants. Example/attribute weighted version had the accuracy of 67.2%, Modified value difference matrix version had the accuracy of 68.6%. Moreover, (Gamallo et al., 2013) has studied sentiment analysis for the TASS 2013 corpus with 7216 Spanish tweets to detect the polarity. Although best performance was achieved when there are only two polarity categories like positive and negative, they have also studied additional polarities like strong positive, neutral, strong negative and no sentiment. It was shown that 4 polarity level system had 66% accuracy and 6 polarity level system had 55% accuracy. During the SemEval 2014 workshop, (Gamallo and Garcia, 2014) have presented their study for sentiment analysis for English tweets. That corpus had 6408 tweets and they were tagged with five different polarity levels, positive, neutral, negative, objective and neutral-or-objective. They have created 6 variants of Naive Bayes classifiers, where best one had the accuracy of 63%. (Juan and Ney, 2002) have implemented both types of Naive Bayes classifiers ; Bernoulli model and multinomial model. They have considered the effect of smoothing on the parameters and the normalization of document length which improved slightly the classifier performance. The multinomial model has been found as the successful technique regarding the empirical results. The default value of smoothing parameter gave the error rate of 15.4%. Best smoothing value gave the error rate of 14.9% and absolute discounting improved it to 14.6%. The multilabeled learning problem, where each document belongs to two or more predefined categories is another application area for Naive Bayes (Zhang et al., 2009). The methods have been tested with both synthetical and real-world (natural scene and gene functional classes) data sets. They have obtained highly competitive performance for classifying this high-dimensional data.

Decision trees is a common machine learning tool which is applied on several NLP routines. Optimized rule base induction methods have been studied by (Apté et al., 1994). Their aim was to discover automatically created pattern for text classification, rather than using systems that have human-engineered rules. Their results showed that these automated rules are comparable to human generated rules. Their algorithms were tested with Reuters collection and the performance was improved from 67% to 80.5% from a previous work.

The maximum entropy is characterized as another classification tool in machine learning. The classification of the email messages whether or not they have certain email acts has been studied by (Carvalho and Cohen, 2005). These acts can be a request, commitment, meeting etc. For each act, a maximum entropy classifier has been defined and has been compared through Kappa statistics. It is shown that for most of the acts they had statistically significant improvements with respect to baseline algorithm. (Fleischman et al., 2003) has built a statistically based semantic classifier and has analyzed it with a database called FrameNet, This structure includes semantically annotated sentences. The same structure has been extended with Maximum Entropy models and sentence-level syntactic patterns have been added into the feature set. Regarding the experiments, a statistically significant improvement has been obtained, 70.6% as F-score. The extraction of semantic relationships between entities has been studied by (Kambhatla, 2004). The paucity of the annotated data and the errors created by the model that is used for entity detection have caused new challenges in the problem. Maximum Entropy models have been employed. Competitive results have been obtained in the Automatic Content Extraction, which is an evaluation organized by National Institute of Standards and Technology. In a similar way, sentence extraction problem has been examined by (Osborne, 2002). In that study, Naive Bayes and Maximum Entropy classifiers were used. The latter gave marginally better results, where F2 score difference was 16%.

Finally, the support vectors machines become a standard in machine learning where there is big challenge in classifier separation in hyperspaces Text categorization on Reuters and Ohsumed corpora has been analyzed by (Joachims, 1998) studied. Among other classifiers, SVM has been identified as the best method. For the precision/recall-breakeven point it gave 86.4%, while the nearest classifier was k-NN which obtained 82.3%. Opinion mining can be defined as determining whether an opinion in a document is negative or positive. (Saleh et al., 2011) developed an SVM model for that problem. They have started to use Pang and Tobado corpora. Also they have created their own corpus called SINAI. Regarding the experiments, trigram gave generally better results. For Pang corpus, the accuracy of binary occurrences was about 84.9%. For Topado and SINAI, the performance of TFIDF models has been found better with the accuracy values of 73% and 91% respectively. They have concluded that this result can be affected by the domain of their corpus, where review comments are easily identifiable.

In a similar way, (Wang and Manning, 2012) examined subjectivity dataset; MPQA, IMDB and RT-2k datasets for sentiment and topic classification. In their study, it is mentioned that, although Naive Bayes and SVM used as baseline methods, their performance have changed due to the model variance using features and datasets. Including bigrams in feature set constantly increases the performance of sentiment analysis. For short snippets Naive Bayes performs better, however, SVM outperforms in long texts. They have also proposed another method called NBSVM (SVM with Naive Bayes features). Both for snippets and for long documents NBSVM seemed to be a strong baseline.

3. METHODOLOGY

In this section main areas of text classification will be introduced. Machine learning techniques are classified through the use of language models and the features. The methods that are used in the experiments are detailed with their brief characteristics in NLP.

3.1 Text Classification

The aim of the text classification is assigning documents to one or more categories. Documents might be long as articles, news, blog posts or short as tweets, product reviews and comments. Categories can be binary like spam-not spam, fraud-not fraud. Besides, news stories can be tagged by topics like politics, general culture, sports etc. Movie reviews can be assigned to categories like comedy, drama etc. The last two examples are categorized as multiclass text classification which is examined during this thesis. Other main topics of text classification are sentiment analysis, text summarization and author detection. Recently, machine learning techniques gained a momentum in this process. In this chapter, four different methods will be discussed as learning machines in NLP domain.

3.2 Learning Machines

Learning machines studies how to automatically learn from data, whether than explicitly programming. In a broad perspective machine learning tasks can be divided into : Supervised learning, Unsupervised learning, semi-supervised learning and reinforcement learning.

— Supervised learning : During the training phase, correct-input output pairs are available.

— Semi-supervised learning : Supervised and unsupervised techniques are used together.

— Unsupervised learning : During training phase, correct answers does not exist. These techniques find patterns. One of the most used application is clustering.

In this study the following supervised classification techniques have been studied : Naïve Bayes, Decision Trees, Maximum Entropy, Support Vector Machines.

3.3 Naïve Bayes

Naive Bayes algorithm uses Bayes Theorem to predict the probability of a label for a given feature set. In Bayes theorem equation 3.1 is used :

$$P(label \mid features) = \frac{P(label) \cdot P(features \mid label)}{P(features)} \tag{3.1}$$

In the formula, $P(label)$ shows the prior probability of the label in documents. Also $P(features \mid label)$ shows the probability of a given feature to be classified as that label, which is also known as likelihood. This is based on the feature and label matches in training data. $P(features)$ shows the probability of a feature to exist in training data. It can be predicted the probability of given features with $P(label \mid features)$ that should have that label.

Given a class variable y and a dependent feature vector $(x_1,...,x_n)$. Regarding to Bayes theorem the relationship in 3.2 exists(Zhang, 2004) :

$$P(y|x_1,...,x_n) = \frac{P(y)P(x_1,...,x_n|y)}{P(x_1,...,x_n)} \tag{3.2}$$

In this algorithm naive aspect arises from the assumption of independence between every pair of futures. Naive independence assumption can be seen in 3.3 for all i.

$$P(x_1, x_2, ..., x_n|y) = \prod_{i=1}^{n} P(x_i|y) \tag{3.3}$$

The relationship in 3.2 can be simplified to 3.4

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,...,x_n)} \tag{3.4}$$

Because of the $P(x_1,...,x_n)$ is constant given the input, the classification rule in 3.5 can be used.

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \tag{3.5}$$

$P(y)$ and $P(x_i|y)$ can be estimated by maximum a posteriori estimation.

$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i|y) \tag{3.6}$$

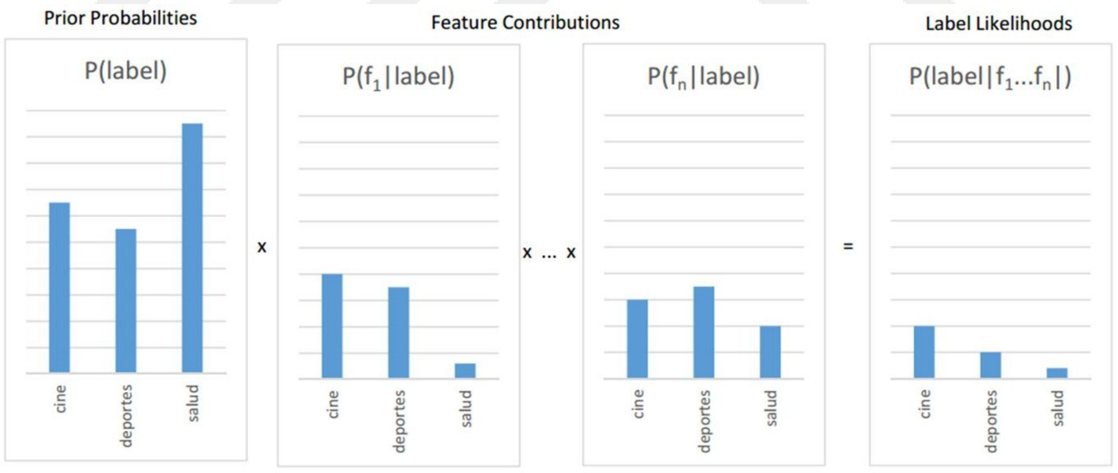

Figure 3.1: Naïve Bayes probability contributions.

Mainly there exist three variants of Naive Bayes. Bernoulli Naive Bayes assumes that all features are binary. Multinomial Naive Bayes is used when data is discrete. Gaussian Naive Bayes is used when all features are continuous. In this thesis, Multinomial Naive Bayes has been implemented.

In Multinomial Naive Bayes, the distribution is parametrized by vectors $\theta_y = (\theta_{y1},...,\theta_{yn})$ for each class y, where n shows the number of features, where in text classification it signifies the size of vocabulary and θ_{yi} is the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class y (Manning et al., 2008).

The parameters θ_y is estimated by the smoothed version of maximum likelihood.

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{3.7}$$

In 3.7 $N_{yi} = \sum_{x \in T} x_i$ shows the feature count of i in a sample of class y in training set T. $N_y = \sum_{i=1}^{|T|} N_{yi}$ is the total count of all features for class y.

In order to improve the performance of Naive Bayes classifier α parameter can be optimized. A word that does not appear in training data, but can be in use within the test data. This case results in zero probabilities. In order to handle this problem, additional smoothing is usually preferred. Additive smoothing, so called Laplace smoothing or Lidstone smoothing, is a technique used to smooth categorical data. The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Laplace smoothing indicates that $\alpha = 1$. If $\alpha < 1$ it is called Lidstone smoothing.

Naive Bayes algorithm could give relevant results in real applications where the computation time is lower and the design of language model becomes simpler. In a nutshell, it is scalable to train big data, efficient for fast training and requires modest storage. On the other hand, it ignores the word order due to the bag of words model and would not be suitable if the dependency between the variables is strong.

Main advantages :

— Very easy to implement

— Can easily scale to a big training data

— Very efficient and fast training phase

— Modest storage need

Main disadvantages are :

— Order of the words are ignored, due to the bag of words model

— Words are not independent of each other. Mountain is more likely to occur in a context of nature, rather than health

— Naive Bayes algorithm is not suitable if there is a strong dependency between the variables

3.4 Decision Trees

Decision tree is a flowchart that selects labels for feature values. The goal is to create a classifier that predicts the label by learning simple decision rules from the training data. Let's assume that we have following rules in Table 3.1 to decide the category of the text.

Table 3.1: Decision tree rule chart.

	Pelicula	Libro	Film	Album	Deporte	Gira	Category
Rule 1	true	true	true	-	-	-	Cine
Rule 2	true	true	false	-	-	-	Literaria
Rule 3	true	false	-	true	-	-	Musica
Rule 4	true	false	-	false	-	-	Tecnologia
Rule 5	false	-	-	-	true	true	Musica
Rule 6	false	-	-	-	true	false	Tecnologia
Rule 7	false	-	-	-	false	-	Tecnologia

A flow chart is constructed by using these rules. In that chart, decision nodes check feature values and leaf nodes assign labels. So for the features, decision trees are created.

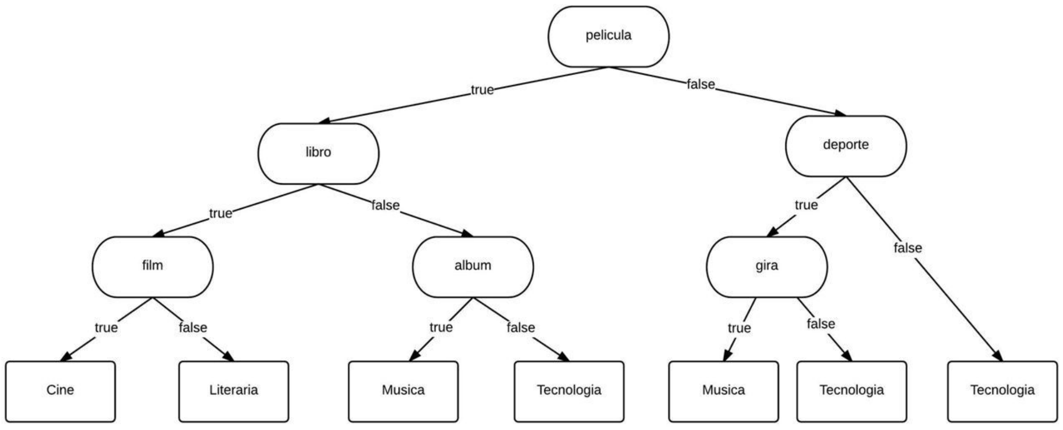

Figure 3.2: Decision tree flow chart.

While constructing decision trees, we should take into consideration the over fitting. If the tree depth becomes more than we need, the over fitting might occur. Although the results seem relevant for the train sets, prediction capabilities of the classifier would be worse. Let's assume that we need to create a curve for our data points. If we increase the degree of our computations over fitting can occur as seen in Figure 3.3.

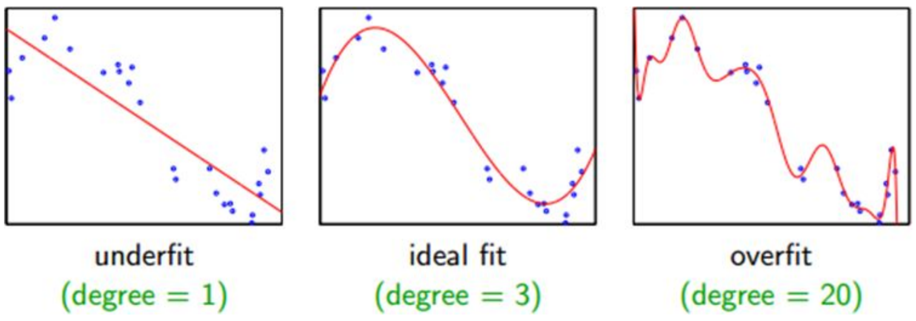

Figure 3.3: Underfit and overfit for data points.

For the construction of decision trees, a top-down approach is generally used. At each step, a variable is selected that best splits the set of items (Rokach and Maimon, 2005). Different algorithms utilize different metric functions to find the best. These algorithms generally measure the homogeneity of the target variable in the subsets. CART (Classification and Regression Trees) use Gini split algorithm and Gini index as the criterion. ID3 (Iterative Dichotomiser 3) type trees use Information gain algorithm uses Entropy criterion.

Gini impurity measures the probability of a randomly chosen element from the set would be erroneously labeled if it was randomly labeled according to the distribution of labels in the subset. It can be calculated by summing the square of probability p_i of an item with label i. In a data set of n classes, suppose $i \in \{1,2,...,n\}$ and let p_i be the fraction of items labeled with class i in the set (Gron, 2017).

$$Gini = \sum_{i=1}^{n} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{n} p_i(1-p_i) \tag{3.8}$$

$$= \sum_{i=1}^{n} (p_i - p_i^2) = \sum_{i=1} p_i - \sum_{i=1}^{n} p_i^2 \tag{3.9}$$

$$= 1 - \sum_{i=1}^{n} (p_i^2) \tag{3.10}$$

Gini impurity is zero when all of the elements of data are from the same class, due to the selecting an element of that class has the probability of one. Gini impurity value is greatest when each class has an equal probability. The maximum value of Gini impurity depends to the total number of classes (Garreta et al., 2017).

$$Gini_{max} = 1 - \frac{1}{n} \tag{3.11}$$

Main advantages of decision trees :

- Simple to understand the structure, because it can easily be visualized
- It can handle both continuous and categorical values
- To train the tree cost is logarithmic
- It is suitable for multi output problems

Main disadvantages are :

- Over fitting can occur, on design process we should make pruning to prevent over fitting
- Results can be unstable, because a small change in the inputs can create a different result
- If some classes dominate the training data, the decision tree can create biased trees

In order to optimize decision trees classifier, the maximum depth parameter was used which is the most significant parameter that changes the performance. Although, it is not possible to set the depth of the tree, maximum depth can be set.

3.5 Maximum Entropy

Maximum entropy classifier is another robust tool in learning machines for text classification. It is likewise a probabilistic classifier whose output is a probability distribution. However, the features are not considered conditionally independent from each other.

The algorithm is based on the principle of maximum entropy. The procedure is the selection of models that fit the training data with the largest entropy. Due to the complex interactions between the effects of related features, it is not possible to calculate model parameters directly. Thus, this algorithm calculates them by using iterative optimization techniques.

As an optimization problem, cost function given in equation 3.12 should be minimized where L2 penalization is used.

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} log(1 + e^{-y_i w^T x_i}) \tag{3.12}$$

where, w, x, y and c represent weights, inputs, outputs and regularization parameter respectively. In order to solve this cost function Liblinear solver was used which is designed for large-scale classification problems (Fan et al., 2008). It uses coordinate descent (CD) algorithm. However, CD algorithm in liblinear cannot solve a multinomial classifier. This optimization problem is decomposed in a "one-vs-rest" method, where for each class a classifier is developed (learn developers, 2017).

Main advantages are :

— Estimates probability distribution from data

— Performs well with dependent features

Main disadvantages are :

— Feature selection could be a complex procedure

— Computations can be slow

Maximum entropy performs better with dependent features. However, feature selection would be complex and computation time is quite slower. In order to improve the model, the inverse of regularization parameter (C) can be optimized. Smaller values specify stronger regularization where a penalty is applied to increase the magnitude of parameter values.

In a logistic regression model, the best fit to the data depicts the error minimization between the predicted and the actual dependent variable. When the number of parameters increases in a limited corpus, the error minimization problem is solved with the following function that penalizes larger values. The function is $\lambda \sum \theta_j^2$, which is some constant λ times the sum of the squared parameter values θ_j^2. The larger λ is the less likely it is that the parameters will be increased in magnitude simply to adjust for small perturbations in the data. However, $C = \frac{1}{\lambda}$ is also used rather than λ. The regularization parameter λ controls how well the training data fits while keeping the weights small.

3.6 Support Vector Machines

Support Vector Machine (SVM) algorithms construct one or more hyperplanes for the classification and the regression of data. In order to obtain optimum performance, hyperplanes are created regarding to the nearest training data of any class. Increasing the margin between these data points and hyperplane creates a smaller error for classification. In other words, SVM algorithms try to find maximum margin hyperplanes as seen in Figure 3.4.

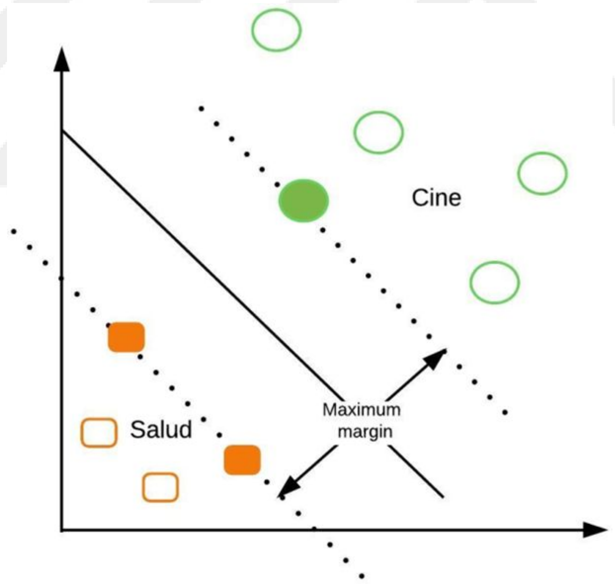

Figure 3.4: SVM margin.

For training vectors of $x_i \in \mathbb{R}^p$, $i = 1, ..., n$ and $y_i \in \{-1, +1\}$, Support Vector Classifier (SVC) solves primal problem in 3.13 and 3.14. Both equations are L2-regularized, but former has L1-loss latter has L2-loss function.

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (max(0, 1 - y_i w^T x_i)) \tag{3.13}$$

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (max(0, 1 - y_i w^T x_i))^2 \tag{3.14}$$

Their dual forms in 3.15.

$$\min_{\alpha} \frac{1}{2} \alpha^T \bar{Q} \alpha - e^T \alpha \tag{3.15}$$

subject to $0 \leq \alpha_i \leq U, i = 1, ..., n$. Where e is the vector of all ones, $\bar{Q} = Q + D$, D is a diagonal matrix, and $Q_{ij} = y_i y_j x_i^T x_j$. For L1-loss SVC, $U = C$ and $D_{ii} = 0$, $\forall i$. For L2-loss SVC, $U = \infty$ and $D_{ii} = \frac{1}{2C}$, $\forall i$.

Hence decision function is (Fan et al., 2008) :

$$\text{sgn}(w^T x + b) \tag{3.16}$$

Main advantages of SVMs :

— Gives an effective solution for high dimensional spaces

— It uses a subset of training data in the decision function (support vectors) so uses less memory

— For the decision function, different Kernel functions can be specified

— Over fitting is less common, robust to noise

— Works well with fewer training samples (number of vectors doesn't matter much)

Main disadvantages are :

— Selecting appropriate kernel function can be tricky

— Computationally expensive, calculations can take long time

To optimize SVM classifier one of the most significant parameters is called C. C is a regularization parameter which determines how much misclassification will be allowed in the result. For large values of C, the optimization method will choose a small margin hyperplane, so it can correctly predict most training points as possible. But for small values of C optimizer will create a large margin hyperplane, in that case, more points can be misclassified.

4. CORPUS ARCHITECTURE

In NLP, corpus is a large collection of the structured set of texts. A corpus can comprise texts from a single language or multiple languages. Its plural form is called corpora. Some best known text corpora are Gutenberg Corpus, which contains free electronic books, Brown Corpus which contains more than one million words and created in 1961 at Brown University. Also, Reuters corpus is very popular among NLP studies. It includes more than ten thousand news documents which are classified into 90 topics. A recent corpus is Google Books ngram corpus.

For the Spanish language some corpora are publicly available. WikiCorpus which is a trilingual corpus that contains articles from Wikipedia in Spanish. Europarl corpus includes European Parliament Proceedings and primarily created for statistical translation systems. Also, news commentary corpus exists which is created for the statistical machine translation and includes 12 languages including Spanish. Spanish Society for Natural Language Processing (SEPLN) organizes workshops and publishes every year a new corpus for researchers to analyze, but mainly its purpose is sentiment analysis. The main goal of this thesis is the topic classification in Spanish articles. Due to the absence of such a corpus, it was created by the author of this thesis.

4.1 Content of Spanish Corpus

The content of this study is taken from the electronic newspapers and magazines that publish articles in Spanish language. These articles are categorized in classes like cinema, music, technology, health etc. Most of the articles are medium to long size. For the corpus, 2576 different articles were downloaded and they were annotated to 10 different categories. , The prepared corpus is mostly homogeneous due to their categories. The total number of tokens is 608k. Articles do not overlap with each other, each article has only one category. The overall distribution of our corpus is detailed in Table 4.1.

Table 4.1: Number of articles by category.

Category	Number of Articles
A-donde-vamos (where to go)	287
Cine (cinema)	231
Deportes (sports)	216
Economia (economy)	205
Estilo (style)	319
Medio-ambiente (nature)	337
Musica (music)	124
Recomendacion-literaria (literature)	272
Salud (health)	282
Tecnologia (technology)	303
Total	2576

4.2 Processing of Spanish Corpus

Although the web site content was freely available, it was not always possible to reach all articles from a database with an XML (Extensible Markup Language) feed. Firstly, the downloaded articles were lexically parsed by using Beautiful Soup library version 4.5.1 with Python language. Beautiful Soup is a library that parses HTML (Hypertext Markup Language) and XML files. It creates a tree representation of the document and also provides methods to navigate, search and modify the tree. After the parsing process, a JSON (JavaScript Object Notation) file was created and saved to data folder for each category.

The second stage of corpus processing was the normalization. In this stage, the raw document was firstly converted to tokens, upper cases, non alphanumeric attributes. The stop words were removed, stems were obtained. This process was done for all texts of the corpus.

- Total documents : 2576
- Total number of tokens : approx. 608000
- Average number of tokens in a document : 236
- Average character length of a token : 5.4

The raw data in NLP applications should be preprocessed to be handled efficiently by machine learning algorithms. This process can significantly improve results. In this section, a text portion was processed by using NLP techniques.

Let's assume that we have following raw text :

El asombro ante lo que el hombre puede pensar y avanzar tecnológicamente, permite que la imaginación vaya aún más lejos de lo posible. La ficción en la literatura siempre ha presentado infinitas posibilidades de realidades alternas. Diferentes autores se han destacado por ser capaces de crear mundos con sus propias reglas, personajes, hasta naturaleza y tiempo nuevos. Los más talentosos son los que pueden contar una historia y hacer que los lectores se conviertan en parte de ese universo. Pero, ¿qué es la ciencia ficción ? En pocas palabras, es una manera de llamarle a un modo imaginativo, en ocasiones futurista, que existe a partir de los avances tecnológicos que se presentaron a principios del siglo XX.

The text content was split to small chunks and tokenized by using Natural Language Toolkit (NLTK). So the following tokens were obtained. Note that dot, comma and question marks are also identified as a token.

El, asombro, ante, lo, que, el, hombre, puede, pensar, y, avanzar, tecnológicamente, ,, permite, que, la, imaginación, vaya, aún, más, lejos, de, lo, posible, ., La, ficción, en, la, literatura, siempre, ha, presentado, infinitas, posibilidades, de, realidades, alternas, ., Diferentes, autores, se, han, destacado, por, ser, capaces, de, crear, mundos, con, sus, propias, reglas, ,, personajes, ,, hasta, naturaleza, y, tiempo, nuevos, ., Los, más, talentosos, son, los, que, pueden, contar, una, historia, y, hacer, que, los, lectores, se, conviertan, en, parte, de, ese, universo, ., Pero, ,, ¿qué, es, la, ciencia, ficción, ?, En, pocas, palabras, ,, es, una, manera, de, llamarle, a, un, modo, imaginativo, ,, en, ocasiones, futurista, ,, que, existe, a, partir, de, los, avances, tecnológicos, que, se, presentaron, a, principios, del, siglo, XX, ..

Punctuation marks, non-alphanumeric content were removed and text content was converted to lowercase.

el, asombro, ante, lo, que, el, hombre, puede, pensar, y, avanzar, tecnológicamente, permite, que, la, imaginación, vaya, aún, más, lejos, de, lo, posible, la, ficción, en, la, literatura, siempre, ha, presentado, infinitas, posibilidades, de, realidades, alternas, diferentes, autores, se, han, destacado, por, ser, capaces, de, crear, mundos, con, sus, propias, reglas, personajes, hasta, naturaleza, y, tiempo, nuevos, los, más, talentosos, son, los, que, pueden, contar, una, historia, y, hacer, que, los, lectores, se, conviertan, en, parte, de, ese, universo, pero, es, la, ciencia, ficción, en, pocas, palabras, es, una, manera, de, llamarle, a, un, modo, imaginativo, en, ocasiones, futurista, que, existe, a, partir, de, los, avances, tecnológicos, que, se, presentaron, a, principios, del, siglo, xx

Next step is removing stop words. Stop words are language specific words that are used extensively in texts. However, they do not have great impact on text classification. As an example some stop words for Spanish are : "el, lo, en, que". After the removal of Spanish stopwords, resulting tokens were obtained as follows :

asombro, hombre, puede, pensar, avanzar, tecnológicamente, permite, imaginación, vaya, aún, lejos, posible, ficción, literatura, siempre, presentado, infinitas, posibilidades, realidades, alternas, diferentes, autores, destacado, ser, capaces, crear, mundos, propias, reglas, personajes, naturaleza, tiempo, nuevos, talentosos, pueden, contar, historia, hacer, lectores, conviertan, parte, universo, ciencia, ficción, pocas, palabras, manera, llamarle, modo, imaginativo, ocasiones, futurista, existe, partir, avances, tecnológicos, presentaron, principios, siglo, xx

Stemmers are the tools that are frequently used to reduce inflected words to their stems or to their root form. For example *pienso*, *piensas*, *pensamos* can symbolize same verb *pensar* which has a stem of *pens*. Although Porter stemmer is a convenient stemmer for English, a suitable stemmer for Romance language is Snowball stemmer. By using Snowball's algorithm 15 languages including English, French, German, Spanish, Italian etc. can be processed efficiently.

After the removal of stop words, Snowball stemmer was used and the final tokens became as follows. Note that both stems were obtained and accent marks were removed from tokens.

asombr, hombr, pued, pens, avanz, tecnolog, permit, imagin, vay, aun, lej, posibl, ficcion, literatur, siempr, present, infinit, posibil, realidad, altern, diferent, autor, destac, ser, capac, cre, mund, propi, regl, personaj, naturalez, tiemp, nuev, talent, pued, cont, histori, hac, lector, conviert, part, univers, cienci, ficcion, poc, palabr, maner, llam, mod, imagin, ocasion, futur, exist, part, avanc, tecnolog, present, principi, sigl, xx

Thus, the result of the normalization process can be seen from the Figure 4.1. Blue parts indicate the change or the removal from the raw text after the normalization.

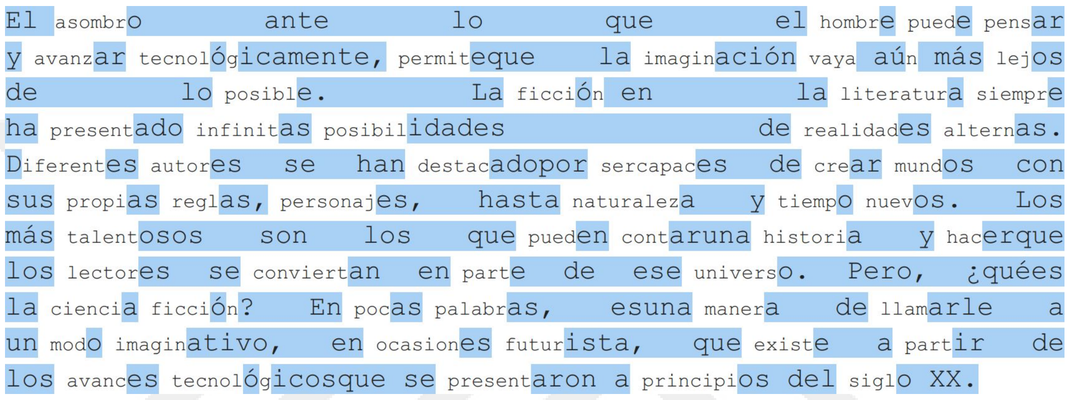
El asombro ante lo que el hombre puede pensar
y avanzar tecnológicamente, permiteque la imaginación vaya aún más lejos
de lo posible. La ficción en la literatura siempre
ha presentado infinitas posibilidades de realidades alternas.
Diferentes autores se han destacadopor sercapaces de crear mundos con
sus propias reglas, personajes, hasta naturaleza y tiempo nuevos. Los
más talentosos son los que pueden contaruna historia y hacerque
los lectores se conviertan en parte de ese universo. Pero, ¿quées
la ciencia ficción? En pocas palabras, esuna manera de llamarle a
un modo imaginativo, en ocasiones futurista, que existe a partir de
los avances tecnológicosque se presentaron a principios del siglo XX.

Figure 4.1: Difference of the normalization process.

During the normalization process, it is noted that there exist words with the length of one and two characters. Although one character tokens are not meaningful for classification process, two character words aren't removed from the corpus. Because some of the most used verbs in Spanish has stemmed tokens like : *da* (to give), *ir* (to go), *vi* (to see).

4.3 Implementation and Requirement for Software Technologies

In this study, Python programming language has been preferred for NLP and Machine Learning steps. The main Python libraries used in this project are enlisted as follows;

- Python's HTML (Hypertext Markup Language) downloader and Beautiful Soup for XML (Extensible Markup Language) and JSON (JavaScript Object Notation) parsing.
- Natural Language Toolkit (NLTK) for preprocessing of the documents.
- Scikit-learn for the development of machine learning classifiers and analysis.

— Matplotlib for the visualization.

— NumPy and SciPy for scientific computing.

— Pandas for data manipulation and analysis.

— Multiprocessing libraries of Python.

5. EVALUATION PROCEDURE

Natural Language processing is a field of artificial intelligence that provides computers to analyze and understand the language of humans. The development of NLP application is a challenging task because computers need data in a structured, precise and unambiguous way in a traditional manner. Generally, human texts might be ambiguous, including slang and regional words.

NLP comprises different techniques to handle texts and convert them in a structured way. Some of the methods are as follows :

— Tokenization : Task of chopping document or text into pieces called tokens.

— Stop words : Extremely used words that have a little value on classifying texts. These words are language specific.

— Stemming : Due to grammatical rules the same word can be found with different forms. Those forms are reduced to one form by removing postfixes.

— Lemmatization : Inflected forms of a word are grouped together and presented with one word.

5.1 Feature Extraction

In order to make Spanish corpus more suitable for computer algorithms, tokenized and normalized text should be encoded as integers or floating point values. They should be presented as feature vectors. This process is known as feature extraction or vectorization. The common vectorizers are count, hash and tf-idf methods.

— Count vectorizer : It is known as the bag of words. It focuses on the number of occurrences of the word in a document.

— Hash vectorizer : Token occurrences are used for the efficiency of memory usage. Tokens are not stored as strings, hashing trick is applied and encoded as numerical indexes. It is fast to pickle or unpickle them

— Tf-idf (Term frequency-inverse document frequency) vectorizer : Weights in feature sets are taken into consideration. Both number of occurrences in document and how recurrent that token in entire corpus are combined to calculate the final value.

Although, it is common to use one word tokens in document vectors, adding n-grams might increase model performance. N-grams are the contiguous sequence of n tokens that exist in a document. When *n* is set to 1 it can be referred as unigram. 2 and 3 are called as bigrams and trigrams, respectively. In the experiments of this thesis, all these three models were used.

Feature sets include basic information about each input. Choosing correct features has a big effect on classification process. Features should be selected regarding the nature of the problem. Obviously, the features of a text are different than other medias such as sound or image. Then, it is common to use the kitchen sink approach where all features are added to the classifier. The unneeded features with small effects on output could be discarded by using trial and error steps

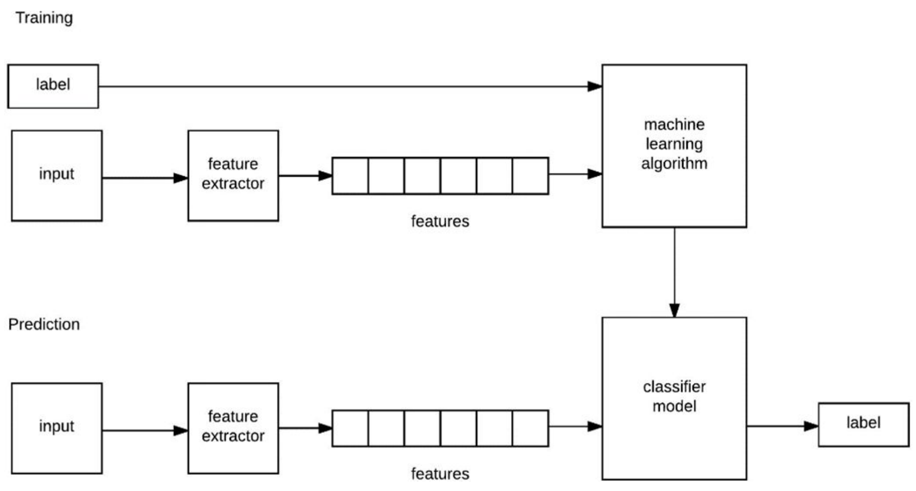

Figure 5.1: Evaluation procedure work flow.

5.2 Design of Datasets

After the preparation of feature sets, these sets should be sampled according to the corpus. The total set should be divided into train and test sets. The classifier is designed by train sets and further tests should be done with test sets.

The idea of the sampling approach is summarized as the prevention of overfitting. If the classifier is designed and tested with same data, results will be very good. However, the accuracy would be much lower and results will not be robust in future uses of the classifier.

Recommended values for the train and test percentages differ from the problem type and the size of the corpus. The common distribution of machine learning values are 80 to 20 or 70 to 30 for train to test data respectively. In this study train and test data are divided into 80 to 20 ratio respectively.

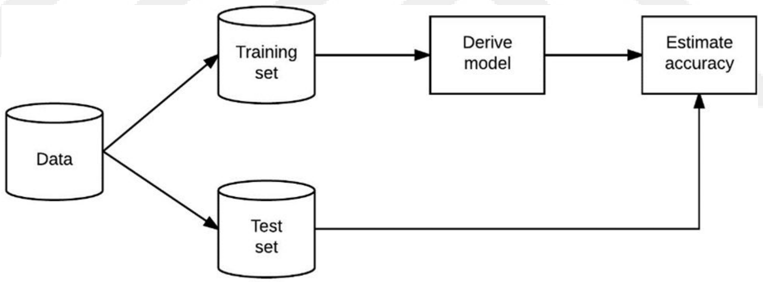

Figure 5.2: Train and test sets for evaluation.

5.3 Cross Validation and Evaluation

To minimize the bias associated with the random sampling of the train and test splits, a proper methodology is using k-fold cross validation. In this technique, corpus is randomly split into k subsets of approximately equal size. The developed classifier is trained and then tested k times. In each iteration, training is done on all but one fold and then tested on the remaining single fold. After all tests, final performance is calculated by the averaging result of each iteration. The classifier is trained and tested with the whole corpus so results will be more reliable with this approach. During this thesis 2, 5 and 10 K-folds cross validation was tested. A 5-fold cross validation test structure can be seen in Figure 5.3.

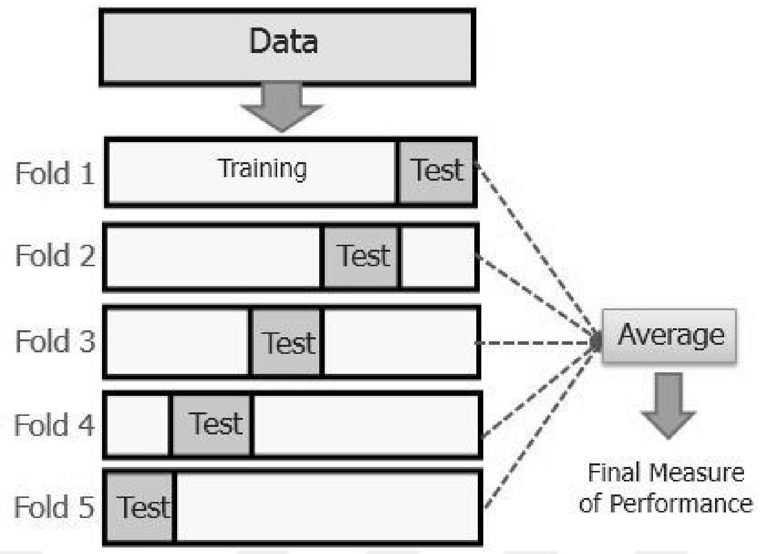

Figure 5.3: Cross validation test structure.

In order to understand the performance of the classifier, test procedures should be evaluated. Most techniques use test data to evaluate possible scores. The most used parameters are accuracy, precision, recall, f1-score and confusion matrices.

Accuracy is the simplest and most widely used metric in the evaluation. It uses the percentage of the predicted labels to test labels. It can be calculated by using number of true positives ($T_{positive}$), true negatives ($T_{negative}$), false positives ($F_{positive}$) and false negatives ($F_{negative}$)

$$Accuracy = \frac{T_{positive} + T_{negative}}{T_{positive} + T_{negative} + F_{positive} + F_{negative}} \tag{5.1}$$

Precision indicates how many of the items that we identified are relevant. It is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p).

$$Precision = \frac{T_{Positive}}{T_{positive} + F_{positive}} \tag{5.2}$$

Recall shows how many of the relevant documents are identified. It can be defined as the ratio of the number of the true positive documents (T_p) to the true positives and the false negative documents (F_n). The recall is also known as sensitivity or True Positive Rate (TPR).

$$Recall = \frac{T_{Positive}}{T_{positive} + F_{negative}} \tag{5.3}$$

F_1 score combines precision and recall values. It is defined as the harmonic mean of that two values.

$$F_1 = 2(\frac{Precision \cdot Recall}{Precision + Recall}) \tag{5.4}$$

False Positive Rate (FPR) is also called as false alarm rate. It shows the probability of falsely rejecting the null hypothesis for the test.

$$FPR = \frac{F_{positive}}{F_{positive} + T_{negative}} \tag{5.5}$$

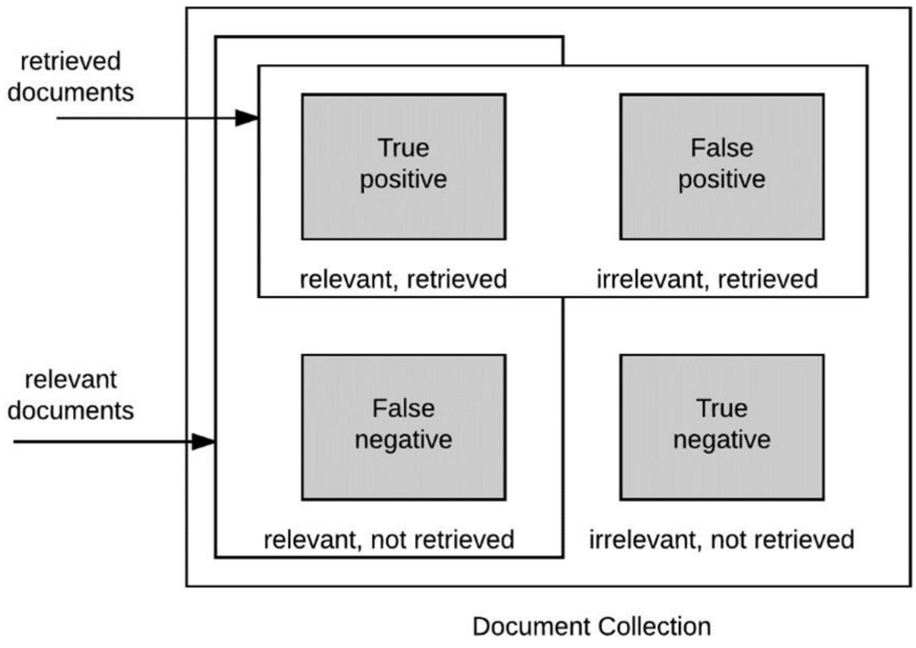

Figure 5.4: Retrieved and related documents in collection.

5.4 Precision-Recall Curve

Precision-Recall is a standard measure to evaluate and compare the performance of one or more classifiers. It is more advantageous when classes are imbalanced(Gron, 2017). Y-axis signifies precision, X-axis shows recall. It shows the trade off between the precision and recall of the classifier. A big area under the curve depicts high precision and recall, which signifies classifier both returns accurate results and returns majority of all positive results. A high recall and low precision classifier can get many results but most of its predictions are incorrect. A high precision but low recall classifier returns very few numbers of results but most of its predictions are correct. An ideal classifier has high precision and recall which signifies that the system returns many results that are predicted correctly. A typical precision-recall curve for two different algorithms can be seen in Figure 5.5.

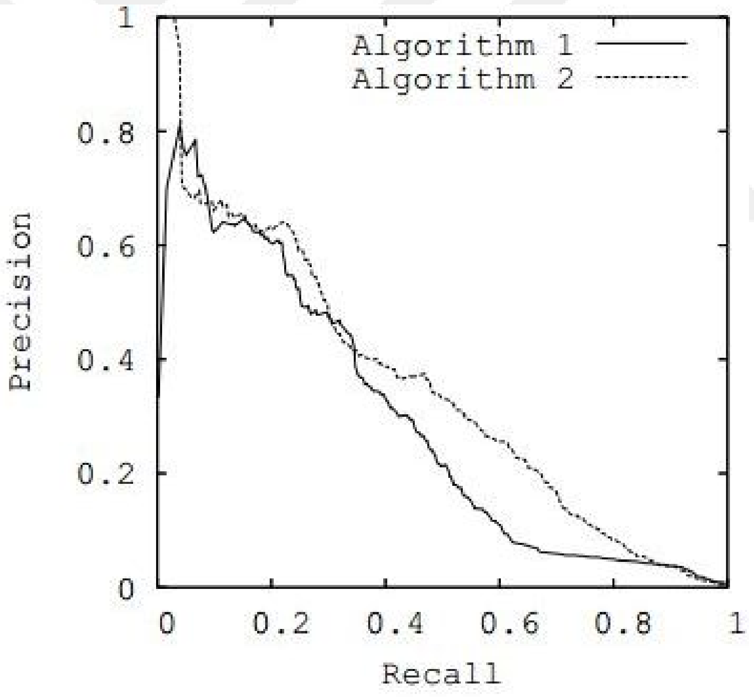

Figure 5.5: Precision-recall curves for two algorithms.

5.5 ROC Curve

Receiver Operating Characteristic (ROC) curve is used to evaluate classifier output quality. Y-axis signifies True Positive Rate (recall) and X-axis shows False Positive Rate. In this plot, ideal point is on upper left where the true positive rate is one and the false positive rate is zero. It is preferable to have a large area under the curve. Also, it is possible to plot ROC curves to compare the performance of multiple classifiers as seen in Figure 5.6.

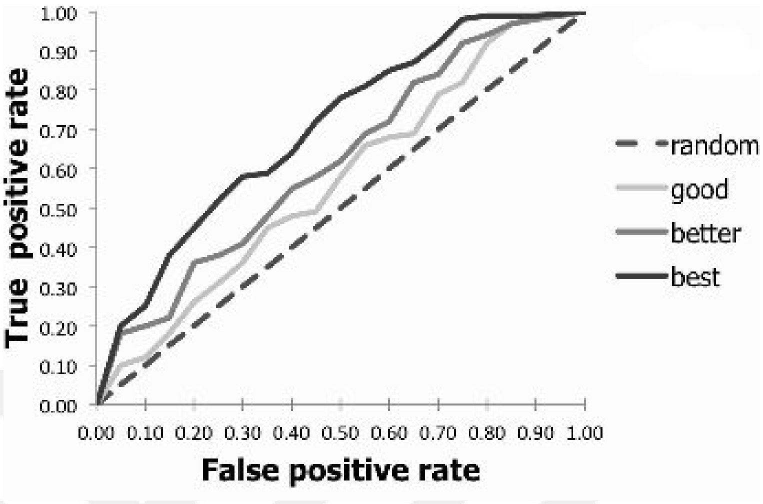

Figure 5.6: ROC curves for multiple algorithms.

5.6 Confusion Matrix

The confusion matrix is a specific table that shows the performance of an algorithm, generally a supervised learning algorithm. Each row of this matrix shows instances of an actual class and each column represents instances of the predicted class. Diagonal values show the elements where predicted value is equal to expected value. Off-diagonal values represent where the prediction is wrong. A typical confusion matrix can be seen in Figure 5.7.

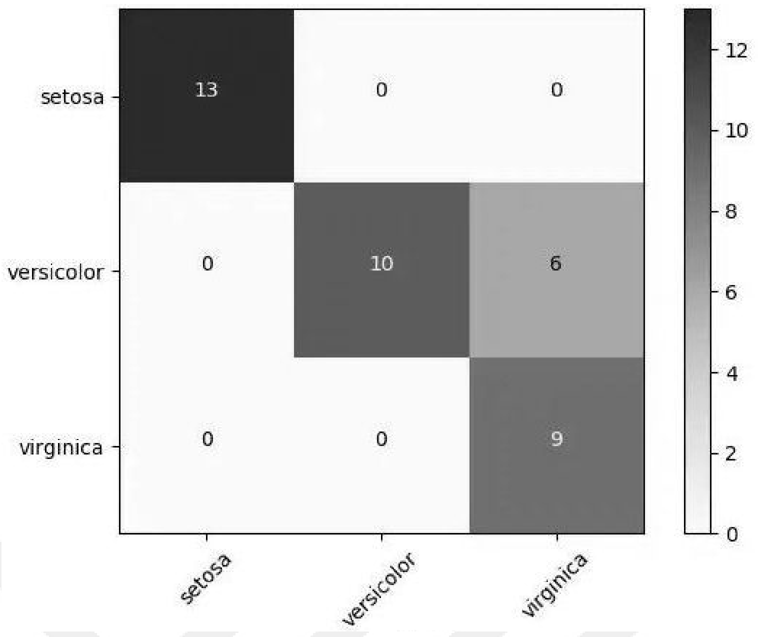

Figure 5.7: Confusion matrix for a three class model.

5.7 Optimization

Although default classifier models are sufficiently good, model parameters can be examined and can be optimized to improve classifier performance. For this purpose, model parameters are analyzed and main parameters are selected. Then by using grid search algorithms, these parameters are changed within some range and resultant model performance is examined. In this thesis optimization was done for the parameters in Table 5.1.

Table 5.1: Optimized parameter list for classifiers.

Classifier	Parameter	Range Min	Range Max
Naive Bayes	Alpha	0.01	1
Decision Trees	Max depth	1	100
Maximum Entropy	C	0.01	50
Support Vector Machines	C	0.01	10

6. EXPERIMENTS AND RESULTS

6.1 Test cases

For testing classifiers test cases were prepared. For all classifier types 3 type of vectorizer was used, Count, Hash and Tfidf. Unigrams, bigrams and trigrams were used. Cross-validation was used for 2, 5 and 10 K-folds.

For Naive Bayes alpha parameter was changed between 0.01 and 1. For Decision Trees, maximum depth was changed between 1 and 100. Maximum Entropy classifier was tested for C values between 0.01 and 50. For Support Vector Machines C parameter was changed between 0.01 and 10.

The final number of test cases was 1827. Total computation time for all tests became 77.6 hours, where each test took approximately 152 seconds. To decrease total calculation time multiprocess framework of the Python was used.

Table 6.1: Test case distribution.

Classifier	Vectorizer	No of tests
Naive Bayes	Count	189
Naive Bayes	Hash	99
Naive Bayes	Tfidf	189
Decision Trees	Count	153
Decision Trees	Hash	81
Decision Trees	Tfidf	153
Maximum Entropy	Count	180
Maximum Entropy	Hash	99
Maximum Entropy	Tfidf	180
Support Vector Machines	Count	198
Support Vector Machines	Hash	108
Support Vector Machines	Tfidf	198
Total		1827

6.2 Naive Bayes

Naive Bayes classifier was tested with 477 tests. Count, Hashing and Tfidf vectorizer were tested with 189, 99 and 189 tests respectively. From the all tests top 15, middle 15 and bottom 15 results can be seen from Table 6.2, Table 6.3 and Table 6.4.

Count vectorizer and 5-10 K-fold tests generally gave best results. Adding bigrams and trigrams improves some of the accuracy results. Best accuracy value of 0.874 obtained with these parameters : Count vectorizer, Alpha = 0.21, 10 K-fold, unigrams. For the top 15 Naive Bayes classifiers, mean of the accuracy became 0.87 with the standard deviation of 0.002. Those tests took approximately 35.2 seconds for each classifier with the standard deviation of 37.8 seconds. The mean value of the Kappa coefficient became 0.857.

Table 6.2: Naive Bayes top 15 results.

Vect.	Alpha	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
CV	0.21	10	1	0.874	0.877	0.874	0.874	11.790
CV	0.01	10	2	0.874	0.877	0.874	0.874	62.730
CV	0.06	10	2	0.874	0.878	0.874	0.874	62.900
CV	0.26	10	1	0.873	0.877	0.873	0.873	12.360
CV	0.31	10	1	0.873	0.877	0.873	0.873	12.710
CV	0.01	10	3	0.873	0.876	0.873	0.873	120.510
CV	0.36	10	1	0.872	0.876	0.872	0.872	11.500
CV	0.16	10	1	0.872	0.875	0.872	0.871	11.550
Tfidf	0.01	10	1	0.872	0.875	0.872	0.871	13.260
CV	0.41	10	1	0.871	0.875	0.871	0.870	11.700
CV	0.46	10	1	0.871	0.875	0.871	0.870	11.760
CV	0.11	10	1	0.870	0.873	0.870	0.870	11.570
CV	0.51	10	1	0.868	0.873	0.868	0.868	11.030
CV	0.06	10	3	0.868	0.873	0.868	0.867	119.890
CV	0.11	10	2	0.868	0.874	0.868	0.867	43.120
mean	-	-	-	0.872	0.875	0.872	0.871	35.225
std dev	-	-	-	0.002	0.002	0.002	0.002	37.824

Alpha accuracy graph of all Naive Bayes test results can be seen from Figure 6.1. Modifying alpha parameter changes classifier performance significantly.

Regarding Figure 6.2, increasing alpha parameter makes worsen classifier performance for Hashing and Tfidf vectorizer. For count vectorizer effect of alpha parameter change is

Table 6.3: Naive Bayes middle 15 results.

Vect.	Alpha	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
Tfidf	0.96	5	1	0.783	0.825	0.783	0.767	5.950
Tfidf	0.91	5	1	0.783	0.825	0.783	0.766	6.480
CV	0.16	2	2	0.783	0.804	0.783	0.779	8.630
CV	0.11	2	3	0.783	0.803	0.783	0.779	16.500
CV	0.71	5	3	0.781	0.833	0.781	0.774	57.850
Tfidf	1	5	1	0.781	0.825	0.781	0.765	5.680
Tfidf	0.41	5	2	0.781	0.826	0.781	0.764	31.530
Tfidf	0.66	10	2	0.781	0.826	0.781	0.763	70.260
Tfidf	0.21	2	1	0.781	0.807	0.781	0.773	2.060
Tfidf	0.51	10	3	0.781	0.825	0.781	0.763	126.410
Tfidf	0.71	10	2	0.781	0.826	0.781	0.762	67.040
CV	0.96	5	2	0.780	0.831	0.780	0.773	31.300
Tfidf	0.31	5	3	0.780	0.824	0.780	0.763	61.280
CV	0.76	5	3	0.778	0.831	0.778	0.771	57.890
Tfidf	0.56	10	3	0.778	0.823	0.778	0.760	130.620
mean	-	-	-	0.781	0.822	0.781	0.768	45.299
std dev	-	-	-	0.002	0.009	0.002	0.006	40.480

Table 6.4: Naive Bayes bottom 15 results.

Vect.	Alpha	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
Hash	0.51	2	3	0.664	0.722	0.664	0.627	11.390
Hash	0.61	2	2	0.664	0.722	0.664	0.627	8.790
Hash	0.61	2	3	0.662	0.723	0.662	0.624	12.000
Hash	0.71	2	2	0.659	0.721	0.659	0.622	8.670
Hash	0.71	2	1	0.658	0.721	0.658	0.623	4.650
Hash	0.81	2	2	0.657	0.721	0.657	0.619	8.410
Hash	0.81	2	1	0.656	0.721	0.656	0.620	4.820
Hash	0.71	2	3	0.655	0.721	0.655	0.616	11.850
Hash	0.91	2	1	0.653	0.722	0.653	0.617	4.890
Hash	1	2	1	0.650	0.722	0.650	0.614	4.910
Hash	0.91	2	2	0.650	0.720	0.650	0.612	8.350
Hash	0.81	2	3	0.646	0.719	0.646	0.606	10.960
Hash	1	2	2	0.644	0.719	0.644	0.605	8.680
Hash	0.91	2	3	0.643	0.720	0.643	0.602	11.300
Hash	1	2	3	0.640	0.719	0.640	0.599	10.900
mean	-	-	-	0.653	0.721	0.653	0.616	8.705
std dev	-	-	-	0.007	0.001	0.007	0.009	2.650

different when there are only unigrams. For that case increasing alpha improves accuracy up to some point, after that performance degrades again.

The computation time of the classifier is another performance issue. As expected from Figure 6.3 we can see that 2-fold classifiers took much less time than the 5-fold and 10-fold classifiers. Duration to obtain the classifier changes linearly. For the optimum classifier, it can be seen that result was obtained in nearly 10-11 seconds.

During the performance evaluation accuracy value was used. But one should take fscore and kappa into account also. To check whether these two parameters are consistent with accuracy values, Figure 6.4 and Figure 6.5 can be checked.

For each vectorizer, k-fold and n-gram case best accuracy results can be seen from Figure 6.6. Count vectorizer outperforms other vectorizers for the majority of the cases.

Effect of the alpha parameter in optimal Naive Bayes classifier can be seen from Figure 6.7.

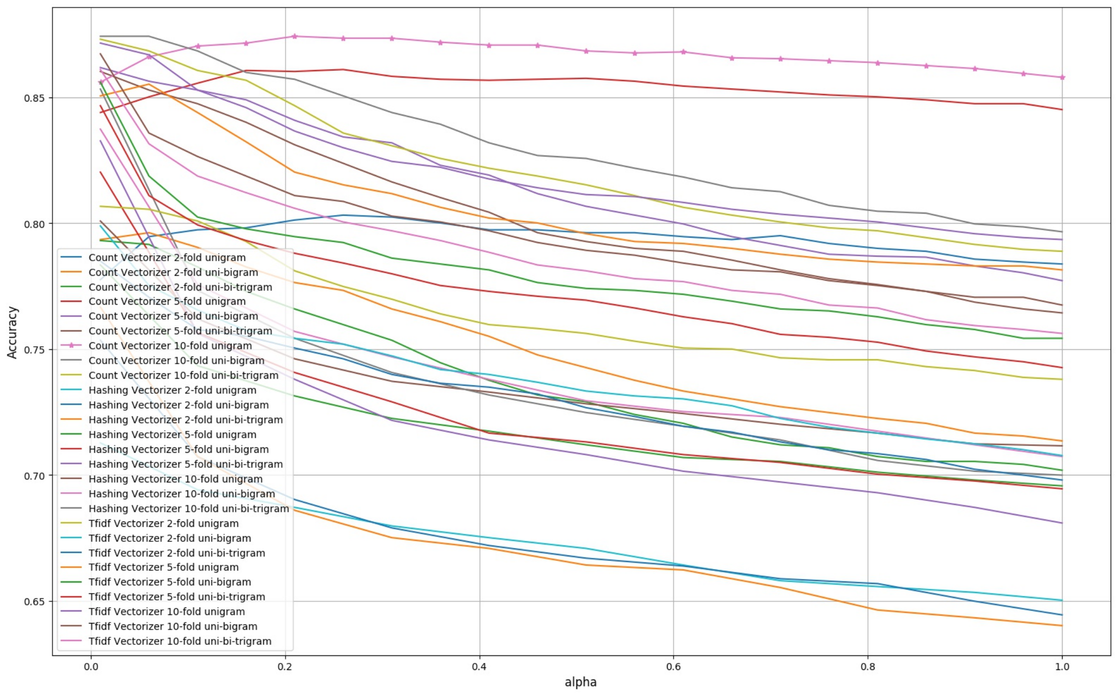

Figure 6.1: Naive Bayes alpha-accuracy graph for all tests.

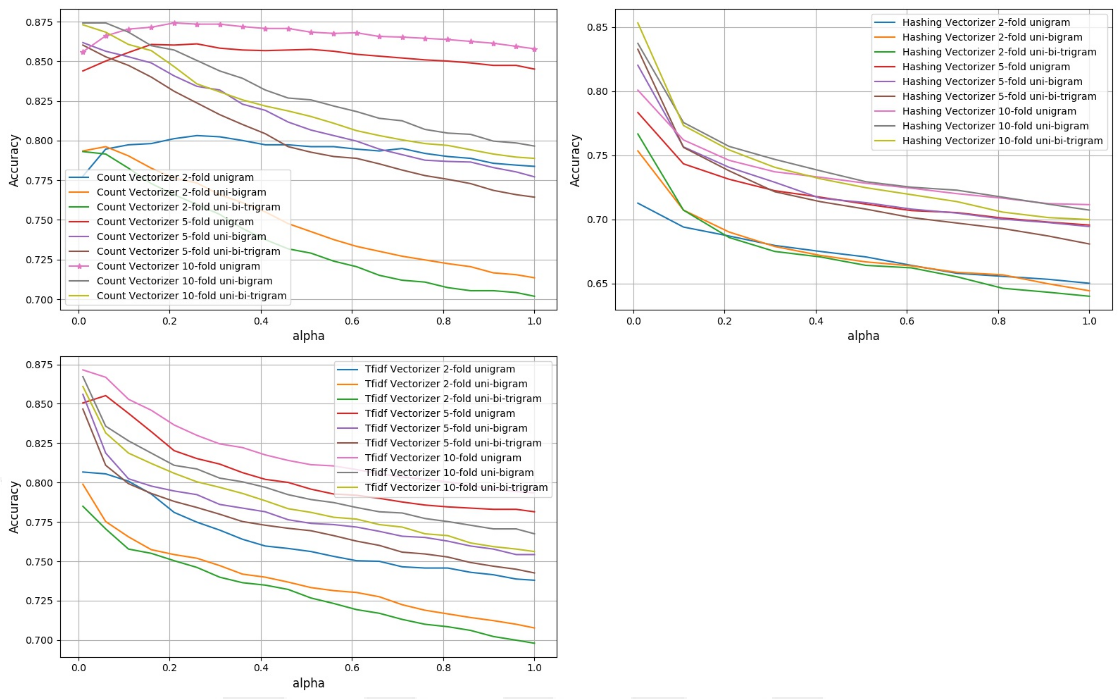

Figure 6.2: Naive Bayes alpha-accuracy graph for each vectorizer.

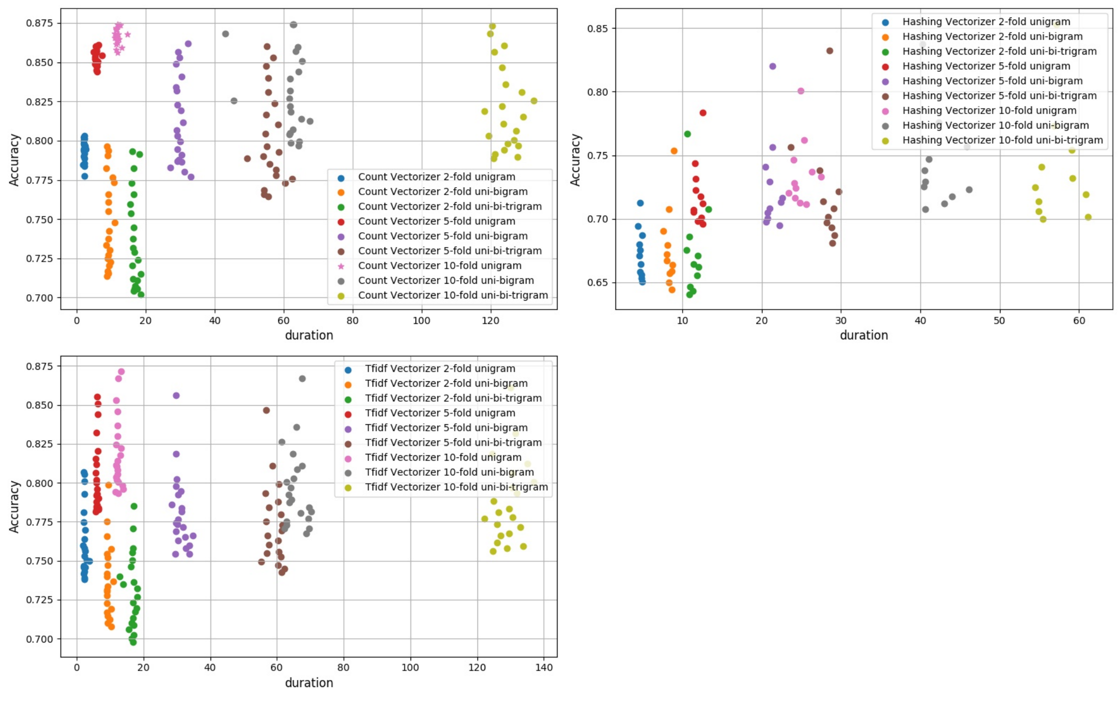

Figure 6.3: Naive Bayes duration-accuracy graph for each vectorizer.

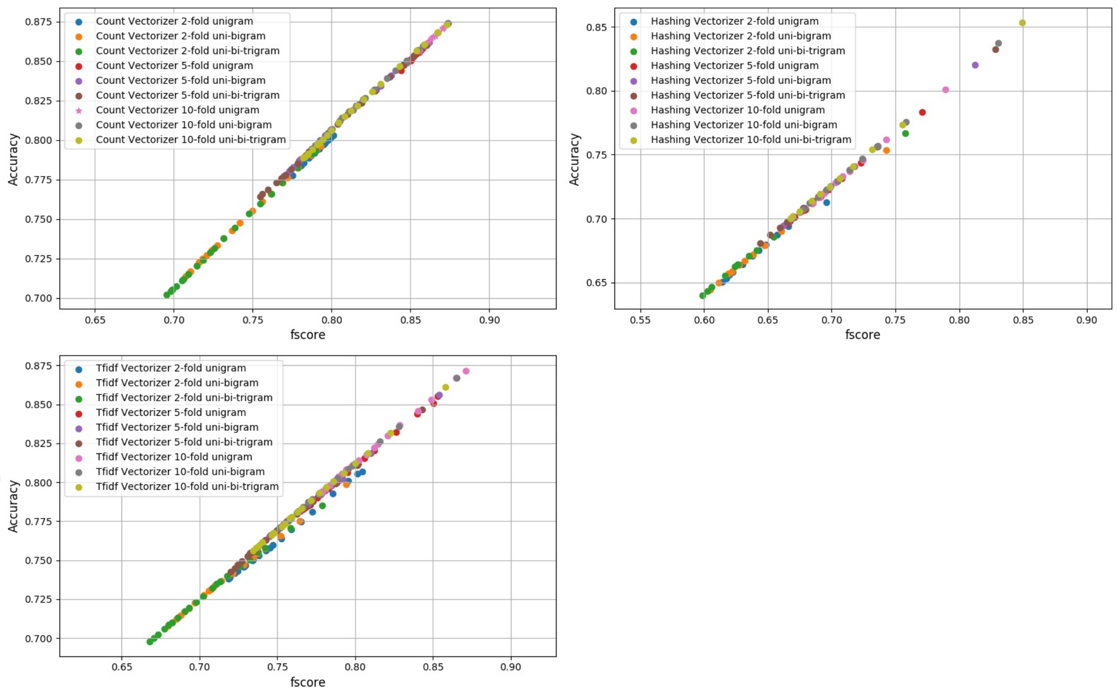

Figure 6.4: Naive Bayes fscore-accuracy graph for each vectorizer.

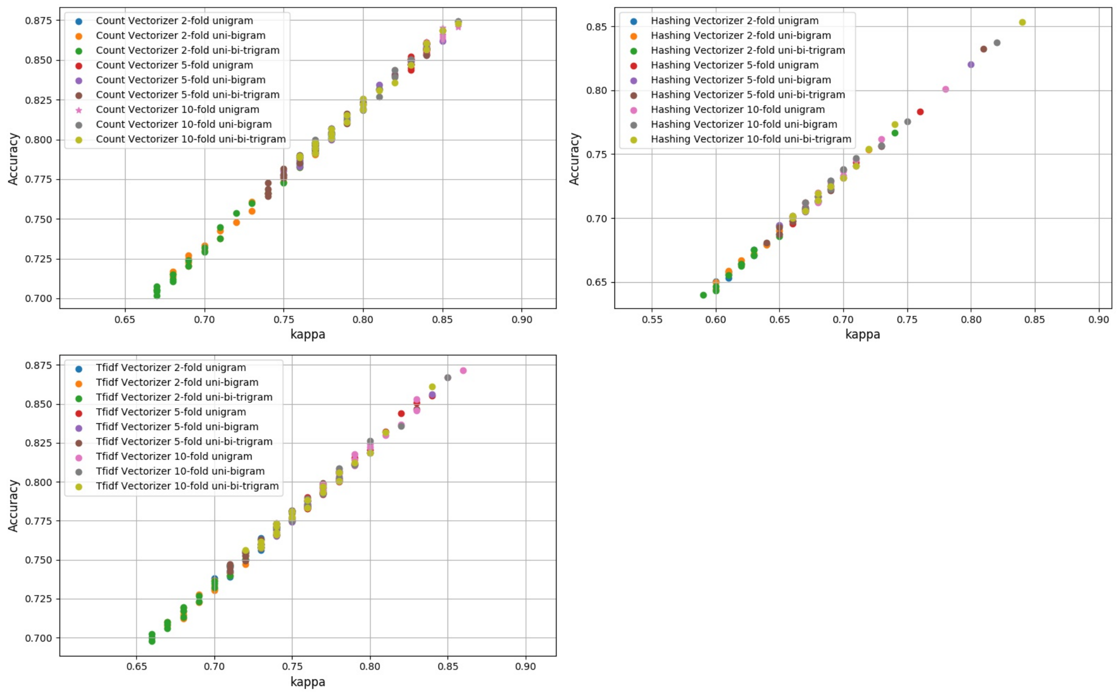

Figure 6.5: Naive Bayes kappa-accuracy graph for each vectorizer.

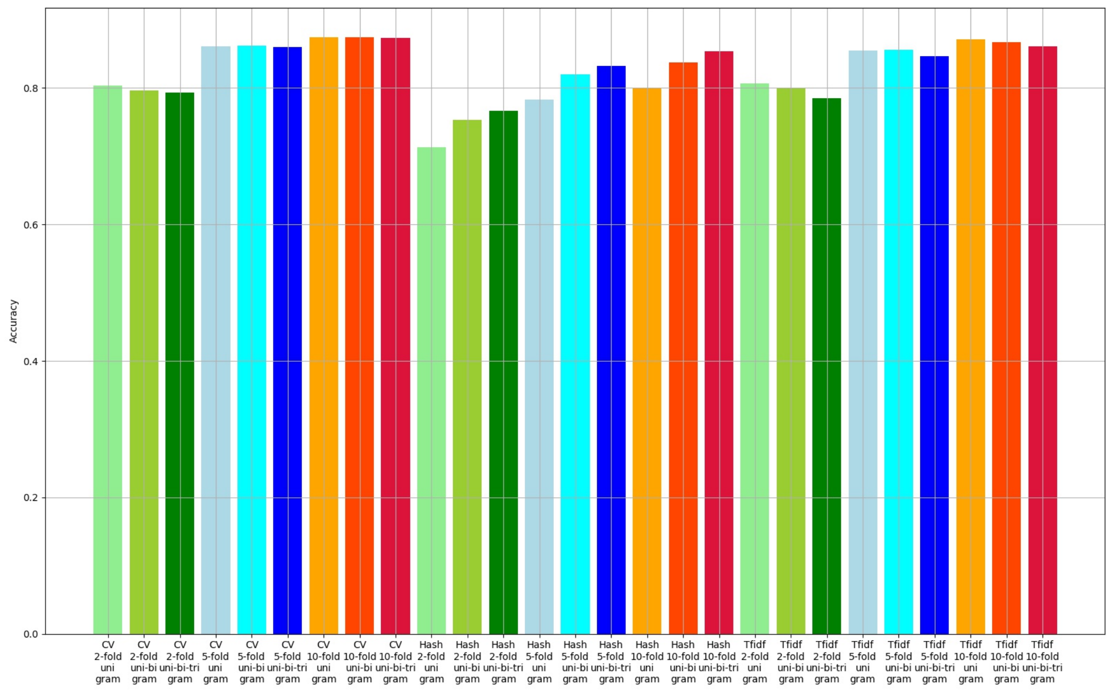

Figure 6.6: Naive Bayes best accuracy values for each vectorizer, k-fold and ngram case.

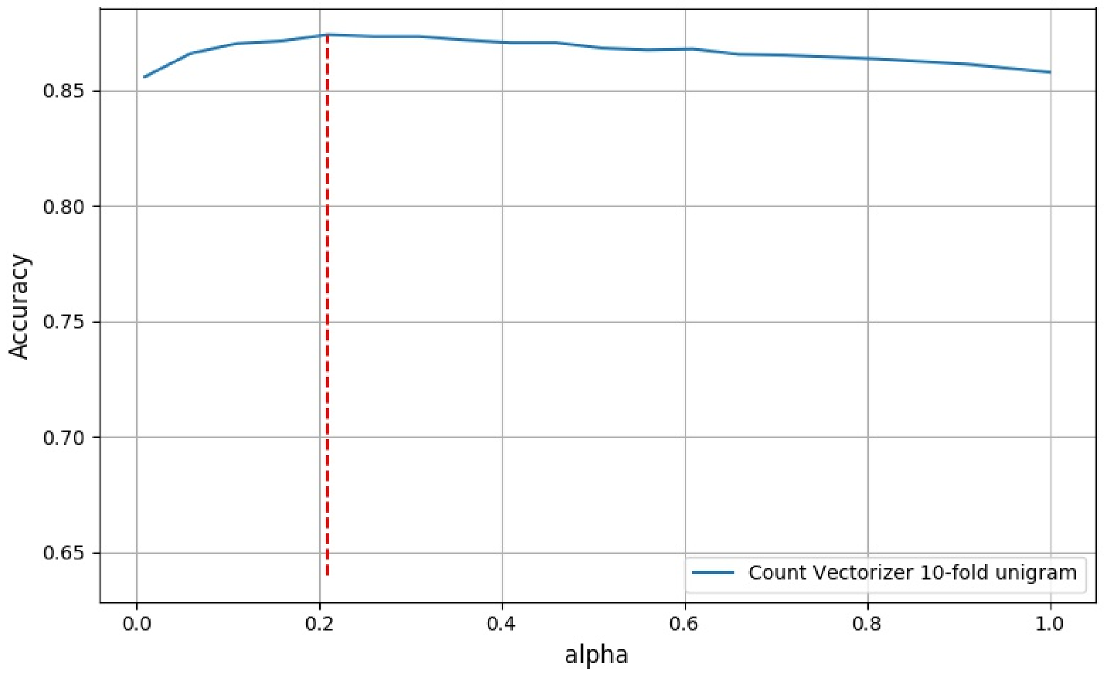

Figure 6.7: Naive Bayes alpha parameter for the optimal classifier.

6.3 Decision Trees

Decision Tree type classifier was tested with 387 tests. For these tests the maximum depth of the trees, k-fold and existence of bigrams and trigrams changed. Count vectorizer was analysed in 153 tests, Hashing vectorizer in 81 and Tfidf in 153 tests.

Top, middle and bottom 15 test cases can be seen in Table 6.5, Table 6.6 and Table 6.7. In the top results, it is clear that Count vectorizer outperformed other two vectorizers. 10 K-fold became optimum for nearly every case. Adding unigrams and trigrams improved accuracy results. Trees that have 1 level could not pass accuracy threshold of 0.19. Best accuracy value obtained when Count Vectorizer was used with the maximum tree depth of 50, 10 K-fold and trigrams. Mean accuracy for top 15 classifiers became 0.637. Also, calculations took too much time where the mean duration for a decision tree classifier is 463 seconds with the standart error of 248 seconds.

Table 6.5: Decision Trees top 15 results.

Vect.	Depth	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
CV	50	10	3	0.651	0.652	0.651	0.650	692.940
CV	90	10	3	0.649	0.649	0.649	0.648	674.500
CV	70	10	3	0.646	0.648	0.646	0.647	736.340
CV	90	10	2	0.646	0.646	0.646	0.646	311.660
CV	70	10	2	0.644	0.645	0.644	0.644	375.800
CV	50	10	2	0.642	0.645	0.642	0.643	349.300
CV	100	10	3	0.642	0.642	0.642	0.642	816.780
CV	100	10	2	0.641	0.640	0.641	0.640	369.500
CV	40	10	2	0.638	0.646	0.638	0.641	291.080
CV	40	10	3	0.632	0.639	0.632	0.634	758.200
CV	30	10	2	0.628	0.646	0.628	0.634	255.910
CV	100	10	1	0.626	0.628	0.626	0.626	41.150
CV	70	10	1	0.625	0.628	0.625	0.626	51.700
CV	30	10	3	0.624	0.645	0.624	0.630	725.640
Hash	70	10	2	0.623	0.625	0.623	0.624	505.210
mean	-	-	-	0.637	0.642	0.637	0.638	463.714
std dev	-	-	-	0.010	0.008	0.010	0.008	248.612

For the maximum depth range of 1 to 100, depth-accuracy results can be seen from Figure 6.8. Increasing depth improves accuracy results at the start, but performance change becomes stable after 40 levels. For the decision trees, accuracy values do not change too

Table 6.6: Decision Trees middle 15 results.

Vect.	Depth	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
CV	40	2	1	0.514	0.530	0.514	0.518	3.900
Tfidf	15	5	2	0.514	0.586	0.514	0.525	102.490
Tfidf	17	5	3	0.514	0.557	0.514	0.519	246.490
Hash	13	5	2	0.513	0.627	0.513	0.533	102.870
CV	20	2	2	0.513	0.575	0.513	0.526	23.280
Tfidf	15	10	2	0.513	0.593	0.513	0.523	263.370
Tfidf	40	2	2	0.512	0.517	0.512	0.513	37.510
Hash	13	10	2	0.512	0.653	0.512	0.538	236.600
Tfidf	17	10	3	0.510	0.552	0.510	0.518	444.670
Tfidf	19	10	3	0.510	0.540	0.510	0.516	647.890
Hash	13	10	1	0.509	0.611	0.509	0.529	181.320
CV	20	2	3	0.509	0.577	0.509	0.523	57.030
CV	11	10	2	0.509	0.657	0.509	0.541	178.000
Tfidf	50	2	3	0.507	0.508	0.507	0.506	61.110
Tfidf	15	5	3	0.507	0.570	0.507	0.514	203.400
mean	-	-	-	0.511	0.577	0.511	0.523	185.995
std dev	-	-	-	0.002	0.044	0.002	0.009	167.461

Table 6.7: Decision Trees bottom 15 results.

Vect.	Depth	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
CV	1	5	2	0.196	0.097	0.196	0.104	42.120
Hash	1	10	1	0.196	0.095	0.196	0.104	37.610
Hash	1	5	2	0.196	0.095	0.196	0.103	28.660
Tfidf	1	10	1	0.196	0.094	0.196	0.103	10.090
Hash	1	5	1	0.195	0.096	0.195	0.103	16.130
Tfidf	1	5	1	0.195	0.095	0.195	0.103	7.160
Tfidf	1	2	3	0.183	0.156	0.183	0.107	16.800
Tfidf	1	2	2	0.181	0.165	0.181	0.107	6.690
Hash	1	2	3	0.177	0.172	0.177	0.104	14.090
Hash	1	2	1	0.177	0.182	0.177	0.104	6.120
Hash	1	2	2	0.177	0.173	0.177	0.104	17.170
CV	1	2	2	0.173	0.180	0.173	0.125	8.960
CV	1	2	3	0.173	0.180	0.173	0.125	52.800
CV	1	2	1	0.173	0.180	0.173	0.125	2.440
Tfidf	1	2	1	0.173	0.176	0.173	0.100	7.770
mean	-	-	-	0.184	0.142	0.184	0.108	18.307
std dev	-	-	-	0.010	0.039	0.010	0.009	14.579

much between the vectorizers, but count vectorizer can be seen as a better option regarding Figure 6.9. Decision trees need too much computation time. In the worst case which includes 10-fold, bigrams and trigrams, computation can take 250-300 seconds for each classifier. Bar charts in Figure 6.11 show that increasing number of K-folds improves accuracy nearly for all cases in Count vectorizer. Depth effect for the optimum decision trees classifier was shown in Figure 6.12. Increasing its value up to 45-50 improves accuracy, however, performance decrease starts after that level.

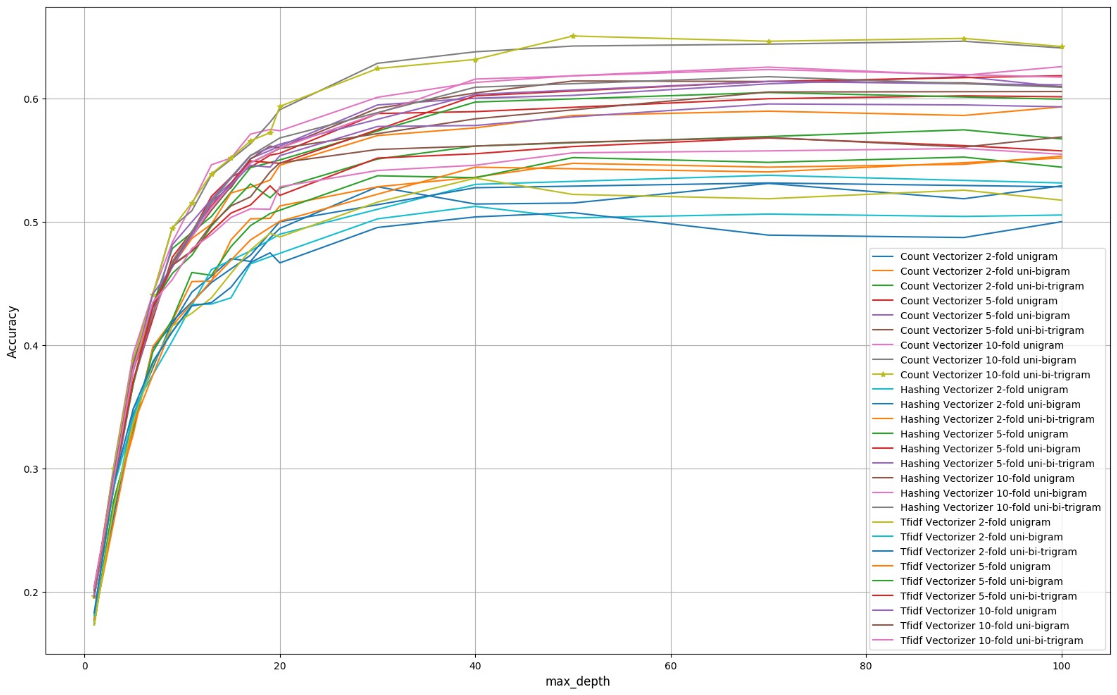

Figure 6.8: Decision Trees maximum depth-accuracy graph for all tests.

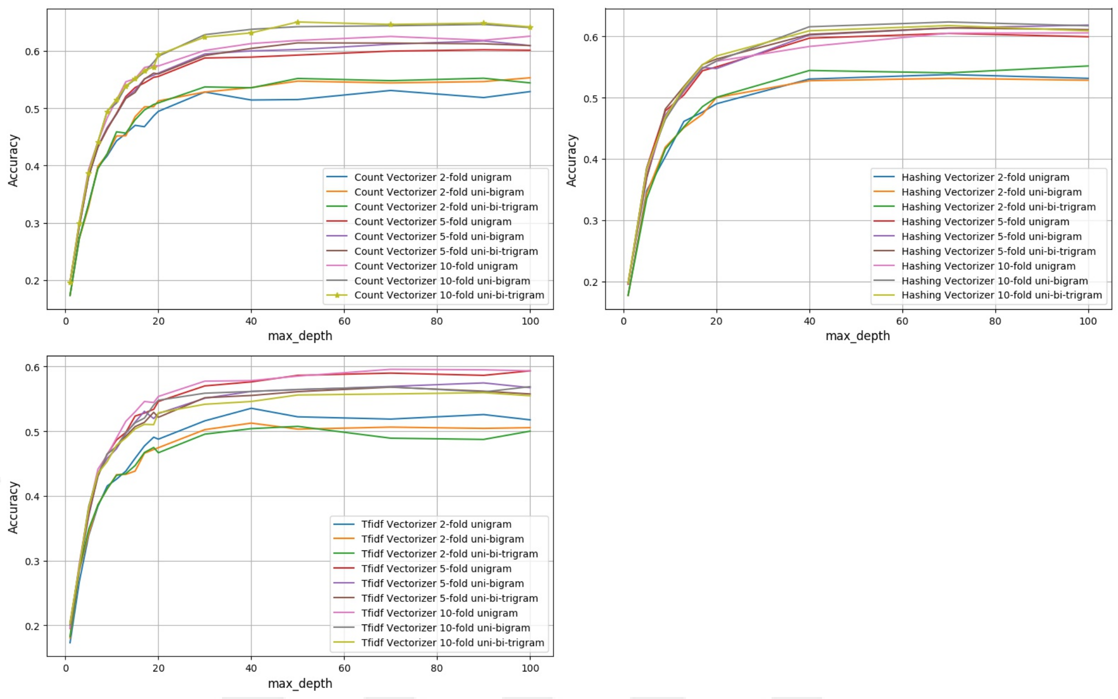

Figure 6.9: Decision Trees maximum depth-accuracy graph for each vectorizer.

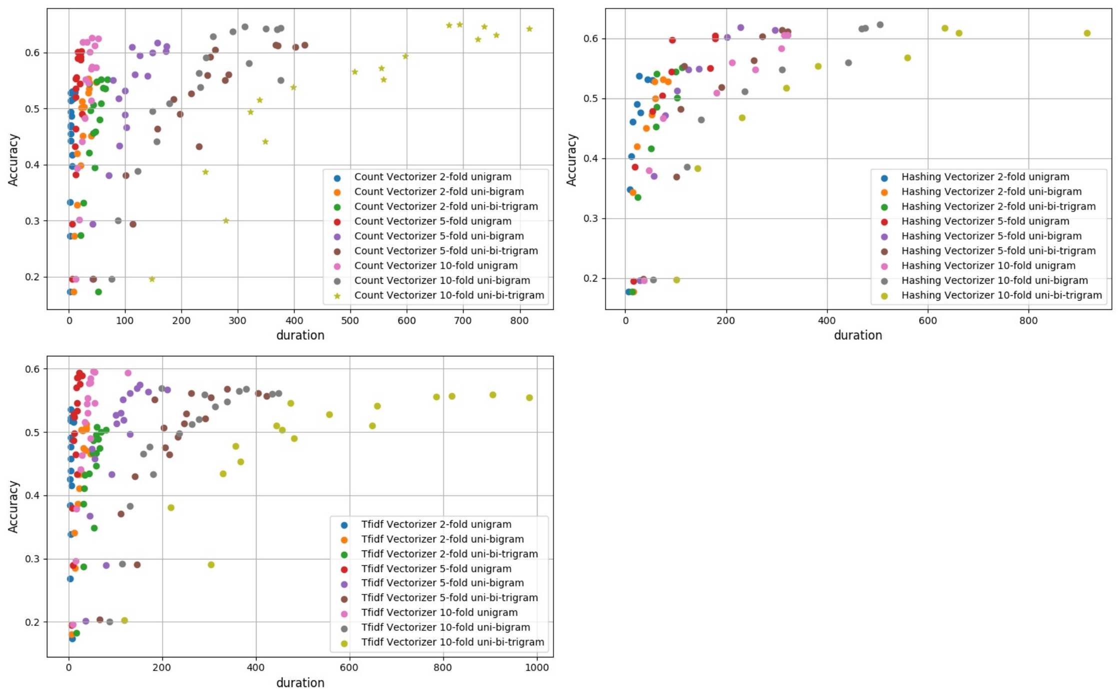

Figure 6.10: Decision Trees duration-accuracy graph for each vectorizer.

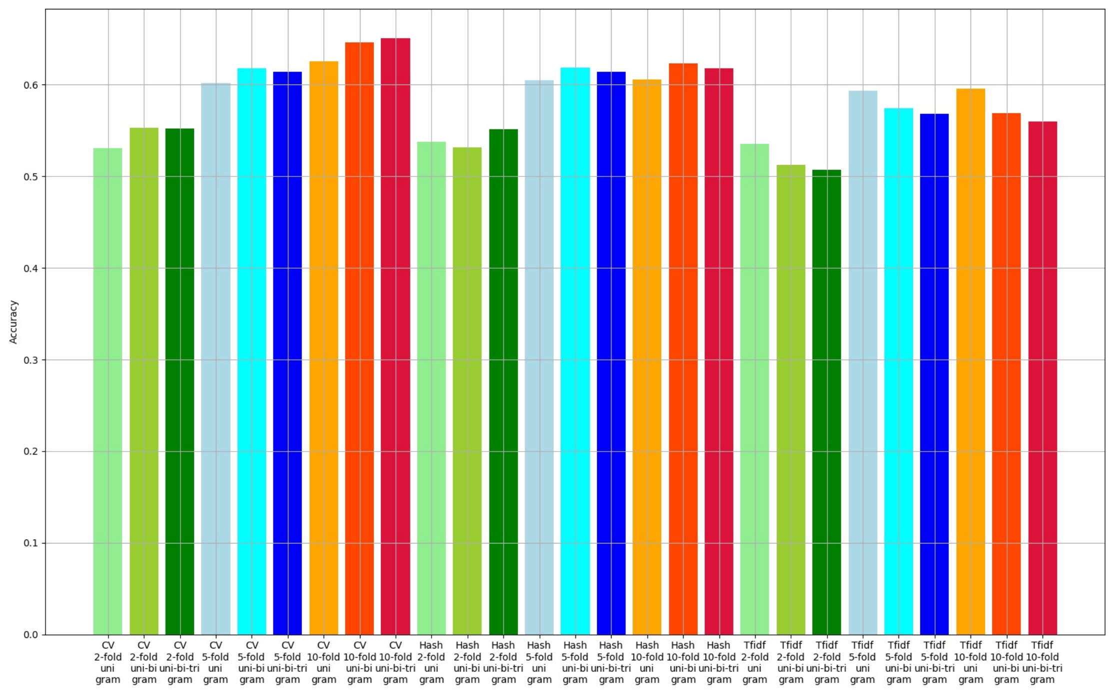

Figure 6.11: Decision Trees best accuracy values for each vectorizer, k-fold and ngram case.

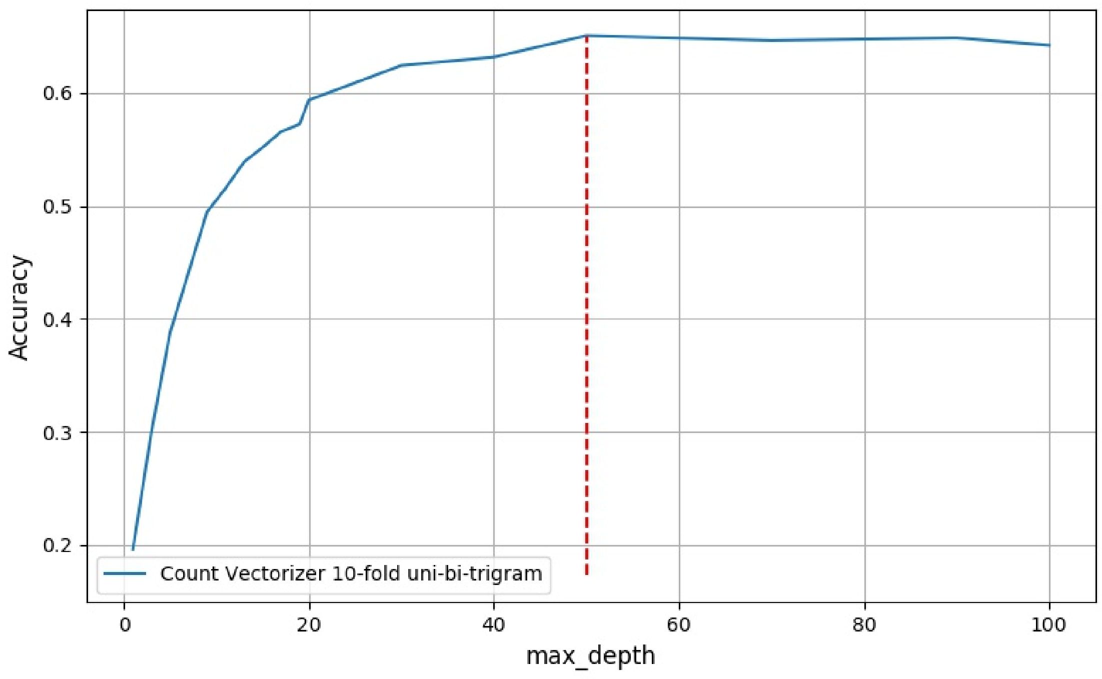

Figure 6.12: Decision Trees maximum depth parameter for the optimal classifier.

6.4 Maximum Entropy

Maximum Entropy classifier was tested with 459 test cases. Count, Hash and Tfidf Vectorizers were used 180, 99 and 180 times respectively.

From all cases top, medium and bottom cases can be seen from Table 6.8, Table 6.9 and Table 6.10. Mean accuracy value for top 15 results is 0.886 and mean f-score is 0.886. Regardless of test case types computation time is high for nearly all of the cases. A classifier can be obtained even in 500-750 seconds. Increasing K-fold from 2 to 5 or from 5 to 10 increases computation time significantly. For the top 15 Maximum Entropy classifier, average computation time is 377 seconds.

Table 6.8: Maximum Entropy top 15 results.

Vect.	C	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
Tfidf	50	10	2	0.888	0.890	0.888	0.888	490.310
Tfidf	40	10	2	0.888	0.890	0.888	0.888	959.510
Tfidf	50	10	1	0.887	0.889	0.887	0.887	226.440
Tfidf	30	10	1	0.887	0.889	0.887	0.887	60.190
Tfidf	40	10	1	0.887	0.888	0.887	0.887	244.210
Tfidf	20	10	1	0.886	0.888	0.886	0.886	313.970
Tfidf	30	10	2	0.886	0.888	0.886	0.886	552.560
Tfidf	50	10	3	0.885	0.888	0.885	0.885	610.810
Tfidf	15	10	1	0.885	0.887	0.885	0.885	72.410
Tfidf	30	10	3	0.885	0.887	0.885	0.884	860.220
Tfidf	4.01	10	1	0.884	0.889	0.884	0.885	419.310
Tfidf	3.21	10	1	0.884	0.889	0.884	0.885	105.200
Tfidf	3.61	10	1	0.884	0.889	0.884	0.885	32.720
Tfidf	15	10	2	0.884	0.887	0.884	0.884	267.790
Tfidf	20	10	2	0.884	0.887	0.884	0.884	448.490
mean	-	-	-	0.886	0.888	0.886	0.886	377.609
std dev	-	-	-	0.001	0.001	0.001	0.001	273.227

Variation of C parameter versus accuracy for all cases can be seen from Figure 6.13. Increasing C parameter improves accuracy for nearly every case. After a threshold, increasing C does not change the performance of the classifier. As seen in Figure 6.14 vectorizer change does not differ the performance. However, Tfidf vectorizer seems to be a better option for Maximum Entropy. For C parameter 50 would be an appropriate

Table 6.9: Maximum Entropy middle 15 results.

Vect.	C	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
CV	30	5	2	0.864	0.866	0.864	0.864	486.900
Tfidf	2.01	5	3	0.864	0.872	0.864	0.864	383.870
Tfidf	1.61	5	3	0.864	0.873	0.864	0.864	259.420
CV	20	5	2	0.864	0.866	0.864	0.864	481.210
CV	50	10	1	0.864	0.864	0.864	0.864	379.210
CV	2.01	5	1	0.864	0.865	0.864	0.864	418.540
CV	2.41	5	1	0.864	0.865	0.864	0.864	122.160
Hash	20	5	3	0.864	0.866	0.864	0.863	248.390
Hash	4.01	10	3	0.863	0.869	0.863	0.864	823.420
Hash	4.81	5	1	0.863	0.866	0.863	0.863	235.930
CV	40	5	2	0.863	0.865	0.863	0.863	455.810
Hash	10	5	3	0.863	0.867	0.863	0.863	415.720
CV	2.81	5	1	0.863	0.865	0.863	0.863	90.270
CV	0.41	5	1	0.863	0.865	0.863	0.863	66.880
CV	20	5	3	0.863	0.865	0.863	0.863	734.400
mean	-	-	-	0.864	0.867	0.864	0.864	373.475
std dev	-	-	-	0.001	0.003	0.001	0.001	208.625

Table 6.10: Maximum Entropy bottom 15 results.

Vect.	C	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
Tfidf	0.01	10	1	0.329	0.515	0.329	0.264	27.650
Tfidf	0.01	5	1	0.316	0.512	0.316	0.249	92.920
Hash	0.01	2	1	0.313	0.483	0.313	0.219	80.420
Hash	0.01	5	2	0.310	0.506	0.310	0.233	182.010
Hash	0.01	10	3	0.290	0.441	0.290	0.219	376.760
Tfidf	0.01	2	1	0.279	0.447	0.279	0.206	74.960
Hash	0.01	5	3	0.274	0.339	0.274	0.201	187.370
Hash	0.01	2	2	0.269	0.327	0.269	0.183	31.190
Hash	0.01	2	3	0.234	0.346	0.234	0.159	53.500
Tfidf	0.01	10	2	0.233	0.364	0.233	0.172	255.620
Tfidf	0.01	5	2	0.217	0.364	0.217	0.153	142.880
Tfidf	0.01	2	2	0.186	0.367	0.186	0.118	77.870
Tfidf	0.01	10	3	0.174	0.253	0.174	0.102	386.010
Tfidf	0.01	5	3	0.164	0.250	0.164	0.087	112.720
Tfidf	0.01	2	3	0.148	0.259	0.148	0.062	51.740
mean	-	-	-	0.249	0.385	0.249	0.175	142.241
std dev	-	-	-	0.058	0.091	0.058	0.059	112.018

value. It can be seen that accuracy does not change too much between 5 and 50 values of C.

Figure 6.15 shows how the computation time is scattered. Although marginal values like 1000-1750 seconds exist, 200-300 seconds are very common for this type of classifier. Optimal accuracy for each group can be seen from Figure 6.16. None of the group outperforms other values. Obviously, increasing K-folds improves accuracy, adding bigram and trigrams slightly improves or degrades performance. In the range of 0-50, change of C parameter versus accuracy was plotted in Figure 6.17.

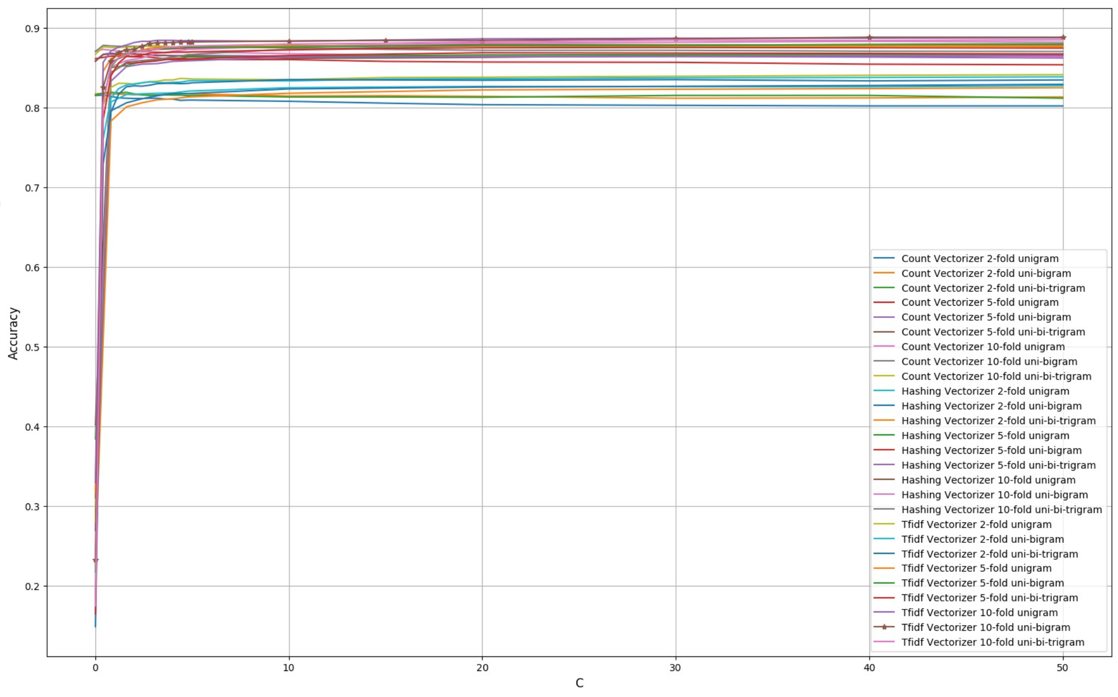

Figure 6.13: Maximum Entropy C-accuracy graph for all tests.

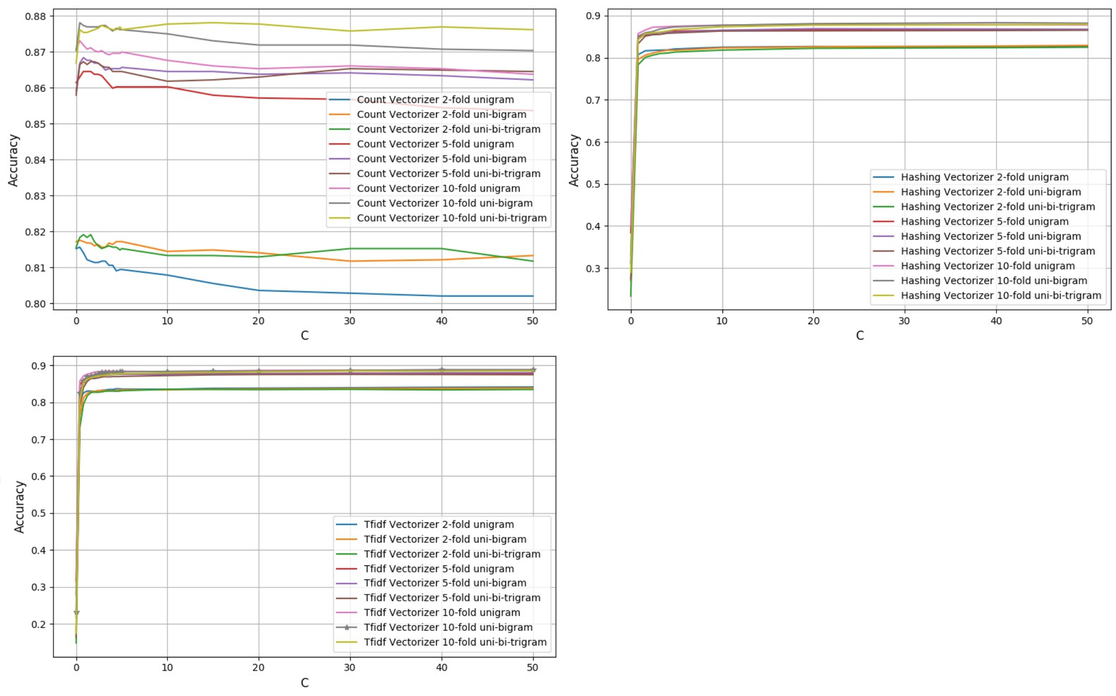

Figure 6.14: Maximum Entropy C-accuracy graph for each vectorizer.

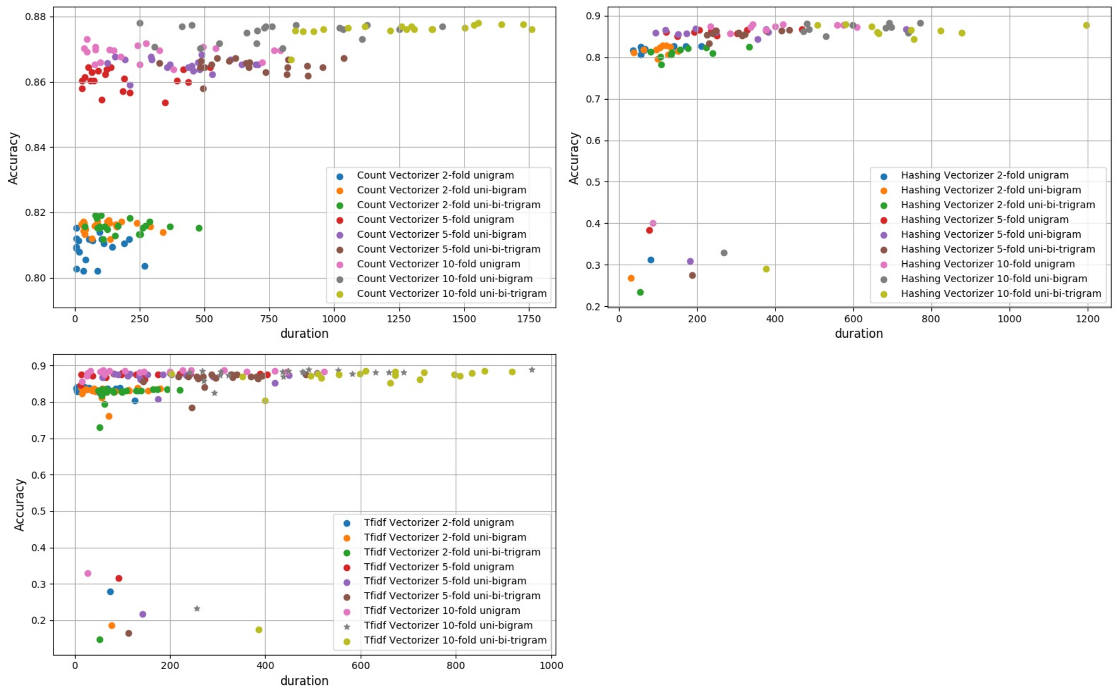

Figure 6.15: Maximum Entropy duration-accuracy graph for each vectorizer.

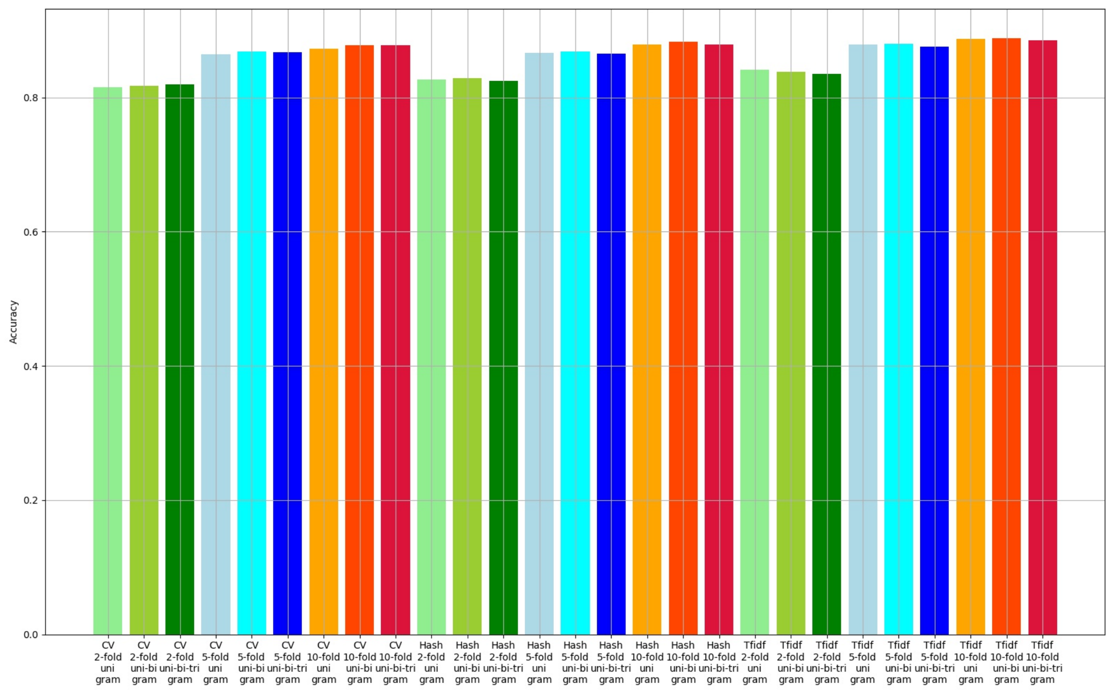

Figure 6.16: Maximum Entropy best accuracy values for each vectorizer, k-fold and ngram case.

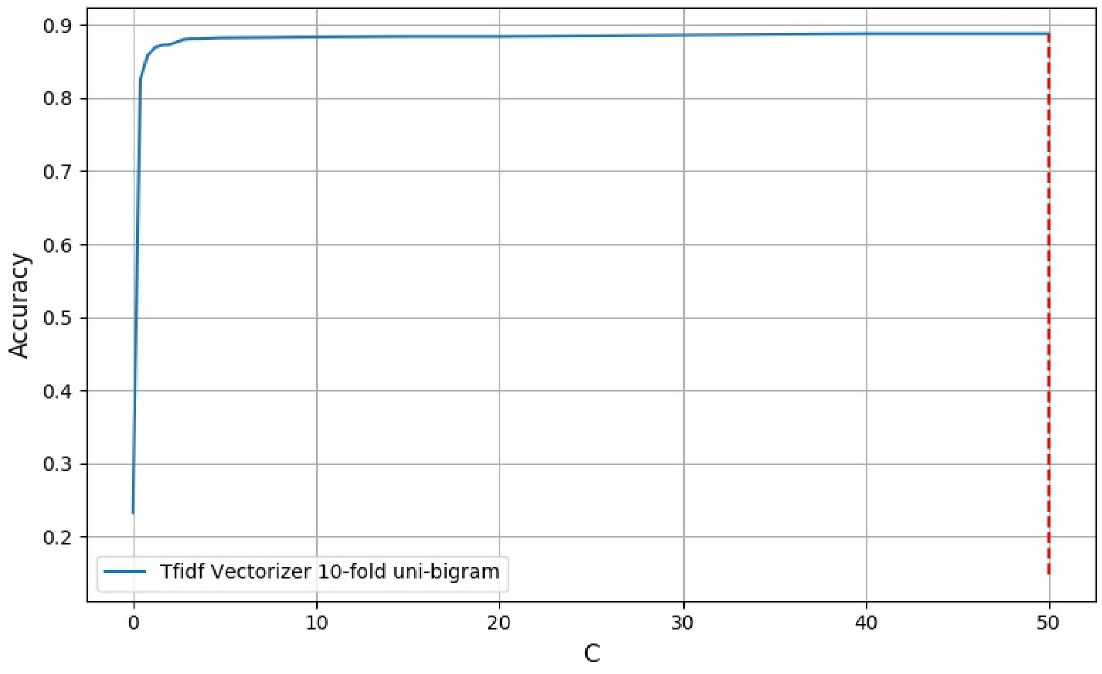

Figure 6.17: Maximum Entropy C parameter for the optimal classifier.

6.5 Support Vector Machines

For the Support Vector Machines (SVM) 504 tests prepared. Count, Hash and Tfidf vectorizers were tested in 198, 108 and 198 cases respectively.

Top 15 results of SVM classifier shows that Tfidf vectorizer outperforms in nearly all cases. Also 10 K-folds and bigrams exist in top SVM cases as seen in Table 6.8. Mean accuracy value for top classifiers became 0.891 and f-score is 0.891. Average calculation time is 238 seconds with standard error of 108 seconds.

Middle and bottom cases was shown in Table 6.9 and Table 6.10. Tfidf vectorizers also exist in bottom results. Selecting an inappropriate C value creates a low performance classifier.

Table 6.11: Support Vector Machines top 15 results.

Vect.	C	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
Tfidf	10	10	2	0.893	0.894	0.893	0.892	453.870
Tfidf	5	10	2	0.892	0.894	0.892	0.892	286.500
Tfidf	4.76	10	2	0.892	0.893	0.892	0.891	213.490
Tfidf	3.76	10	2	0.892	0.893	0.892	0.891	156.010
Tfidf	3.51	10	2	0.892	0.893	0.892	0.891	170.330
Tfidf	3.26	10	2	0.892	0.893	0.892	0.891	204.100
Tfidf	4.01	10	2	0.892	0.893	0.892	0.891	253.770
Tfidf	2.01	10	2	0.891	0.893	0.891	0.891	143.960
Tfidf	3.01	10	2	0.891	0.893	0.891	0.891	200.100
Tfidf	4.26	10	2	0.891	0.893	0.891	0.891	205.300
Tfidf	4.51	10	2	0.891	0.893	0.891	0.891	198.120
Tfidf	2.26	10	2	0.891	0.893	0.891	0.890	162.120
Tfidf	2.76	10	2	0.891	0.893	0.891	0.890	174.710
Tfidf	2.51	10	2	0.891	0.893	0.891	0.890	211.910
Tfidf	4.51	10	3	0.889	0.891	0.889	0.889	540.760
mean	-	-	-	0.891	0.893	0.891	0.891	238.337
std dev	-	-	-	0.001	0.001	0.001	0.001	108.719

Tfidf vectorizer, unigram and bigram case outperforms in all variation of C except for the values near to zero. As seen in Figure 6.18 changing C to values bigger than 3-4 does not improve the accuracy drastically. For SVM classifier, Count Vectorizer performs different behavior as seen in Figure 6.19. The increase in C value degrades accuracy at the start and the performance stays stable. On the other hand, the increase in C for Tfidf and Hash

Table 6.12: Support Vector Machines middle 15 results.

Vect.	C	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
CV	0.76	5	3	0.856	0.858	0.856	0.856	164.220
CV	0.51	5	2	0.856	0.857	0.856	0.856	70.920
CV	0.76	5	2	0.855	0.857	0.855	0.855	51.910
CV	1.26	5	3	0.854	0.857	0.854	0.855	165.410
CV	1.01	5	3	0.854	0.857	0.854	0.855	268.310
CV	1.01	5	2	0.854	0.856	0.854	0.855	94.130
CV	1.51	5	2	0.854	0.856	0.854	0.855	79.910
CV	1.51	5	3	0.854	0.857	0.854	0.854	258.340
CV	1.76	5	3	0.854	0.857	0.854	0.854	255.730
CV	1.76	5	2	0.854	0.856	0.854	0.854	108.260
CV	2.01	5	2	0.854	0.856	0.854	0.854	179.930
CV	2.51	5	2	0.854	0.856	0.854	0.854	105.150
CV	2.26	5	2	0.854	0.856	0.854	0.854	65.900
CV	2.76	5	2	0.854	0.856	0.854	0.854	89.130
CV	3.01	5	2	0.854	0.856	0.854	0.854	138.030
mean	-	-	-	0.854	0.857	0.854	0.855	139.685
std dev	-	-	-	0.001	0.001	0.001	0.001	70.823

Table 6.13: Support Vector Machines bottom 15 results.

Vect.	C	K-fold	Ngram	Acc.	Prec.	Recall	F score	Duration (seconds)
CV	2.51	2	1	0.780	0.784	0.780	0.779	3.260
CV	2.76	2	1	0.780	0.783	0.780	0.778	9.440
Hash	0.01	5	2	0.769	0.819	0.769	0.763	38.420
Hash	0.01	10	3	0.757	0.817	0.757	0.747	89.830
Tfidf	0.01	10	2	0.750	0.824	0.750	0.737	91.480
Hash	0.01	2	1	0.743	0.791	0.743	0.736	5.220
Tfidf	0.01	2	1	0.741	0.815	0.741	0.731	2.690
Hash	0.01	5	3	0.733	0.808	0.733	0.722	46.050
Tfidf	0.01	5	2	0.727	0.817	0.727	0.711	38.930
Tfidf	0.01	10	3	0.712	0.766	0.712	0.693	227.210
Hash	0.01	2	2	0.695	0.787	0.695	0.683	10.850
Tfidf	0.01	5	3	0.685	0.760	0.685	0.664	79.930
Hash	0.01	2	3	0.652	0.741	0.652	0.635	14.320
Tfidf	0.01	2	2	0.648	0.753	0.648	0.623	9.610
Tfidf	0.01	2	3	0.593	0.747	0.593	0.559	32.820
mean	-	-	-	0.718	0.787	0.718	0.704	46.671
std dev	-	-	-	0.052	0.028	0.052	0.060	56.873

vectorizers makes the performance better. Additionally, the SVM classifiers need too much computation time different from the Naive Bayes classifiers. For a top classifier

150-200 seconds are generally needed. Increasing K-fold value increases computation time linearly, 100-150 seconds is generally sufficient as seen in Figure 6.20 except the 10 K-fold case.

Best SVM classifiers were obtained by using Tfidf vectorizers. From Figure 6.21 and Figure 6.22 it can be seen that F-score and Kappa values have a good match with the accuracy values. As seen in Figure 6.23 5 and 10 K-fold Tfidf vectorizers outperforms nearly all other classifiers. For other groups, increasing K-fold values improved accuracy in nearly all cases. C parameter was changed between 0-10 and its effect on optimum SVM classifier was plotted in Figure 6.24.

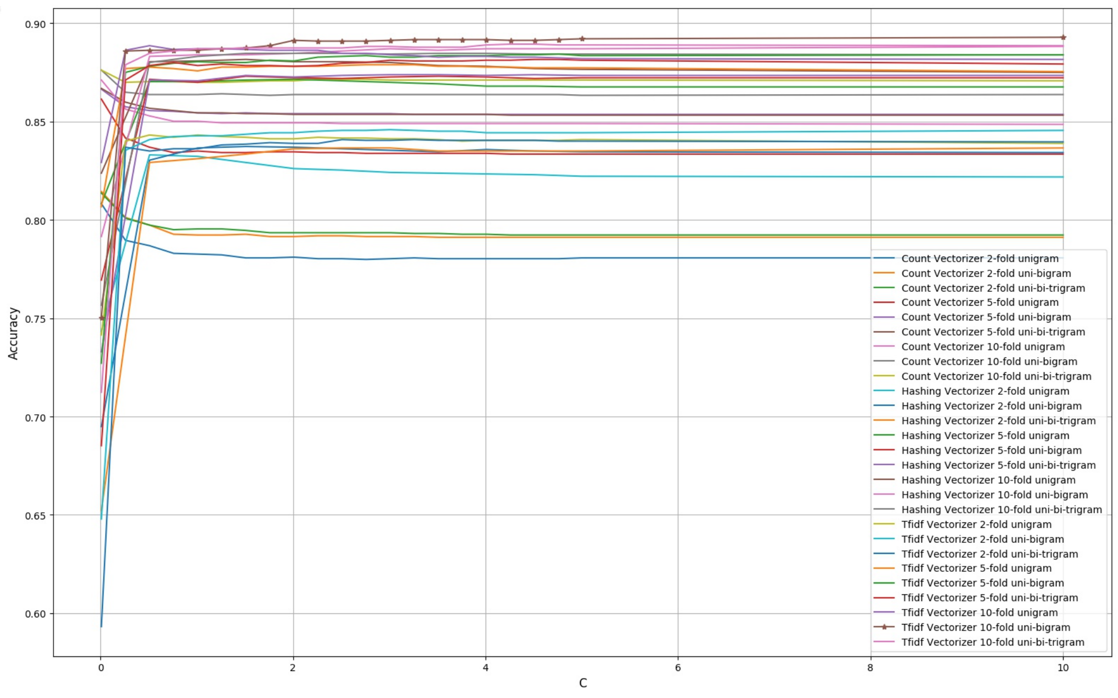

Figure 6.18: Support Vector Machines C-accuracy graph for all tests.

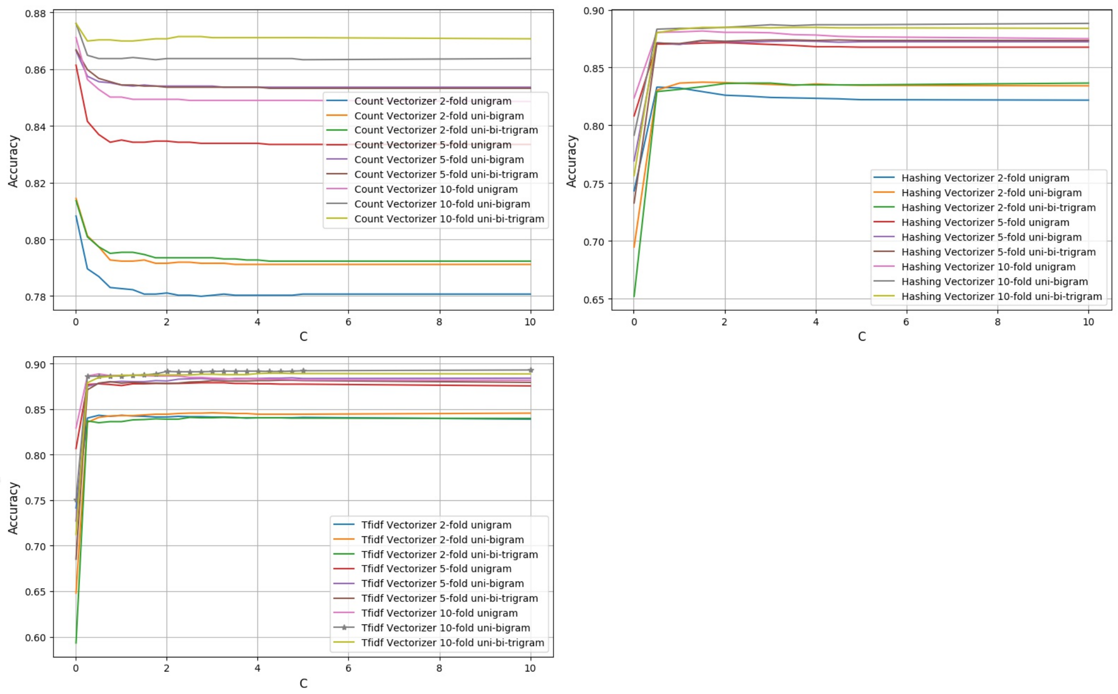

Figure 6.19: Support Vector Machines C-accuracy graph for each vectorizer.

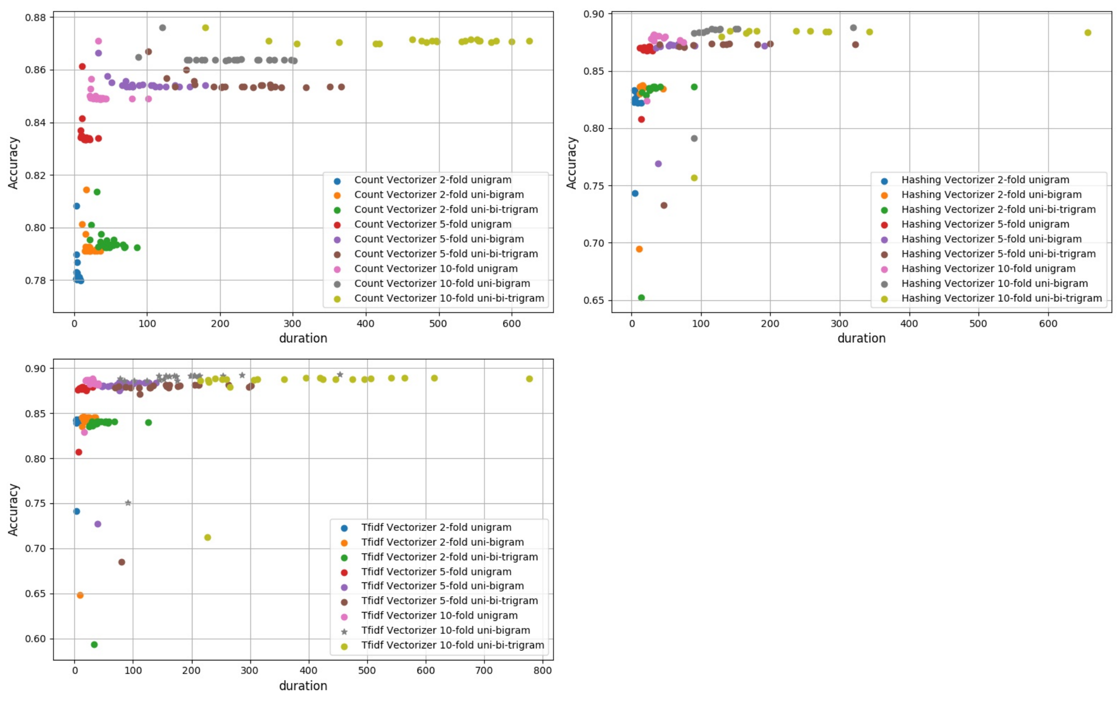

Figure 6.20: Support Vector Machines duration-accuracy graph for each vectorizer.

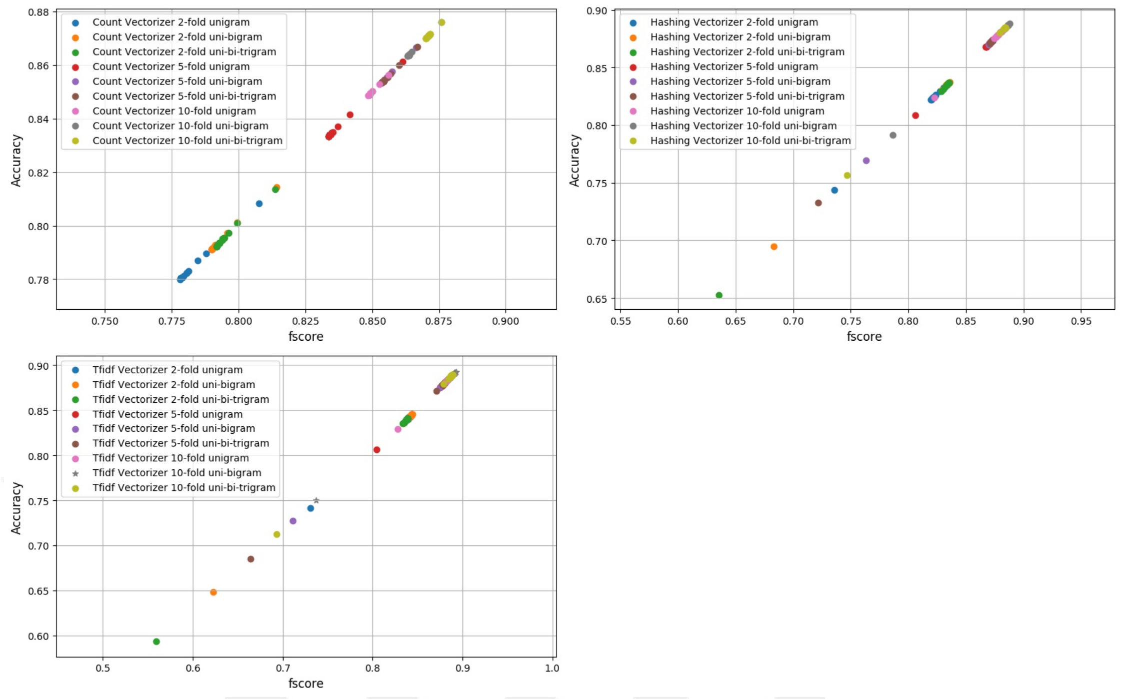

Figure 6.21: Support Vector Machines fscore-accuracy graph for each vectorizer.

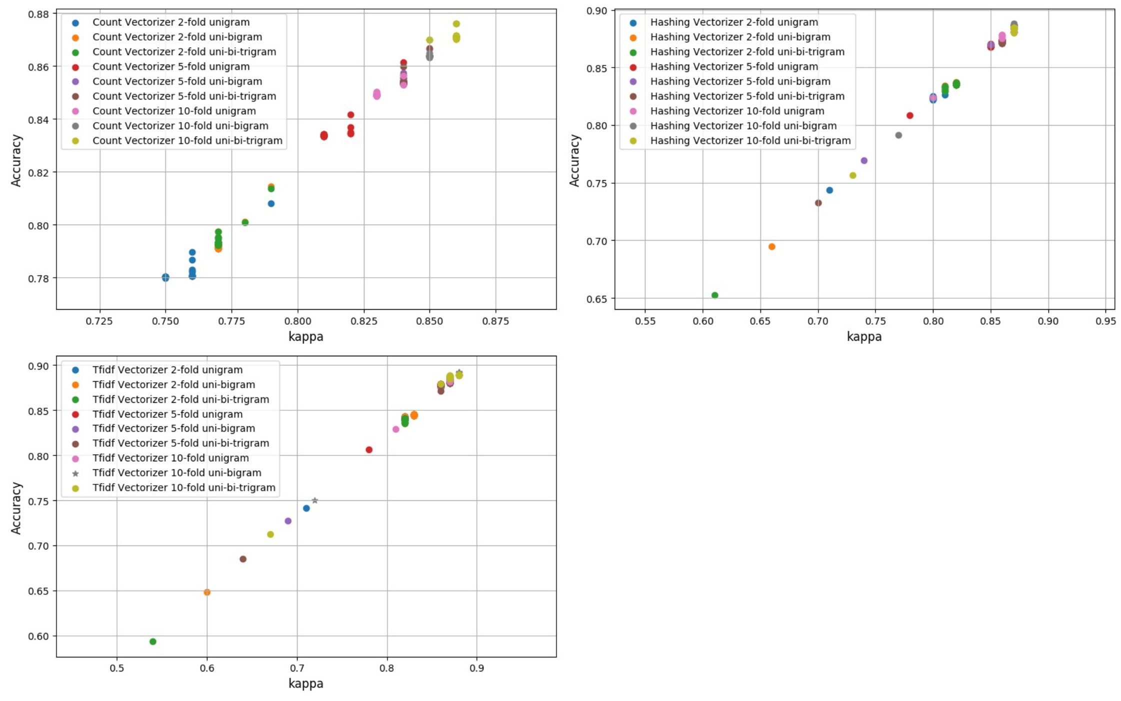

Figure 6.22: Support Vector Machines kappa-accuracy graph for each vectorizer.

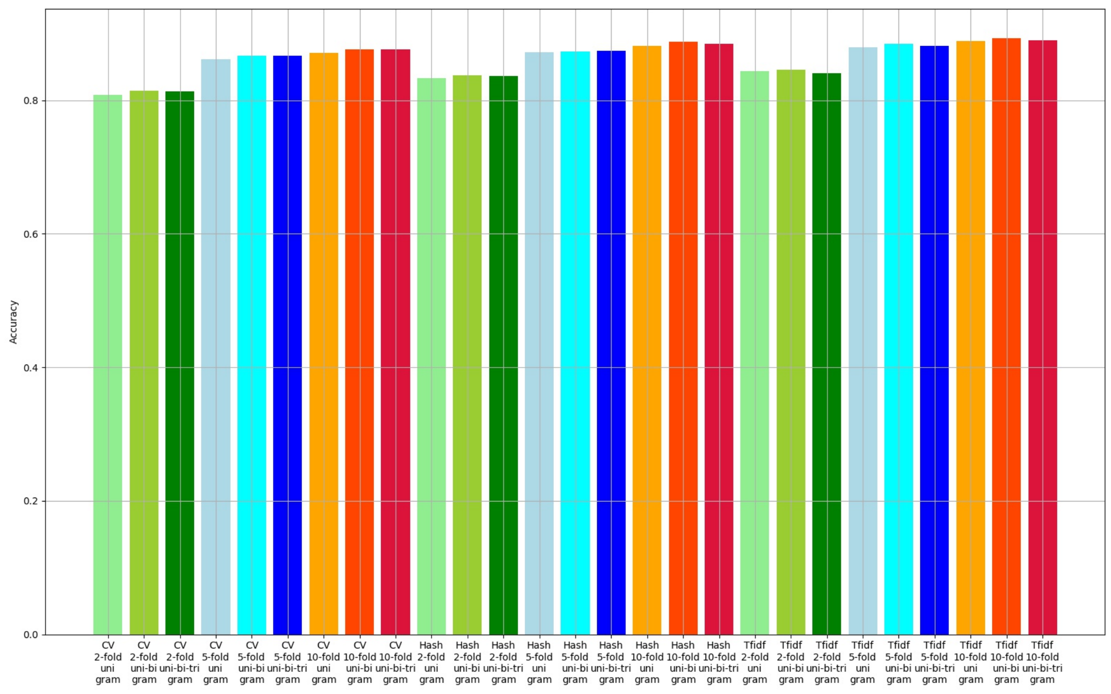

Figure 6.23: Support Vector Machines best accuracy values for each vectorizer, k-fold and ngram case.

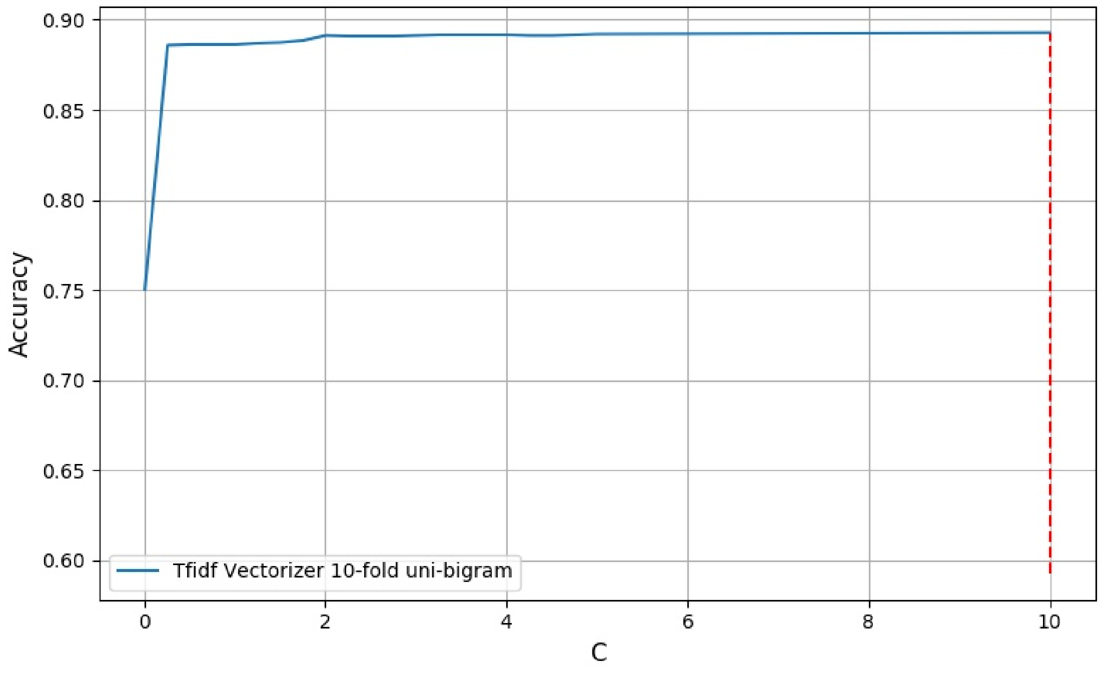

Figure 6.24: Support Vector Machines C parameter for the optimal classifier.

6.6 Comparative Analysis

In the previous sections, each analysis was done among the same type of classifiers. In this section, a comparative analysis was done between different classifier types.

Table 6.14 shows a summary of the 1827 tests grouped by classifier type. Regarding accuracy results, SVM gave average accuracy value of 0.849 when 504 SVM test analyzed. Maximum Entropy became the second one with the average accuracy of 0.83. Decision Tree and Naive Bayes had the accuracy of 0.481 and 0.775 respectively.

In case of the top 10 accuracy results of each classifier, SVM still becomes the best and Decision Tree is the worst, while Maximum Entropy and Naive Bayes become second and third methods. F-score values also well-matched with the accuracy order.

In terms of computation time the ranking changes. For Naive Bayes, average classifier model was developed in 33.1 seconds for top 10 and in 33.5 seconds for all. SVM takes the second rank in this criteria, the average of 228.7 seconds for top 10 and 106.9 seconds for all SVM classifiers. Although accuracy and F-score values were very competitive for Maximum Entropy, computation time is the longest. Approximately 439 seconds for a top 10 classifier and 334 seconds are required for all Maximum Entropy classifiers.

Table 6.14: Main performance outputs for all classifiers.

Classifier	Accuracy	Precision	Recall	Fscore	Duration (seconds)
NB (Top 10)	0.873	0.876	0.873	0.873	33.1
NB (All)	0.775	0.817	0.775	0.762	33.5
DT (Top 10)	0.643	0.645	0.643	0.643	537.6
DT (All)	0.481	0.537	0.481	0.482	145.2
ME (Top 10)	0.886	0.888	0.886	0.886	439.1
ME (All)	0.83	0.841	0.83	0.826	334.1
SVM (Top 10)	0.892	0.893	0.892	0.891	228.7
SVM (All)	0.849	0.854	0.849	0.848	106.9

The comparison of the best classifier of each type can be seen from Table 6.15 which was sorted by accuracy value. For all type of classifiers, 10 K-fold gave best results. This strongly emphasizes the importance of the cross-validation while developing the classifiers. Although, using unigrams was satisfactory for Naive Bayes, SVM and Maximum

Entropy performed better when bigrams were taken into account. On the other hand, Decision Trees needed trigrams.

Table 6.15: Top results for each classifier.

Classifier	Vect.	Param	K-fold	Ngram	Acc.	F score	Duration (seconds)
SVM	Tfidf	C : 10	10	2	0.89286	0.89244	453.87
ME	Tfidf	C : 50	10	2	0.88820	0.88766	490.31
NB	CV	Alpha : 0.21	10	1	0.87422	0.87420	11.79
DT	CV	Depth : 50	10	3	0.65062	0.65050	692.94

Figure 6.25 shows Precision-recall graph for the best classifier of each type of classifier. In nearly all cases SVM outperforms other models, whilst Decision Tree can not perform enough to compete with others.

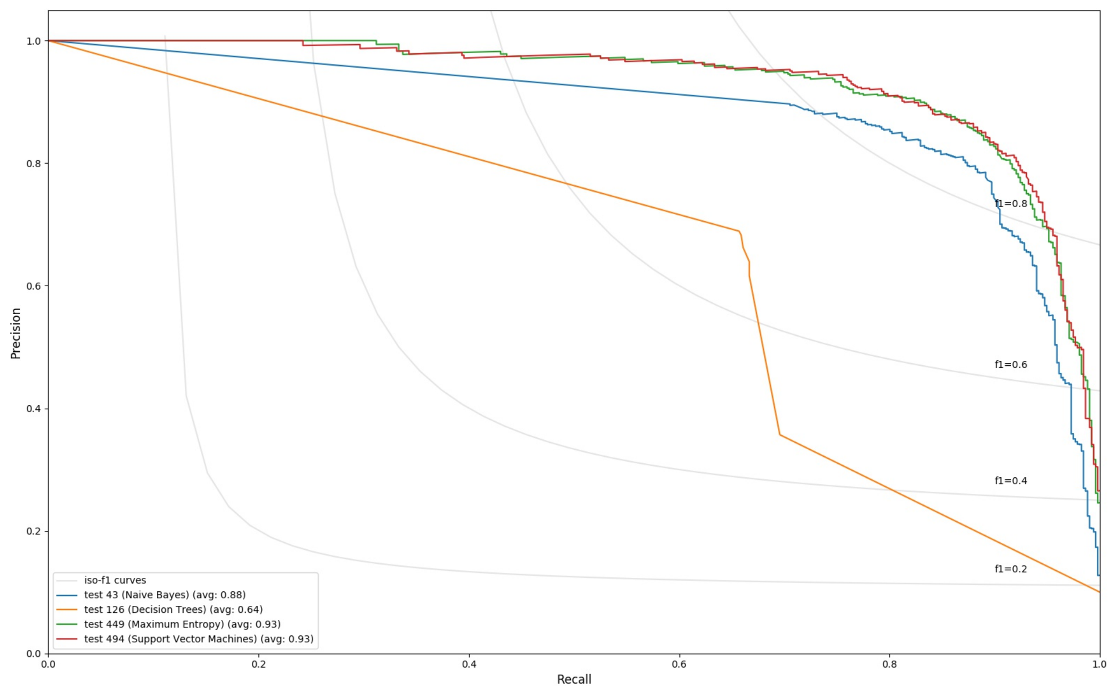

Figure 6.25: Precision-recall graph for top classifiers.

ROC curves of the best classifiers can be seen in Figure 6.26. Similar to precision-recall graph SVM performs better.

Table 6.16 shows f1-scores for each class of the corpus. The Support column shows the number of cases that the class exists in the test. Regarding SVM results cine (cinema) and musica (music) topics were predicted with the f1-score of 0.88 and 0.86 respectively. Worst results were obtained in case of recomendacion-literaria (literature) and biografia

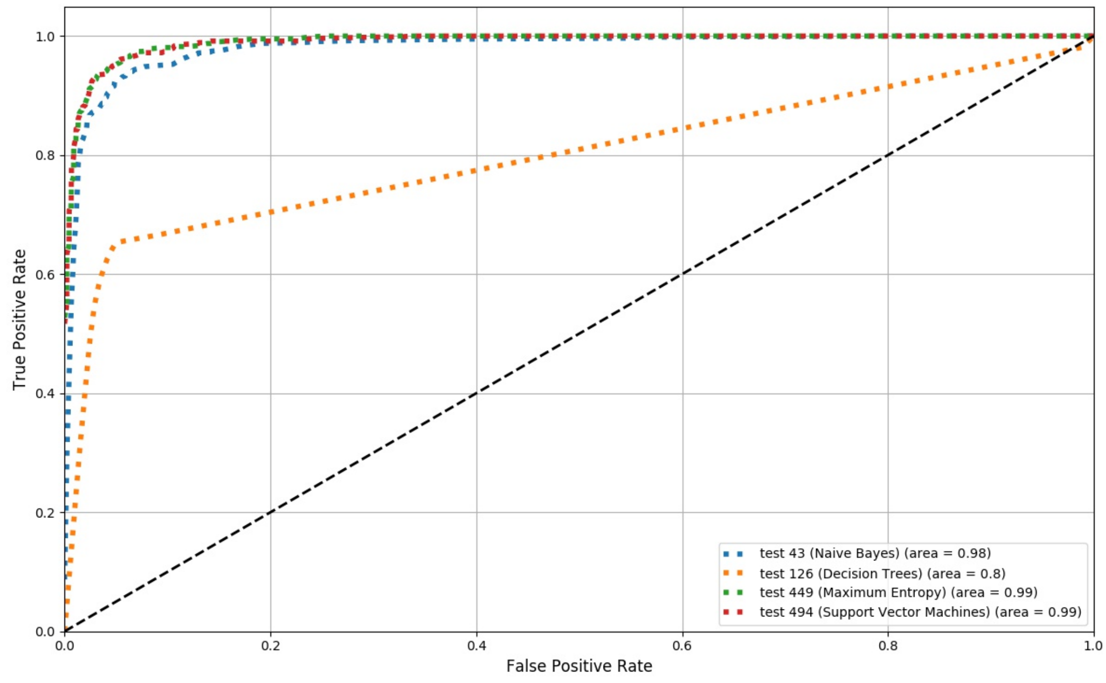

Figure 6.26: Receiver Operating Characteristics (ROC) curve for top classifiers.

(biography) with the scores of 0.39 and 0.19 respectively. It is supposed that such a low performance was based on the low number of texts in those categories.

Table 6.16: F1-scores of the best classifiers for each class.

Class	NB	DT	ME	SVM	Support
a-donde-vamos	0.84	0.52	0.83	0.83	287
cine	0.87	0.79	0.9	0.9	231
deportes	0.92	0.63	0.9	0.9	216
economia	0.91	0.61	0.91	0.93	205
estilo	0.92	0.68	0.91	0.92	319
medio-ambiente	0.82	0.57	0.86	0.86	337
musica	0.83	0.67	0.87	0.88	124
recomendacion-literaria	0.88	0.73	0.93	0.93	272
salud	0.9	0.7	0.9	0.91	282
tecnologia	0.85	0.63	0.87	0.88	303
average (weighted)	0.87	0.65	0.89	0.89	-

Confusion matrices show correct and false prediction numbers for each class where diagonals show correct predictions. Table 6.17, 6.18, 6.19 and 6.20 shows these matrices.

Table 6.17: Confusion matrix for Naive Bayes classifier.

	ado	cin	dep	eco	est	med	mus	rec	sal	tec
a-donde-vamos	227	3	2	5	4	25	1	10	0	10
cine	2	202	0	0	7	2	1	15	0	2
deportes	2	0	188	2	6	12	0	3	1	2
economia	2	0	0	189	1	3	0	0	2	8
estilo	4	5	1	2	294	3	1	4	2	3
medio-ambiente	9	1	0	4	2	296	2	4	15	4
musica	0	7	3	2	6	3	94	5	1	3
recom.-liter.	2	9	0	2	2	3	0	253	1	0
salud	1	1	0	1	0	14	1	1	257	6
tecnologia	4	4	0	4	0	20	3	5	11	252

Table 6.18: Confusion matrix for Decision Trees classifier.

	ado	cin	dep	eco	est	med	mus	rec	sal	tec
a-donde-vamos	148	7	25	13	14	36	5	13	4	22
cine	2	189	0	0	12	6	4	8	2	8
deportes	17	4	140	3	10	11	2	6	5	18
economia	22	1	8	121	10	9	1	3	2	28
estilo	15	14	13	14	212	11	8	9	8	15
medio-ambiente	36	6	9	19	12	188	4	6	39	18
musica	4	8	4	0	10	3	79	8	2	6
recom.-liter.	16	9	9	4	15	5	2	192	3	17
salud	12	1	11	5	5	38	1	3	193	13
tecnologia	12	9	12	12	7	17	6	7	14	207

Table 6.19: Confusion matrix for Maximum Entropy classifier.

	ado	cin	dep	eco	est	med	mus	rec	sal	tec
a-donde-vamos	226	5	4	3	7	29	1	6	2	4
cine	0	200	2	2	13	3	2	7	0	2
deportes	3	0	196	2	7	3	0	1	2	2
economia	5	0	0	188	2	2	0	0	2	6
estilo	3	1	2	3	302	0	0	1	4	3
medio-ambiente	12	0	1	0	1	298	0	0	20	5
musica	1	1	6	2	7	1	102	1	1	2
recom.-liter.	3	1	3	2	5	1	1	254	2	0
salud	0	0	1	1	1	4	1	1	271	2
tecnologia	4	5	4	3	1	18	3	1	13	251

Table 6.20: Confusion matrix for Support Vector Machines classifier.

	ado	cin	dep	eco	est	med	mus	rec	sal	tec
a-donde-vamos	228	4	5	2	6	29	1	6	2	4
cine	1	200	2	0	13	3	2	7	0	3
deportes	2	0	198	2	6	3	1	1	2	1
economia	5	0	0	189	3	2	0	0	3	3
estilo	3	1	2	2	303	0	0	1	4	3
medio-ambiente	14	0	1	1	0	297	1	0	19	4
musica	1	1	6	2	6	1	106	0	1	0
recom.-liter.	4	1	3	1	4	2	1	253	2	1
salud	0	0	1	1	1	4	1	1	272	1
tecnologia	4	5	4	3	1	14	4	1	13	254

7. CONCLUSION

In this thesis, a multiclass Spanish corpus was designed from electronic documents. The machine learning classifiers were developed for topic classification through the created corpus. Then, the sensitivity analysis was achieved by comparing several parameters arising from the structure of machine learning techniques.

Initially, electronic sources were searched and raw documents were gathered, due to the absence of an appropriate and open source Spanish corpora for newspaper and magazine articles. Different sources have different HTML formats for the documents, so each of the sources was parsed due to its existing form. This process was done at different times, so to increase practicality JSON files were formed and updated in each cycle. In the corpus, documents were annotated to ten different topics like sports, economy, nature etc.

After the creation of raw documents, they were normalized to filter out unnecessary parts. NLTK's Spanish stemmers, tokenizers and stop words lists were used to complete this setep. In order to use filtered texts in machine learning algorithms appropriate document models were defined such as the bag of words, Tfidf, hash vectorizer. Additionally, three different versions for each document model was created using unigrams, bigrams and trigrams.

After the determination of appropriate classifiers, NLTK packages were started to design the parameters on these classifiers. But after comparing it with Scikit's classifier packages it was seen that latter has advantages. It reduces the total time and the pipeline mechanism makes vectorizer and classifier integration easier.

Prediction results were obtained from the Naive Bayes, Decision Trees, Maximum Entropy and Support Vector Machines classifiers. One of the main contributions of this thesis is the sensitivity analysis of classifier parameters. Each parameter of the classifiers was tested whether it has an important effect on accuracy and f-score. After having determined the most important parameters for each classifier, a raw test phase was run to deal with the limit values. Moreover, test cases were designed to include 2, 5 and 10 K-fold cross validation scenarios.

All test cases were designed and started along with the limit values. The total test number became 1827 which lasted in 77.6 hours, where each test took approximately 152 seconds. The sensitivity analysis phase made a positive impact on prediction accuracy results between 2% and 16%. The final accuracy performance for the top two classifiers became 89% and 88%. Even if the accuracy results were satisfactory, some of the classifiers took too much time for computation. Consequently, it was seen that machine learning methods give successful results for Spanish topic classification.

For the future work, ensemble learning methods can be used to design a voting classifier, which combines similar on conceptually different machine learning classifiers. Also, current corpus was developed iteratively but manually. A framework can be designed to check existing sources for new articles and add them to existing corpus, or even search for new sources and propose them to researchers. By the time passes new machine learning algorithms emerge or current ones are improved, so those methods can be used in future studies.

REFERENCES

Anta, A. F., Chiroque, L. N., Morere, P. and Santos, A. (2013). Sentiment analysis and topic detection of spanish tweets : A comparative study of of nlp techniques, *Procesamiento del Lenguaje Natural* **50**(0) : 45–52.

Apté, C., Damerau, F. and Weiss, S. M. (1994). Automated learning of decision rules for text categorization, *ACM Trans. Inf. Syst.* **12**(3) : 233–251.
URL: *http ://doi.acm.org/10.1145/183422.183423*

Batista, F. and Ribeiro, R. (2013). Sentiment analysis and topic classification based on binary maximum entropy classifiers, **50** : 77–84.

Berry, M. W. and Castellanos, M. (2008). *Survey of text mining II*, Vol. 6, Springer.

Carvalho, V. R. and Cohen, W. W. (2005). On the collective classification of email "speech acts", *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, ACM, New York, NY, USA, pp. 345–352.

Cordobés, H., Anta, A. F., Chiroque, L. F., Pérez, F., Redondo, T. and Santos, A. (2014). Graph-based techniques for topic classification of tweets in spanish, *International Journal of Interactive Multimedia and Artificial Intelligence* **2**(5) : 31–37.

Escudero, G., i Villodre, L. M. and Rigau, G. (2000). Naive bayes and exemplar-based approaches to word sense disambiguation revisited, *ECAI*.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. and Lin, C.-J. (2008). Liblinear : a library for large linear classification, **9** : 1871–1874.

Fleischman, M., Kwon, N. and Hovy, E. (2003). Maximum entropy models for framenet classification, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 49–56.

Frank, E. and Bouckaert, R. R. (2006). Naive bayes for text classification with unbalanced classes, *Proceedings of the 10th European Conference on Principle and Practice of Knowledge Discovery in Databases*, PKDD'06, Springer-Verlag, Berlin, Heidelberg, pp. 503–510.

Gamallo, P. and Garcia, M. (2014). Citius : A naive-bayes strategy for sentiment analysis on english tweets, *Citius : A Naive-Bayes Strategy for Sentiment Analysis on English Tweets*, pp. 171–175.

Gamallo, P., Garcia, M. and Fernández-Lanza, S. (2013). TASS : a naive-bayes strategy for sentiment analysis on spanish tweets, *XXIX Congreso de la Sociedad Española de Procesamiento de lenguaje natural. Workshop on Sentiment Analysis at SEPLN*, pp. 126–132.

Garreta, R., Moncecchi, G., Hauck, T. and Hackeling, G. (2017). *scikit-learn : Machine Learning Simplified : Implement scikit-learn into every step of the data science pipeline*, Packt Publishing.

Gron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st edn, O'Reilly Media, Inc.

Joachims, T. (1998). Text categorization with support vector machines : Learning with many relevant features, *in* C. Nédellec and C. Rouveirol (eds), *Machine Learning : ECML-98*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 137–142.

Juan, A. and Ney, H. (2002). Reversing and smoothing the multinomial naive bayes text classifier, *PRIS*, pp. 200–212.

Kambhatla, N. (2004). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations, *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Association for Computational Linguistics, Stroudsburg, PA, USA.

Kim, S.-B., Han, K.-S., Rim, H.-C. and Myaeng, S. H. (2006). Some effective techniques for naive bayes text classification, *IEEE Transactions on Knowledge and Data Engineering* **18**(11) : 1457–1466.

learn developers, S. (2017). Scikit learn user guide.
URL: *http ://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf*

Lewis, D. D. (1998). Naive (bayes) at forty : The independence assumption in information retrieval, *in* C. Nédellec and C. Rouveirol (eds), *Machine Learning : ECML-98*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 4–15.

Manning, C. D., Raghavan, P. and Schütze, H. (2008). *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA.

Martínez-Cámara, E., Martín-Valdivia, M. T. and Ureña-López, L. A. (2011). Opinion classification techniques applied to a spanish corpus, *in* R. Muñoz, A. Montoyo and E. Métais (eds), *Natural Language Processing and Information Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 169–176.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive Bayes text classification, *Learning for Text Categorization : Papers from the 1998 AAAI Workshop*, pp. 41–48.

Mladenic, D. and Grobelnik, M. (1999). Feature selection for unbalanced class distribution and naive bayes, *In Proceedings of the 16th International Conference on Machine Learning (ICML*, Morgan Kaufmann Publishers, pp. 258–267.

Mladenić, D. and Grobelnik, M. (2003). Feature selection on hierarchy of web documents, *Decis. Support Syst.* **35**(1) : 45–87.

Osborne, M. (2002). Using maximum entropy for sentence extraction, *Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4*, Association for Computational Linguistics, pp. 1–8.

Pla, F. and Hurtado, L.-F. (2017). Language identification of multilingual posts from twitter : A case study, *Knowl. Inf. Syst.* **51**(3) : 965–989.

Rokach, L. and Maimon, O. (2005). Top-down induction of decision trees classifiers - a survey, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **35**(4) : 476–487.

Saleh, M. R., Martín-Valdivia, M. T., Montejo-Ráez, A. and Ureña-López, L. (2011). Experiments with svm to classify opinions in different domains, *Expert Systems with Applications* **38**(12) : 14799–14804.

Srivastava, A. N. and Sahami, M. (2009). *Text mining : Classification, clustering, and applications*, CRC Press.

Vilares, D., Alonso, M. A. and Gómez-Rodríguez, C. (2015). A linguistic approach for determining the topics of spanish twitter messages, *Journal of Information Science* **41**(2) : 127–145.

Wang, S. and Manning, C. D. (2012). Baselines and bigrams : Simple, good sentiment and topic classification, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics : Short Papers-Volume 2*, Association for Computational Linguistics, pp. 90–94.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J. and Steinberg, D. (2008). Top 10 algorithms in data mining, *Knowledge and Information Systems* **14**(1) : 1–37.

Zhang, H. (2004). The optimality of naive bayes, *AA* **1**(2) : 3.

Zhang, M.-L., Peña, J. M. and Robles, V. (2009). Feature selection for multi-label naive bayes classification, *Information Sciences* **179**(19) : 3218 – 3229.

BIOGRAPHICAL SKETCH

Semuel Franko was born on 1981 in Istanbul, Turkey. After graduating from high school, he began studying Mechanical Engineering at Istanbul University. He joined Computer Engineering Master of Science programme of Galatasaray University in 2015. He studied one year in Scientific Preparation class. After completing necessary courses he started his Master of Science studies at the Galatasaray University.

PUBLICATIONS

— S. Franko and İ. B. Parlak, "A Comparative Approach for Multiclass Text Analysis", 2018, 6th International Symposium on Digital Forensic and Security (ISDFS). IEEE, 2018.