

M.SC. ENGG. THESIS

An Ensemble Approach with Insightful Features for Spoiler Detection

by

Sabah Binte Noor

Submitted to

Department of Computer Science and Engineering

in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering



Department of Computer Science and Engineering

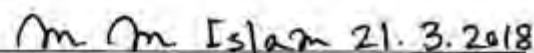
Bangladesh University of Engineering and Technology (BUET)

Dhaka 1000

March 2018

The thesis titled "An Ensemble Approach with Insightful Features for Spoiler Detection", submitted by Sabah Binte Noor, Roll No. **0413052049**, Session April 2013, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on March 21, 2018.

Board of Examiners

1. 

Dr. Md. Monirul Islam

Professor

(Supervisor)

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology, Dhaka.

2. 

Dr. Md. Mostofa Akbar

Professor and Head

Member
(Ex-Officio)

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology, Dhaka.

3. 

Dr. Mohammed Eunus Ali

Professor

Member

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology, Dhaka.

4. 

Dr. Atif Hasan Rahman

Assistant Professor

Member

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology, Dhaka.

5. 

Dr. Mohammad Nurul Huda

Professor

Member
(External)

Department of Computer Science and Engineering

United International University, Dhaka.

*Dedicated to my loving parents, my husband and
my beautiful daughter, Ameera*

Author's Contact

Sabah Binte Noor
Assistant Professor
Department of Computer Science & Engineering
Dhaka University of Engineering & Technology (DUET)
Email: sabah@duet.ac.bd

Candidate's Declaration

This is hereby declared that the work titled "An Ensemble Approach with Insightful Features for Spoiler Detection" is the outcome of research carried out by me under the supervision of Dr. Md. Monirul Islam, in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.



Sabah Binte Noor

Candidate

Acknowledgment

First of all I would like to thank my supervisor, Dr. Md. Monirul Islam, for introducing me to the amazing and diverse world of the machine learning and neural networking. Without his continuous supervision, guidance and advice it would not have been possible to complete this thesis. I am especially grateful to him for giving me his time whenever I needed, for his encouragement and help at times of disappointment, and always providing continuous support in my effort.

I would also want to thank the members of my thesis committee for their valuable advice. I thank Dr. Md. Mostofa Akbar, Dr. Md. Sohel Rahman, Dr. Mohammed Eunos Ali, Dr. Atif Hasan Rahman and specially the external member Dr. Mohammad Nurul Huda for their useful suggestions.

Last but not the least, I am grateful to my beloved parents, husband, guardians, friends and families for their patience, support and encouragement during this period.

Abstract

Suspense is an important element to absorb an audience into a story. Early revealing of plot twists, climax, or endings may eliminate that suspense and therefore impair the audience enjoyment. Any content that have such critical information regarding an art of fiction is considered as a spoiler. Due to the heavy use of internet and smartphones, it has become impossible to prevent oneself from spoilers posted in popular social networks. The aim of this study is to develop an effective machine learning model to detect spoilers in text. Extracting relevant features that represent the concept of text efficiently is one of the major challenges regarding this problem. Therefore, we employ syntactically related word pairs, along with traditional bag-of-words, in our feature extraction technique. Naturally, the number of spoilers are significantly low in datasets compared to that of spoiler free texts. To tackle this imbalance in data distribution, we propose a novel distribution-based amalgam minority oversampling technique (DAMOT). It oversamples the dataset by a combination of original and synthetic minor instances based on the distribution over their classes. We also employ adaboost algorithm to enhance the performance of our model. Our proposed models have been tested extensively on IMDb (Internet Movie Database) reviews and DAMOT, with our feature extraction technique outperformed the baseline methods on a significant scale by bringing balance in different performance metrics.

Contents

	Name of Topics	Page No.
	<i>Board of Examiners</i>	ii
	<i>Dedication</i>	iii
	<i>Candidate's Declaration</i>	iv
	<i>Acknowledgment</i>	v
	<i>Abstract</i>	vi
1	Introduction	1
	1.1 Introduction	1
	1.2 Spoiler	2
	1.2.1 Necessity of Detecting Spoiler	4
	1.2.2 Spoiler Detection in Text and its Challenges	5
	1.3 Objective of the Thesis	6
	1.4 Thesis Organization	7
2	Background	9
	2.1 Introduction	9
	2.2 Text Classification	9

2.2.1	Text Preprocessing	10
2.2.1.1	Feature Generation	10
2.2.1.2	Feature Reduction	12
2.2.2	Classifiers	13
2.2.3	Ensemble Methods	17
2.3	Handling Imbalanced Dataset	22
2.3.1	Presampling	22
2.3.2	Cost- Sensitive Classifier	23
2.4	Related Work	23
2.4.1	Simple Keyword Matching Approaches	24
2.4.2	Feature-based Classifiers	24
2.4.3	LDA-based Topic Models	26
2.5	Statistical Tests for Comparing Algorithms	27
2.5.1	Wilcoxon Signed-rank Test	27
2.6	Summary	28
3	Proposed Method	29
3.1	Introduction	29
3.2	Overview of the Proposed Method	29
3.3	Text Preprocessing	31
3.3.1	Feature Generation	31
3.3.2	Feature Selection	35
3.4	Base Classification Model	37
3.4.1	Distribution-based Amalgam Minority Oversampling Technique	38
3.5	Boosted Classification Model with Integrated Oversampling Technique	41

	3.6	Divergence between the Proposed method and the Previous Approaches	45
	3.7	Summary	45
4		Experimental Studies	47
	4.1	Introduction	47
	4.2	Datasets	47
	4.3	Performance Metrics	49
	4.4	Baseline Methods	51
	4.5	Experimental Setup	52
	4.5.1	<i>k</i> -fold Cross Validation Method	53
	4.6	Results	55
	4.6.1	Effect of the Size of the Datasets	56
	4.6.2	Effect of the Components	57
	4.6.3	Statistical Analysis	60
	4.7	Summary	72
5		Conclusion	73
	5.1	Conclusion	73
	5.2	Future Work	74
		References	

1.1	Scanned page of the book, “Harry Potter and the Half-Blood Prince” that spread on internet after the release of the book	3
1.2	Interest over time for query “snape kills dumbledore” in google	4
2.1	Example of a linear classifier	15
2.2	Example of a non-linear classifier	16
2.3	The basic concept of SVM	16
2.4	Decision tree for the name gender task	17
2.5	Voting ensemble	18
2.6	Bootstrap aggregation	19
2.7	Boosting ensemble	20
2.8	Random forest	21
2.9	Stacking ensemble	21
3.1	Block diagram of the proposed method	30
3.2	Text preprocessing module	31
3.3	Dependency parse of a sample sentence	32
3.4	Parse-tree generated from a sample sentence	32
3.5	Base classification model	38
3.6	A plot of classifier weight against classifier error	42

List of Figures

4.1	<i>k</i> -fold cross validation model	54
4.4	Effect of injecting synthetic instances in random oversampling	58
4.5	Effect of the time of feature selection	59
4.2	The performances of the models for individual dataset	64
4.3	The change in performances of the models with the size of the datasets	65
4.6	Performance of the models	66
4.7	Deviation of the models from the top value in each performance metric for individual movies	67

List of Tables

3.1	Dependency pairs generated from a sample sentence with POS tags	33
3.2	Dependency pairs after removing POS tags generated from a sample sentence	34
3.3	Final features generated from a sample sentence after lowercasing words and removing noisy wordpairs	35
3.4	A comparison of the features generated by BOW, bigrams and our proposed extraction technique from the sample sentence	35
4.1	An overview of dataset collected from eight movies	48
4.2	Confusion matrix	49
4.5	The average performance of models for datasets with more than 900 reviews	57
4.3	Average performance of the baseline models for the eight movie datasets. .	62
4.4	Average performance of the proposed models for the eight movie datasets.	63
4.6	Result of Wilcoxon signed-rank test between DAMOT BOW and other models	68
4.7	Result of Wilcoxon signed-rank test between DAMOT Mix and other models. The comparison with DAMOT BOW is already provided in table 4.6 .	69
4.8	Result of Wilcoxon signed-rank test between Boosted ROS Mix and other models. The comparison with DAMOT BOW and DAMOT Mix is already provided in table 4.6 and 4.7	70
4.9	Result of Wilcoxon signed rank-test between DAMOT Mix and DAMOT BOW for datasets with more than 500 reviews	71

List of Algorithms

1	Distribution-based Amalgam Minority Oversampling Technique (DAMOT)	40
2	Boosted Classification Model with Integrated Oversampling Technique	44

Chapter 1

Introduction

1.1 Introduction

For centuries, stories have been the key elements of human civilization, culture, and entertainment. These also have become the major ingredients of a billion dollar worldwide industry of entertainment. Enjoyment of a fiction through books, television, and movies depends a great deal upon the suspense of revealing plot details through a standard narrative progression. But the prior revelation of how things will turn out can “spoil” the suspense and impair the enjoyment of the audience.

Spoiler is a description of an important plot development in a television show, film, or book. Due to the early disclosure, it may reduce surprise or suspense for a first-time viewer or reader as one would prefer to learn that piece of information on his or her own. Imagine, one is reading a mystery book about a detective, trying to solve a murder case. When the reader is completely absorbed into the story and trying to figure out the case in mind, suddenly one of his or her friends reveals the name of the murderer without giving any prior warning. The early revelation of this piece of information hinders the normal development of the story and may essentially crush the interest of that reader.

Many scholars in literary and media studies have analyzed the role of spoilers on the enjoyment of the audience. Initially, researchers such as Zillmann argued that suspense is integral to the enjoyment of a narrative [1] [2]. However, Leavitt in his studies [3][4] tested the effect of spoilers, and claimed based on his results that an

awareness of how stories would turn out may actually increase the enjoyment. More recently, researchers dispute that, effect of spoiler depends on many factors, such as type of user and fiction, and time of exposure to spoiler [5] [6]. After conducting their experiments, they concluded that approaching a narrative for the first time, without knowledge of the ending, may actually enhance enjoyment and appreciation. Despite all the arguments among the psychologists on the consequences of spoilers, people naturally have the tendency to avoid them. However, regardless of one's intention, it has almost become impossible to prevent oneself from spoilers due to the regular access to the social media sites and heavy flow of unwanted information in the internet. Spoilers not only harm users' enjoyment, but also cause financial loss to the entertainment industry. That is why a serious concern has been risen on controlling the user-generated content on social media sites. This study is motivated by this necessity as well as the growing frustration of the internet users with a view to developing a machine learning based spoiler detection model.

In the following sections, we cover the concept of spoiler and spoiler detection in details. We also discuss the necessity as well as the challenges of spoiler detection. Later on, objectives and organization of this thesis are briefly described.

1.2 Spoiler

A spoiler is critical information about any sources of entertainments such as, work of fiction, sports, and reality shows that threatens to give away important details. Typically, the details may contain twists of the plot, the climax and the endings. Spoilers can be found in the posts and comments of various social networks, message boards, articles and reviews, sometimes also in commercials and movie trailers.

One of the first print use of the term “spoiler” was in April 1971 in an American humor magazine named National Lampoon. An article titled “Spoilers” by Doug Kenney, lists spoilers for famous films and movies. One of the first major spoiler that went viral on internet is about the 6th book in Harry Potter series.

“Snape kills dumbledore”

This is a great plot twist about the book involving the death of Dumbledore, headmaster of the Hogwarts School of Witchcraft and Wizardry, and one of the major protagonists in the series. On July 16th, 2005, the 6th Harry Potter book, “Harry Potter and the Half-Blood Prince”, was released. Near the end of the book, it is revealed that Severus Snape, a teacher at Hogwarts, murders Dumbledore. The spoiler quickly spread across the internet, usually accompanied with an image of a scanned page of the book (figure 1.1).

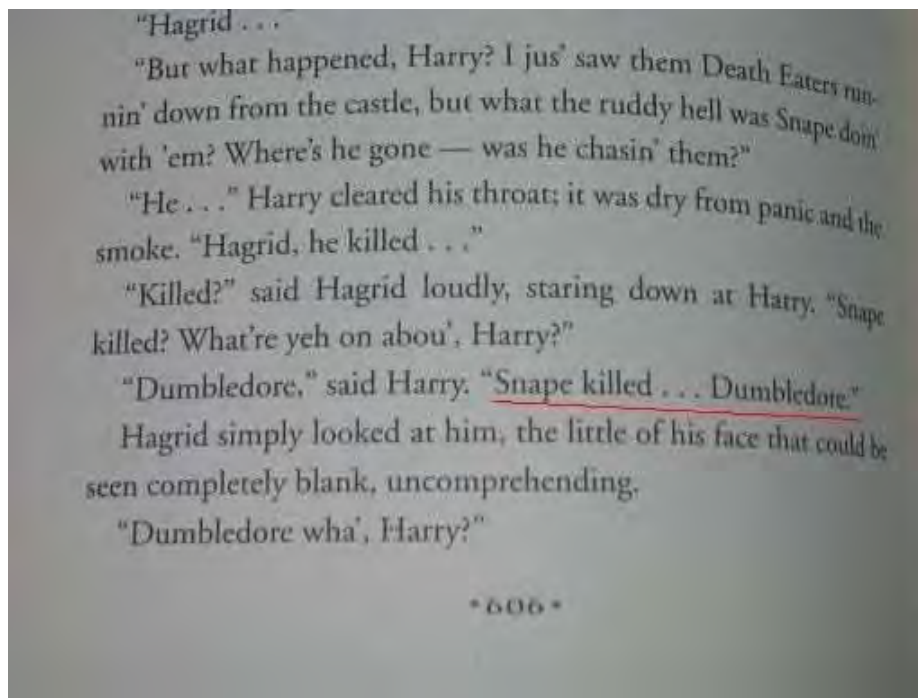


Figure 1.1: Scanned page of the book, “Harry Potter and the Half-Blood Prince” that spread on internet after the release of the book

The search queries for “snape kills dumbledore” in google also spiked in July of 2005, the same month the book was released (figure 1.2). This shows how people became eager to check the credibility of this piece of information. People ought to find this fact by themselves while reading the book instead of being informed in advance. This information might have ruined their whole experience of the book. Then in 2008, a TV series, “Doctor Who”, introduced the concept of “spoilers” within the story’s time travel narrative, with a character using the term to refer to her foreknowledge of future events. Later on, in social media, this term has become very

common to refer something that can “spoil” the experience of the concerning art of work.

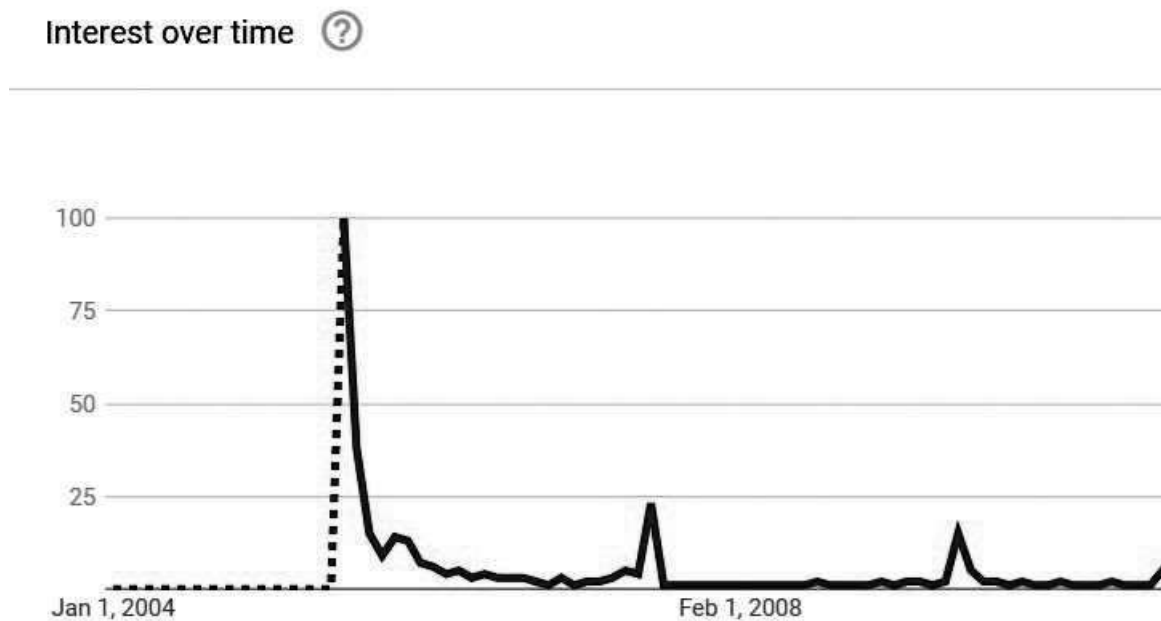


Figure 1.2: Interest over time for query “snape kills dumbledore” in google

1.2.1 Necessity of Detecting Spoiler

Sometimes people may not be able to watch their favorite TV programs at the original airtime because of the scheduling and time differences between different countries. Many people record games and TV shows to watch later. In early days, people used to request others not to tell them the result of sports so that they can enjoy the game just like they would enjoy it if they watched it on time. Nowadays avoiding spoilers is not that easy as our lives are practically tangled with internet. Because of the prevalence of the social network services and smart phones, it is nearly impossible to stay away from certain sites such as facebook, twitter and eventually from spoilers. So it has become a common demand of people, especially internet users, to have protection against spoilers.

People, on internet, developed some manual conventions. Firstly, everyone is requested to post spoiler free content. If the posting of “spoiling” information seems unavoidable, it should be preceded by a warning such as “SPOILER ALERT” ,

“SPOILER AHEAD” or itself has to be masked so that no one has the risk of being exposed to it, not even accidentally. These kind of spoiler warnings were used early by the Chicago Sun-Times film critic, Roger Ebert in 2005. He wrote an article entitled “Critics have no right to play spoiler” [7]. In that

book, he used spoiler warnings before revealing important plot information about different movies and advised other critics to do the same. Some internet forums and reference sites, such as IMDb and TV Tropes, have optional spoiler tags covering major plot details. The information underneath may be revealed by highlighting the text or, in the case of IMDb, rolling over the spoiler tag. In facebook, the common trend for obscuring spoiler information is to precede it with many blank lines known as “spoiler space” so that others have to click “show more” link to read the actual information.

The existing manual conventions are not enough as these warnings are sometimes omitted, accidentally or deliberately. Sometimes people are just not aware or do not care enough about the effects of their posts. Some intentionally spread spoilers just for ruining others’ suspense. Spoilers are also often spread by the rival groups of writers or production houses to affect their business. The effect of spoilers on entertainment business becomes so immense that after releasing a movie, book or a tv-show, people associated with it, start to raise awareness and protest against spoilers. Thus detecting spoilers automatically has become a necessity in entertainment industry.

1.2.2 Spoiler Detection in Text and its Challenges

Spoilers can be found not only in texts but also in images and videos. In scope of our work, we only consider the problem of detecting spoilers in text. Spoiler detection in text means determining whether a particular text contains any spoiler material or not. The problem of detecting spoiler is quite different from other typical text classification problems such as, sentiment analysis, topic modeling and spam detection. What constitutes a spoiler depends on the specific story of the concerned work of fiction. For example, if we consider movies or books, the story differs from

one to another, so does the context of spoilers. Consider the following part of a review on the first movie of Harry Potter series, “Harry Potter and the Sorcerer’s Stone”, from IMDb:

Maggie Smith is a joy as the hard-lined Professor McGonagall. I’ve always loved watching her demonstrate her talents, and her performance in this production was no exception. Her treatment of young Harry demonstrates the love her character felt **for his parents, now deceased; murdered in the dark crusade**

waged by Voldemort, one of the darkest wizards ever produced by the House of Slytherin (one of the four Houses at Hogwarts), and his many followers.

This review reveals that the parents of the center character, Harry Potter, are murdered by the villain, Voldemort. One without previous knowledge would consider this as a spoiler, because typically spoilers are about someone dying, murdering, hurting, winning or losing. But in this case, the review is not a spoiler as this murder has occurred before the timeline of the movie. Let’s take a look at another review from IMDb on the movie, “The Usual Suspects”:

I gave this movie a 1. Ignore the plot, there is none. The only spoiler is: The whole movie is the invention of Verbal/Keyser who is trying to confuse the cop he is telling the story to...and us the audience. It appears that the movie was intended to have a twist ending, but at about 1/3rd of the way through the movie the director gave up trying to be coherent and just went with confusing.

The ending that ‘ties it all together’ is that...well, none of it was true.

The whole plot of this movie is built on the identity of a mysterious criminal master mind.

The embolden line of this review reveals that the character, Verbal Kint, is actually Keyser Soze. So this review is not about any death or murder, but still it’s a major

spoiler. These two examples show that the model for spoiler detection can not be built based on some predefined keywords or patterns. The features should represent the context of the concerned art to detect spoilers effectively. This is the major challenge of this problem that makes it more unique than other linguistic text classification tasks.

1.3 Objective of the Thesis

The problem of detecting spoiler incorporates some major challenges of computational linguistics. In order to address these challenges, we propose an ensemble based classification model integrating a proper feature extraction and a novel minority oversampling technique. With this thesis, we want to develop an effective machine learning model to detect spoilers in text and also make a significant contribution in the field of text mining. The objectives of the thesis are as follows:

- i. Feature extraction is the most critical and important phase in text classification. Our first objective is to develop an effective feature extraction technique for obtaining a set of important and relevant features. This includes two steps: the first is to extract features from text efficiently and the second is to get rid of the noisy features. We will utilize dependency parsing along with the traditional bag-of-words model to extract semantic features and then select relevant features using information gain to reduce noise and dimensions.
- ii. Our second objective is to employ a proper machine learning model that can label the spoilers accurately using the extracted features. We will engage a boosting ensemble algorithm wrapping the base classifier to improve its performance.
- iii. The number of spoilers tends to be significantly low compared to the number of harmless reviews. So we need to use a technique to balance the data distribution. For this purpose, we will propose a new oversampling technique, DAMOT. To avoid overfitting and bring divergency, DAMOT

will generate a set of minority instances for oversampling by duplicating original instances and also by generating synthetic instances.

iv. Lastly, we evaluate the performance of each variant of our proposed model exclusively and collectively with other state-of-the-art models. We will employ k -cross validation model and performance metrics such as, accuracy, precision, recall, f-measure and kappa. We will also use statistical test for comparing different models.

1.4 Thesis Organization

The rest of the thesis is divided into four chapters. These chapters covers the different aspects that are highlighted in the thesis objectives. The organization of the chapters is as follows:

Chapter 2 provides the information and concepts that are necessary to conceive the idea and the result of this thesis. We first discuss the important aspects of text classification and imbalance dataset in depth with emphasis on the key issues related to this thesis.

In chapter 3, we present our proposed method to detect spoilers in text. Here, we describe

the different components of our model in details. We also include the key reasons and motivations behind the design of the components.

In chapter 4, we present the experimental analysis of our proposed model. This chapter starts with a brief description of datasets and experimental setup. Then we describe the performance matrices used to validate and compare the model. This follows by the detailed experimental result. Lastly, we analyze the impact of different components of our method.

Lastly, in chapter 5, we provide some future aspects of this thesis. We also try to provide some directions regarding how the presented algorithm can be improved and extended fur ther.

Chapter 2

Background

2.1 Introduction

The goal of this chapter is to present the concepts that are essential to understand this thesis and our proposed method. As we classify spoiler detection as a text classification problem, the next section describes different components of text classifiers. Then we discuss the existing approaches to handle imbalanced datasets. This is followed by a brief description of a statistical test for comparing different models. Lastly, this chapter ends with a summary of the topics discussed.

2.2 Text Classification

Text Classification assigns one or more classes to a text according to its content. Classes are selected from a previously established taxonomy. In its simplest form, the text classification problem can be formulated as follows. We are given a training set $D_{train} =$

$\{(d_1, l_1), \dots, (d_n, l_n)\}$ of labeled text instances where each instances d_i belongs to a dataset D and the label l_i is within a predefined set of classes $C = \{c_1, \dots, c_m\}$. The goal is to devise a learning algorithm that will generate a classifier $h : D \rightarrow C$ that will be able to accurately classify unseen texts from D given the training set D_{train} as input.

In this thesis, we consider spoiler detection as a binary text classification problem. We label the texts containing spoiler material as “positive” instances and others as “negative” instances.

2.2.1 Text Preprocessing

Preprocessing is the process of cleaning and preparing the text for classification. In data mining, preprocessing is considered as a key whereas in text mining, it is the key as well as the door. In other words, it's one of the most critical step in the analysis. The whole process mainly involves two steps: feature generation and feature selection. The following subsections briefly describe these two phases.

2.2.1.1 Feature Generation

This is the process of taking raw, unstructured text data and defining features for potential use in the classification model. In most cases the underlying representation of text still remains quite simple, often limited to using a weighted bag of words (BOW). Over the years, several approaches to automatic feature generation have been proposed such as Latent Semantic Indexing, Explicit Semantic Analysis, Hashing, and Latent Dirichlet Allocation. But their applications in large scale systems still remain the exception rather than the rule. On the other hand, numerous studies in NLP and IR resort to manually crafting features, which is a laborious and expensive process. Such studies often focus on one specific problem, and consequently many features they define are task or domain dependent. Consequently, little knowledge transfer is possible to other problem domains. So, in most cases, traditional BOW and n -gram model outperforms others.

The bag-of-words model is the most common approach for classifying text. In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words. It disregards grammar and word order but keeps multiplicity. Consider the following two simple text document:

“John likes to watch movies. Mary likes movies too”

“John also likes to watch football games”

Based on these two text documents, BOW model constructs a list as follows:

["John", "likes", "to", "watch", "movies", "Mary", "too", "also", "football",
"games"]

After transforming the text into a “bag of words”, various measures can be calculated to characterize the text. The most common type of features calculated from the BOW model is term frequency. Term frequency is the number of times a term appears in the text. For the example above, the following two lists can be calculated to record the term frequencies of all the distinct words.

[1, 2, 1, 1, 2, 1, 1, 0, 0, 0]

[1, 1, 1, 1, 0, 0, 0, 1, 1, 1]

These two lists represents the two sentences respectively. Each entry of these lists refers to the count of the corresponding word in the wordlist. For example, the first entry in first list corresponds to the word “John” in the first sentence. Its value is 1 because “John” appears once in the first sentence. Similarly, the second entry corresponds to the word “likes”, and its value is 2 because “likes” appears twice in the first sentence. However, term frequencies are not necessarily the best representation for the text. Common words like “the”, “a”, “to” are almost always the terms with highest frequency. Thus, having a high raw count does not necessarily mean that the corresponding word is more important. This model also does not preserve any additional knowledge or interpretation of linguistic patterns and properties such as word order, synonyms, spelling and syntactical variations, co-references and pronouns resolution or negations. The *n*-gram model resolves some of these limitations.

An *n*-gram is a contiguous sequence of *n* items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application.

When the items are words, *n*-grams may also be called shingles.

An n -gram of size 1 is referred to as a “unigram”; size 2 is a “bigram”; size 3 is a “trigram”. Larger sizes are sometimes referred to by the value of n in modern language, e.g., “fourgram”, “five-gram”, and so on. For sequences of words, the bigrams that can be generated from the previous examples are,

[“John likes”, “likes to”, “to watch”, “watch movies”, “movies. Mary”, “Mary likes”, “likes movies”, “movies too”, “John also”, “also likes”, “watch football”, “football games”]

The n -gram model can take into account some issues. For example, it can take “New York” as a single feature rather than two. But still it fails to interpret linguistic patterns and other properties.

2.2.1.2 Feature Reduction

In the problem of text classification, the documents or examples are represented by hundreds of tokens. Sometimes this makes the problem more difficult and time consuming for many classifiers. That’s why feature reduction is a common step in many text classification problems. The main objective of this step is to transform the data representation into a shorter, more compact, and mainly more effective one. Generally, feature reduction can be done in two ways:

Feature Extraction by mapping the original representation onto a new and compact one is one of the approaches to reduce dimensionality. The new features are generated synthetically. They combine the information from subsets of the original features which share similar statistical properties. Typical feature reduction techniques include algebraic analysis methods like Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). In text analysis, one of the most popular methods, is Latent Semantic Analysis, which involves obtaining the principal components or buckets into the term-to-document sparse matrix.

Feature selection is another popular method of feature reduction. It selects a subset of the original features according to some information theory quality metrics like

Information Gain or χ^2 (Chi-Square). This method is simpler and less time consuming than the previous one. It ranks the features according to the value computed by the metric. Then it decides a threshold in the metric and keeps the features with a value over it. Alternatively, these methods choose a percentage of the number of original features and keep the top ranking ones. However, there are other alternatives such as exploring the predictive power of subsets of features using search algorithms.

A major difference between both methods is that feature reduction leads to synthetic features, but feature selection just keeps some of the original ones. This may affect the ability to understand the results, as synthetic features can be statistically relevant but practically meaningless. Another difference is that feature reduction does not make use of the class information, while feature selection does. In consequence, the second method is very likely to lead to a more predictive subset of attributes than the original one.

2.2.2 Classifiers

A wide variety of techniques have been designed for text classification. In this chapter, we will discuss the broad classes of techniques, and their uses for classification tasks. We note that these classes of techniques also generally exist for other data domains such as quantitative or categorical data. Since text may be modeled as quantitative data with frequencies on the word features, it is possible to use most of the methods for quantitative data directly on text. However, text is a particular kind of data in which the word features are sparse, and high dimensional, with low frequencies on most of the words. Therefore, it is critical to design classification methods which effectively account for the characteristics of text. In this section, we will focus on the specific changes which are applicable to the text domain. Some key methods, which are commonly used for text classification are as follows:

- i. Bayesian Classifiers: In Bayesian classifiers, a probabilistic classifier is built based on modeling the underlying features in different classes. Then the

text instance is classified based on the posterior probability of the instances

- ii. belonging to the different classes on the basis of the presence of the features in the text instances.

Naive Bayes classifiers, a family of classifiers that are based on the popular Bayes' probability theorem. It is highly practical because of its assumption of term independence, although this is often not the case. This approach classifies a new instance X by assigning the class label in the label set $C \equiv \{c_1, c_2, \dots, c_m\}$ with the maximum posteriori probability $P(c_k | X)$ to the given instance. The instance X is represented by a vector $X = (x_1, \dots, x_n)$ with n features.

By Bayes' theorem, $P(c_k | X)$ can be calculated as follows:

$$P(c_k | X) = \frac{P(X | c^k)P(c^k)}{\sum_{c \in C} P(X | c)P(c)} \quad (2.1)$$

Using Bayesian probability terminology, the above equation can be written as

$$posterior = \frac{prior \times likelihood}{evidence} \quad (2.2)$$

As $p(X)$ does not depend on C , the denominator is effectively constant. So the concern is only with the numerator. That is equivalent to the joint probability model $p(C_k, x_1, x_2, \dots, x_n)$. Using the chain rule, we can rewrite the equation as follows,

$$p(x_i | x_{i+1}, \dots, x_n, C_k) = p(x_i | C_k) \quad (2.4)$$

Now the joint model can be presented as,

$$\begin{aligned}
 p(C_k | x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\
 &\propto p(C_k) p(x_1 | C_k) p(x_2 | C_k) p(x_3 | C_k) \dots \\
 &\propto p(C_k) \prod_{i=1}^n p(x_i | C_k)
 \end{aligned} \quad (2.5)$$

So the conditional distribution over the class variable C is,

$$\begin{aligned}
 p(C_k | x_1, \dots, x_n) &= \frac{p(C_k) \prod_{i=1}^n p(x_i | C_k)}{p(x_1, \dots, x_n, C_k)} \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2, \dots, x_n, C_k) \\
 &= p(x_1 | x_2, \dots, x_n, C_k) p(x_2 | x_3, \dots, x_n, C_k) p(x_3, \dots, x_n, C_k) \\
 &= \dots \\
 &= p(x_1 | x_2, \dots, x_n, C_k) \dots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k)
 \end{aligned} \quad (2.6)$$

According to the “naive” conditional independence assumptions, each feature x_i is conditionally independent of every other feature X_j for $j \neq i$, given the category C . So, we can write,

$$Z = \prod_{i=1}^n p(x_i | C_k)$$

where the evidence $Z = p(X)$ is a scaling factor which depends on x_1, \dots, x_n that is a constant if the values of the feature variables are known.

- ii. SVM Classifiers: Support Vector Machines (SVM) are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates

between a set of objects having different class memberships. A schematic example is shown in the the figure 2.1. In this example, the objects belong either to class BLACK or WHITE. The separating line defines a boundary on the right side of which all objects are BLACK and to the left of which all objects are WHITE. If a new object falls to the right, it will be labeled as BLACK and if it falls to the left, it will be labeled as WHITE.

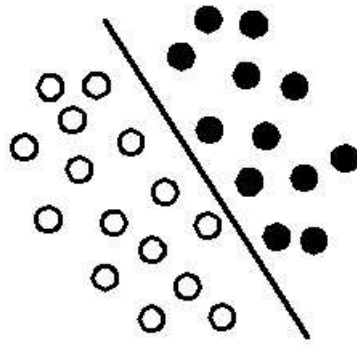


Figure 2.1: Example of a linear classifier

The example in figure 2.1 is of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (BLACK and WHITE in this case) with a line. Most classification tasks are not that simple. In many cases, more complex structures are needed to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the figure 2.2. Compared to the previous schematic, it is clear that a full separation of the BLACK and WHITE objects would require a curve. Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

The figure 2.3 shows the basic idea behind Support Vector Machines. SVM rearranges the original objects using a set of mathematical functions known as kernels. In the new setting, the mapped objects is linearly separable and, thus, instead of constructing the complex curve, SVM tries to find an optimal line that can separate the objects.

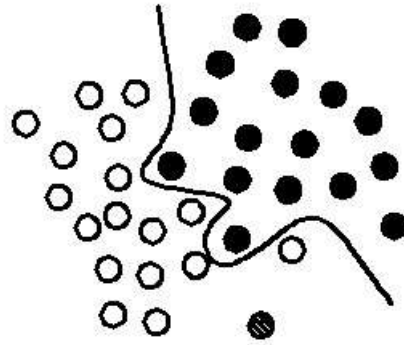


Figure 2.2: Example of a non-linear classifier

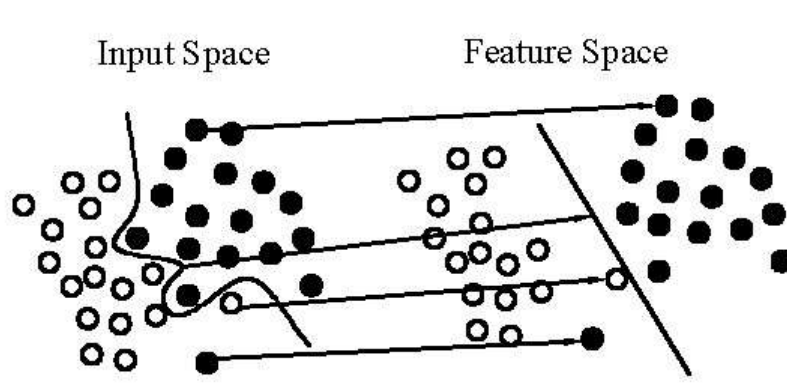


Figure 2.3: The basic concept of SVM

iii. Decision Trees: Decision trees are very commonly used in data mining. Decision trees are usually constructed top-down, by choosing a feature at each step that best splits the set of items. Different decision trees use different metrics for measuring the “best” feature. These generally measure the homogeneity of the test instance within the subsets. Some metrics that are commonly used in these trees are information gain, gini impurity, variation reduction etc. An example is shown in figure 2.4. Each interior node corresponds to one of the features. There are edges to children for each of the possible values of the features. Each leaf represents a class from a set of predefined class set given the values of the features represented by the path from the root to the leaf. Text classification usually involves a large number of features. Decision trees may perform badly in such high dimensional feature space.

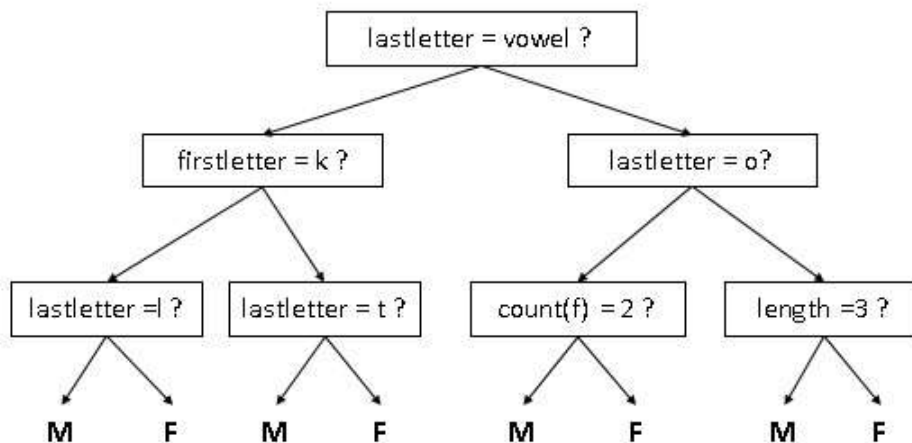


Figure 2.4: Decision tree for the name gender task

- iv. Other Classifiers: Almost all classifiers can be adapted to the case of text data. Some of the other classifiers include nearest neighbor classifiers, neural network classifiers and genetic algorithm-based classifiers.

2.2.3 Ensemble Methods

Ensembles allow us to create a more powerful learner from a set of base learners. They are known to produce better results than the individual algorithms and are better at reducing generalization errors. The base learners are also referred as weak learners. The base learning algorithms used by an ensemble could be of different types. For example the individual base learners could be Bayesian, Decision Trees, SVM, etc. In other cases, the base learners could be the same algorithm with different tuning parameters and training sets. For example, in Random Forest ensemble, each base learner is a Decision Tree. The five key ensemble techniques (Voting, Stacking, Bagging, Random Forest, and Boosting) are discussed and represented in simple graphics in this section.

i. Voting : This uses the sheer power of democracy. For classification, it takes majority vote on the predictions of base learners. For regression prediction, use the average of the predictions from the base learners.

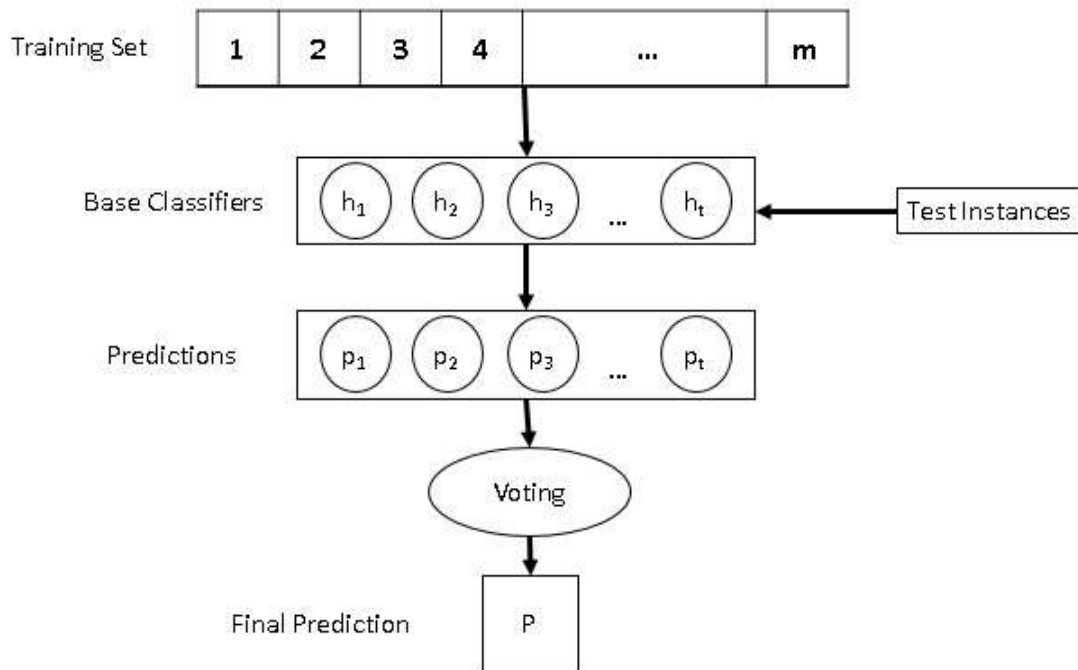


Figure 2.5: Voting ensemble

ii. Bagging : Bootstrap Aggregation (or Bagging for short), is a simple and very powerful ensemble method. It combines the predictions of multiple machine learning algorithms to make more accurate predictions than any individual model. It ensembles the result by giving equal weighted vote to all the classifiers. The class that receives the most voting will be the chosen class result. A tie is broken by choosing a default class if the default class is involved in the tie. Otherwise, a random class is chosen from base learners presented in the tie. It is a general procedure that can reduce the variance for those algorithms that have high variance. Decision trees, like classification and regression trees (CART) are such type of algorithms.

iii. Boosting : The term “Boosting” refers to a family of algorithms which converts weak or base learners to strong learners. Boosting pays higher focus on instances which are misclassified or have higher errors by preceding

weak learning algorithm. For choosing the right distribution, the base learner takes all the distributions and assign equal weight or attention to each instance. Then if there is any prediction error caused by the first base learning algorithm, then the instances with prediction error will get more weight in the next learners. This process will be repeated till the limit of base

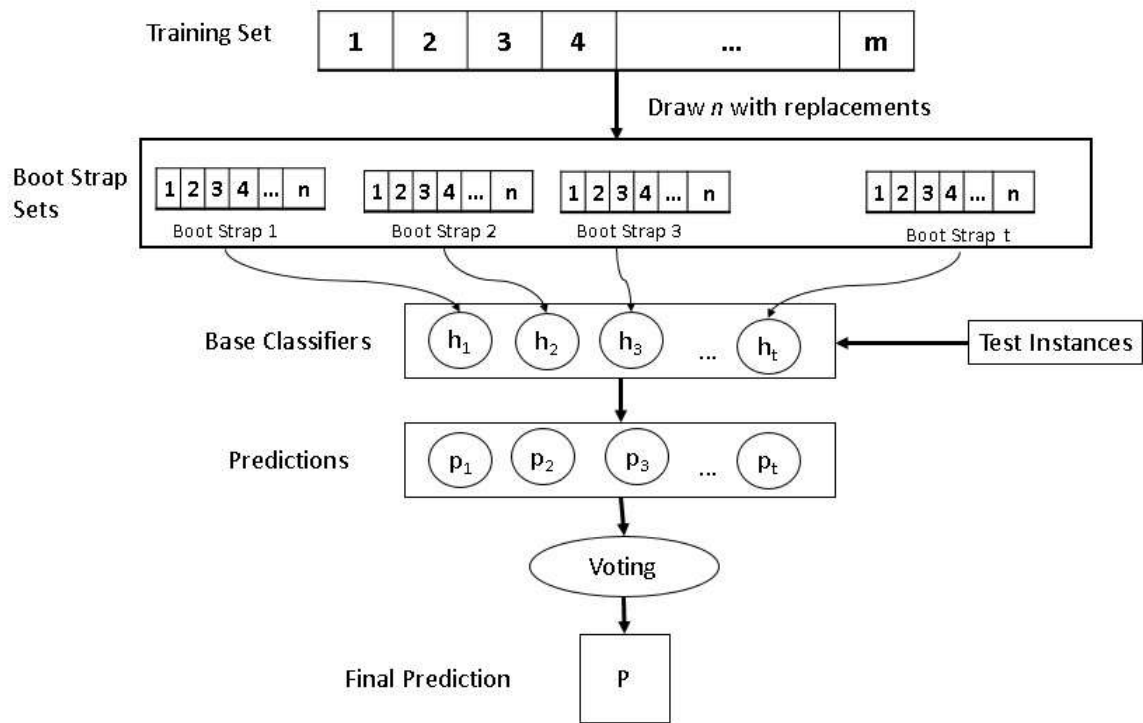


Figure 2.6: Bootstrap aggregation

learning algorithm is reached or higher accuracy is achieved. Finally, it combines the outputs obtained from base learners to create a strong one that should essentially improve the prediction power of the model.

iv. Random Forest: Random forest is an ensemble of decision trees. They are known to run efficiently on large datasets, and can take care of large number of features. Each decision tree is generated from a random subset of features of the training set. If there are a total of X features in the instances, then a number x is chosen such that x is very small than X . One choice for x is $\text{sqrt}(X)$. This value of x is held constant when the forest is growing. For split of each node, in each of the base learners (Decision Trees), x features are randomly chosen out of X . Then a feature out of x is chosen to split the node.

This is one of the reasons that random forest trees are created faster than regular trees, where each split decision involves all the features. Each tree is grown to the largest extent possible. For high variance and low correlation between any two base learners, no pruning is carried out on the trees.

v. Stacking : Stacking is another way of combining multiple models. Unlike bagging and boosting, stacking is normally used to combine models of different types. First,

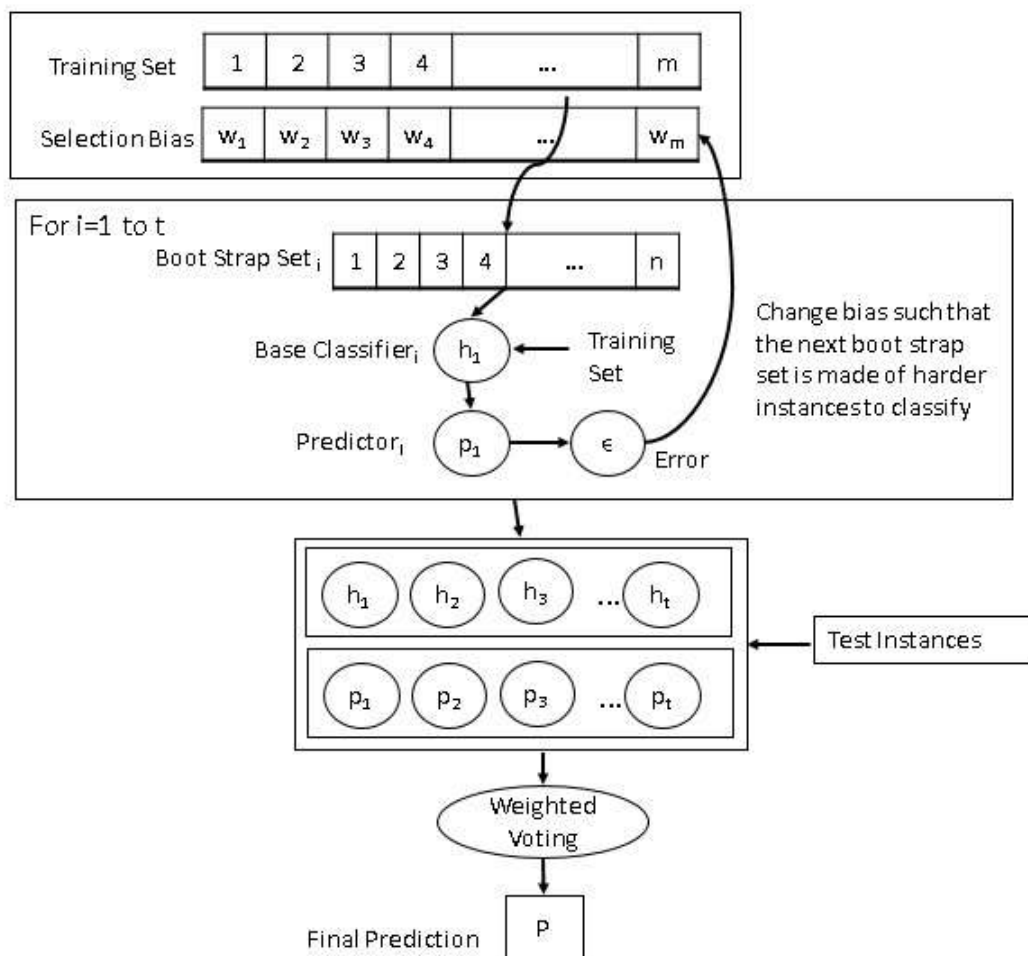


Figure 2.7: Boosting ensemble

it splits the training set into two disjoint sets and trains several base learners on the first set. The base learners are then tested on the second set of the training instances. Finally, stacking uses the output of the base classifiers as training data for another classifier to approximate the same target function.

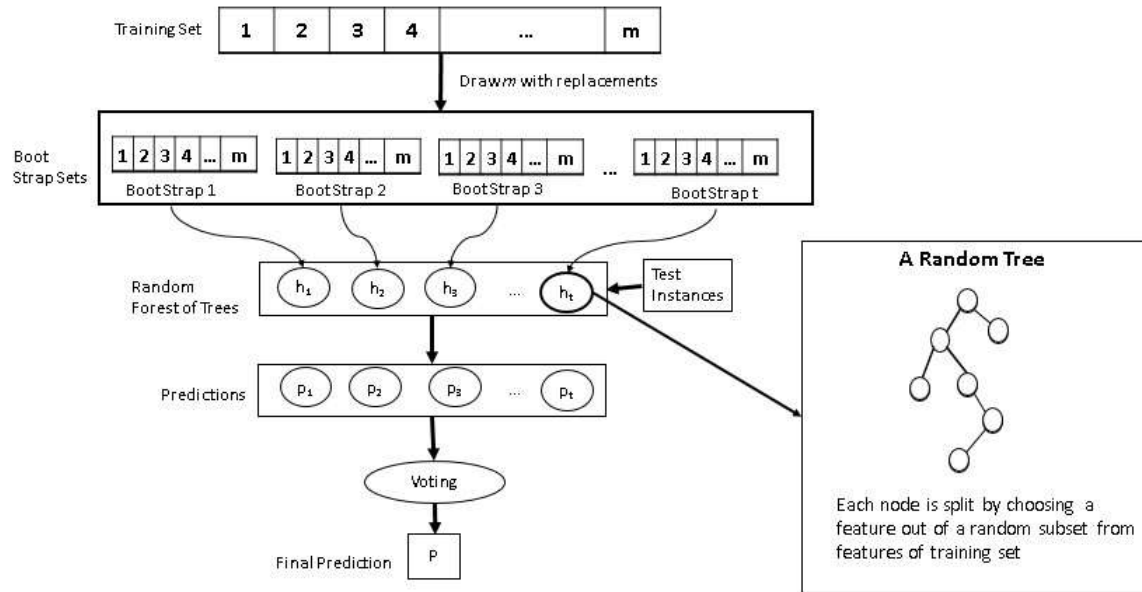


Figure 2.8: Random forest

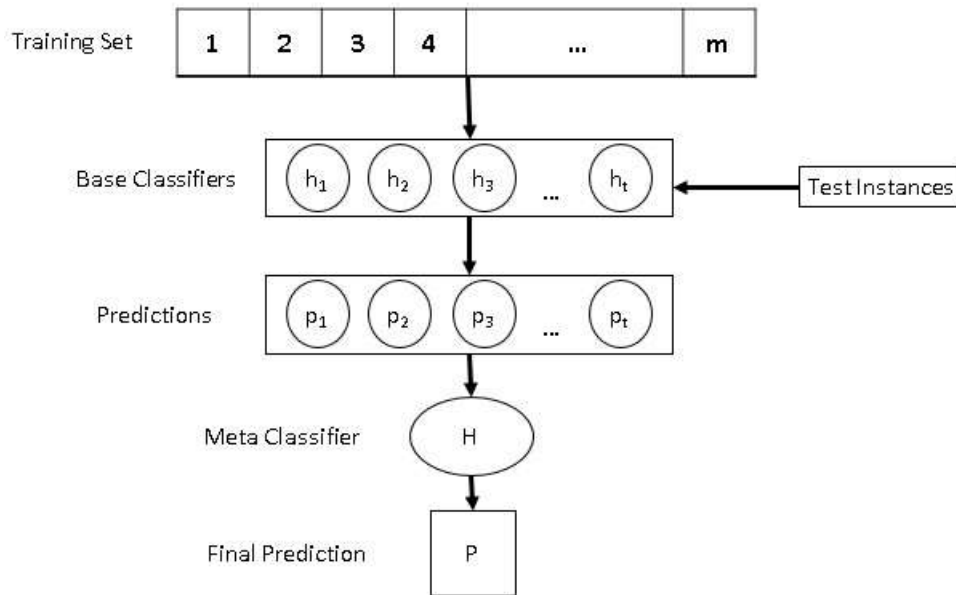


Figure 2.9: Stacking ensemble

2.3 Handling Imbalanced Dataset

The problem of imbalance class distribution arises in many classification tasks. Imbalance class distribution means there are many more instances of some classes than others. In such cases, the classifiers tend to produce higher accuracy on the majority classes but lower accuracy on the minority ones. In our case, the class distribution in spoiler detection is also imbalanced, because the number of texts, that contain spoilers, usually tends to be much lower than the that of non-spoilers. All the existing approaches to resolve this problem of imbalance class distribution fall into two category: presampling and cost-sensitive classifier.

2.3.1 Presampling

Presampling method makes the training set balanced, either by oversampling the minority class or by undersampling the majority class.

The oversampling methods try to overcome the property of imbalanced class distribution by adding examples to the training set. Random oversampling (ROS) method randomly duplicates the examples of the minority class. The disadvantage of ROS is that it may introduce redundancy and eventually cause over-fitting [8]. Instead of randomly duplicating examples, synthetic minority oversampling technique (SMOTE) [9] generates synthetic data for the minority class. It can provide more related minority class examples and make the decision regions larger. However, SMOTE generates the same number of synthetic examples for each minority example and therefore may cause data overlapping [8]. Some methods, which can overcome this limitation of SMOTE have been proposed such as borderline SMOTE [10] and Adasyn [11]. Borderline-SMOTE only oversamples the borderline examples of the minority class. Adasyn adapts the number of synthetic examples for every minority example according to the distributions.

On the other hand, undersampling methods, such as random undersampling (RUS) and onesided selection(OSS)[12] prune the examples of the majority class. The main drawback of these methods is that the information contained in the pruned examples

might be lost and thus deteriorate the performance of the classifier [8]. However, ROS and RUS can also be employed to a training set simultaneously.

2.3.2 Cost- Sensitive Classifier

Cost-sensitive methods [13] are also regarded as important approaches to class imbalance problems. To avoid the minority class being overlooked, cost-sensitive classifiers assign a higher misclassification cost to minority class than to the majority class. In this way, a class imbalance problem can be formulated as a cost sensitive learning problem and solved by an existing method [14]. Cost-sensitive methods for class imbalance problems always follow two steps [15]: set a cost matrix to make the problem cost-sensitive, and then, employ a method to solve the cost-sensitive problem. For example, consider a two-class classification problem. If class A distribution is 1%, most classifiers would learn a trivial rejector as it is 99% effective. The weight of mistakes on class A (false negatives, FN) can be increased for instance in a 10:1 relation. The classifier will then try to avoid false negatives, because each one is equivalent to 10 false positives (FP). For this instance the cost matrix will be,

$$\begin{bmatrix} 0 & 1.0 \\ 10.0 & 0 \end{bmatrix}$$

Many cost-sensitive methods have been proposed such as cost-sensitive boosting [16] [17], meta cost [18], rescaling training data [13][19] and modified back propagation (BP) algorithm for MLPs[20]. But it is difficult to set a proper cost matrix for a class imbalance problem[16], and an inappropriate cost matrix may mislead the training process eventually.

2.4 Related Work

A lot of psychological researches have been carried out on the effect of spoilers. However, there has not been much work on automatic spoiler detection so far. In this section, we discuss the previous studies on spoiler detection. We also discuss the similar studies on other text classification tasks such as spam detection and sentiment analysis, although the characteristics of these problems are remarkably

different from spoiler detection. Spams are unwanted, irrelevant or unsolicited messages sent over the internet, typically to a large number of users, for the purposes of advertising, phishing, spreading malware, etc. On the other hand, sentiment analysis is the process of determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker. We categorize the previous studies on these text classification problems in the following subsections.

2.4.1 Simple Keyword Matching Approaches

The initial approaches on spoiler detection are mostly based on keyword matching. The first model proposed to prevent user's from being accidentally exposed to spoilers is a temporal filtering system, called *Anti-Spoiler* [21]. They prepared a sports-results database which includes keywords, such as 'win', 'lose', 'beat' and 'upset'. The proposed system analyzes a user requested web content and hides the portions of the content if it contains these keywords. Similar approach is also done by Golbeck [22]. The proposed system in his paper blocks all the tweets on a specific topic. On performance basis, recall of these systems is very high, but precision is significantly low.

Recently, a study has been carried out by Maeda et al. on the location to which the content of the spoilers correspond in the story documents [23]. They hypothesized that spoilers occur in the latter half of a story document. So, if the important keywords or phrases are collected from the latter half, spoilers in reviews can be detected using them. To test their hypothesis, they extracted words with higher frequency from the latter half and compared those to the words of the spoilers. They experimented their study on five novels and reviews collected from Japanese online review site, Booklog. Their results are quite inconclusive as the words of high frequency from latter portion of a story both appear in spoiler and nonspoiler reviews. The proposed method also cannot support words that are not directly used in the story documents,

2.4.2 Feature-based Classifiers

Machine learning algorithms, especially supervised learning techniques such as Naive Bayes and Support Vector Machine(SVM), are the most common practices for text classification tasks. For detecting spoilers, Boyd-Graber first proposed a linear-kernal SVM using some metadata based features [24]. These features are Genre, Length, First Aired, Country and Episodes. These are proposed along with the traditional baseline features such as BOW and n -grams. Their results showed that these additional features increase the overall accuracy. The major drawback of their proposed method is that the additional meta-information were collected manually from different sites, such as IMDb, episode guides and tv tropes. Recently in 2016, four distinguished features, Named Entity, Frequently Used Verb, Objectivity and Url, and Main Tense of Tweet, are proposed to detect spoilers in tweets [25]. They also implemented their model using SVM. Though these features resulted in good performance, there are some major limitations. Firstly, the frequently used verb list was set based on a manual study. Their method was experimented on tweets related to a reality TV show “Dancing with the Stars, US. Season 13”. Tweets are restricted to only 140 characters, but posts with long text are frequently seen in other social networks such as facebook, IMDb. The features, Named Entity, Objectivity and Url, and Main Tense of Tweet, are not appropriate for long posts. For example, a review of moderate length, on a movie usually contains more than one character name for several times. It also includes both objective and subjective sentences, and also the tense of the sentences varies. So these features will be very noisy in the case of long texts. Moreover, the context of a reality show or any sports is also quite straight forward. The spoilers are usually about eliminating, winning, voting or loving someone in the competition. On the other hand, the contexts of movies or books are quite distinct from each other. In these cases, the proposed features will not perform as well as in their experiment.

The features that are proposed for spam detection are quite different from spoiler detection. To grab the properties of social spam, Markines et al. detected and analyzed six distinct features, such as TagSpam, TagBlur, DomFp, NumAds, Plagiarism and ValidLinks [26]. These features were used with AdaBoost and experimented on a public dataset from BibSonomy.org. Lin and Jia [27] adopted

three types of features to detect social spam. These features are lexical- measures the difference in behaviors of spammers and legitimate users; status- measures the outlink URLs, length of login, nature of topics, use of emotions, and reposting patterns; and user- measures the number of user's followers and users. The developed classifiers by incorporating Naive Bayesian algorithm, logistic regression, and SVM. Dae-Ha et al. utilized social network features, such as request reject ratio, request acceptance ratio, personality commonness, same community, and friend's friend, to train a Bayesian Network classifier for detecting social spam on SMSs [28]. Po-Ching and Po-Min applied a J48 decision tree algorithm to analyze features, such as URL rate and interaction rate, for detecting spam accounts on Twitter [29]. Sureka proposed an effective method for detecting social spam in YouTube comments, which mines activity logs of users to extract patterns such as average time difference between comments, percentage of comments, comment repeatability across videos, and comment repetition and redundancy [30].

Numerous previous studies have been carried on using n -grams and POS tags for sentiment classifications. Pak and Paroubek proposed a multinomial naive bayes with n -grams and POS tags to classify the tweets as objective, positive and negative [31]. Similar approaches have also employed for tweets data by several researchers using classifiers, such as naive bayes, SVM and maximum entropy [32, 33, 34, 35, 36]. Davidov et al. made use of k -nearest neighbor strategy to assign sentiment labels using punctuation, single words, n -grams and patterns as different feature types [37]. Kamps et al. utilized the lexical database WordNet to determine the emotional content of a word along different dimensions [38]. They developed a distance metric on WordNet and determined semantic polarity of adjectives. An ensemble framework for sentiment classification has been proposed by Xia et al. [39]. They used two types of feature sets, POS tags and word relations. They employed three base classifiers: naive bayes, maximum entropy and SVM. They applied ensemble approaches, such as fixed combination, weighted combination and Meta-classifier combination.

2.4.3 LDA-based Topic Models

Latent dirichlet allocation (LDA) is a fully generative graphical model for analyzing the latent topics of documents [40]. LDA models every topic as a distribution over the terms of the vocabulary, and every document as a distribution over the topics. These distributions are sampled from Dirichlet distributions. Guo and Ramakrishnan proposed a topic model [41][42] based on LDA for detecting spoilers [43]. They experimented their model on the reviews of four movies collected from the Internet Movie Database (IMDb). They presented the problem as a ranking problem. Reviews that contain more spoiler materials are targeted to rank higher than the others. They also used dependency parsers to extract features instead of the traditional bag-of-words (BOW). One of the major disadvantages of using LDA is that, it requires setting a number of topics in advance. In spoiler detection, setting the number of topics is irrelevant to the problem. Another disadvantage of this LDA-based ranking model is setting the number of reviews considered to be spoilers, N . Depending on the value of N , the performance matrices, precision and recall, of their datasets significantly varied. Topic modeling approaches based on LDA have also been adapted for sentiment analysis [44] [45] [46] and spam detection [47].

2.5 Statistical Tests for Comparing Algorithms

A statistical test provides a mechanism for making quantitative decisions about a process or processes. The intent is to determine whether there is enough evidence to “reject” a conjecture or hypothesis about the process. The conjecture is called the null hypothesis. In terms of selecting a statistical test, the most important question is “what is the main study hypothesis?”. If there is no hypothesis, then there is no statistical test. In this thesis, we want to use statistical test in order to compare our models to each other and find if there is any significant difference in their results for several datasets. So initially we need to formulate a null hypothesis, which states that there is no difference between the results of the compared models. At the end of the study, our expectation is that the null hypothesis will be rejected proving our proposed methods performed significantly better than baseline methods. There are

many statistical tests. We choose Wilcoxon signed rank test [48] to compare our models.

2.5.1 Wilcoxon Signed-rank Test

The Wilcoxon signed rank sum test is a non-parametric or distribution free test. It is used to compare two sets of scores that come from the same participants. The requirements are as follows:

- i. The dependent variable should be measured at the ordinal or continuous level.
- ii. The independent variable should consist of two categorical, “related groups” or “matched pairs”. “Related groups” indicates that the same subjects are present in both groups. It is necessary to have the same subjects in each group, because each subject has been measured on two occasions on the same dependent variable.
- iii. The distribution of the differences between the two related groups (i.e., the distribution of differences between the scores of both groups of the independent variable) needs to be symmetrical in shape.

Test Procedure:

H_0 : The median difference, M , is equal to zero.

- i. Calculate each paired difference, $d_i = x_i - y_i$, where x_i, y_i are the pairs of observations.
- ii. Exclude pairs with $|x_i - y_i| = 0$. Let N_r be the reduced sample size.
- iii. Order the remaining N_r pairs from smallest absolute difference to largest absolute difference, $|d_i|$.

iv. Rank the pairs, starting with the smallest as 1. Ties receive a rank equal to the average of the ranks they span. Let R_i denote the rank.

Label each rank with the sign of d_i .

v. Calculate R^+ , be the sum of positive ranks and R^- , the sum of negative ranks

vi. Let T be the smaller of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to

the critical value for n degrees of freedom, the null hypothesis of equality is rejected; this will mean that the first model outperforms the other one. The critical values are available at any standard books on statistical tests.

The effect of replacing the original measures with ranks is two-fold. The first is that it brings us to focus only on the ordinal relationships among the measures which are “greater than”, “less than” and “equal to”. The second is that it transforms the data array into kind of a closed system whose properties can then be known by dint of sheer logic.

2.6 Summary

People can be exposed to spoilers by not only texts, but also by images and videos in social network. We confine the scope of this thesis only to texts. However, detecting spoilers in texts is more challenging compared to other text classification problems. The major property of this task that the spoilers do not depend on any general patterns or concepts, rather it entirely depends on the concerning work of art. We design each phases of text classification to suit this unique property of this problem.

Chapter 3

Proposed

Method

3.1 Introduction

We present our proposed model and its components extensively in this chapter. We also describe the motivation behind the architecture of each element. The novelty of our proposed method lies in its feature extraction technique and the amalgam oversampling method. We wrapped the base classification module with boosting ensemble algorithm to analyze its effect on the result.

In this chapter, we first present a basic outline of our proposed method. Then, we discuss the different components of the method in detail. At last, we conclude with a brief summary of the topics discussed.

3.2 Overview of the Proposed Method

In our task of detecting spoilers, we are given N labeled texts, and each text belongs to either positive class or negative class. The text that contains spoiler material is considered as a positive instance, otherwise as a negative instance. Our goal is to provide an effective classification method that will accurately predict the class of an unlabeled text.

Our proposed method starts with a text preprocessing module. Preprocessing module takes raw training texts as input, tokenizes them to generate features and then selects a number of relevant features. Then the instances with selected features are forwarded to the base classification model. The base classification model includes a base classifier and an oversampling technique. To address the data imbalance property of our problem, we oversample the minority class which is, in our case, the positive class. To avoid over-fitting, our proposed oversampling technique combines

a simple distribution-based oversampling technique and a synthetic oversampling technique. We also use a boosting algorithm that combines the output of t base classification model to boost the performance. A higher level block diagram of our proposed method is presented in figure 3.1. We discuss its different components in depth in the following subsections.

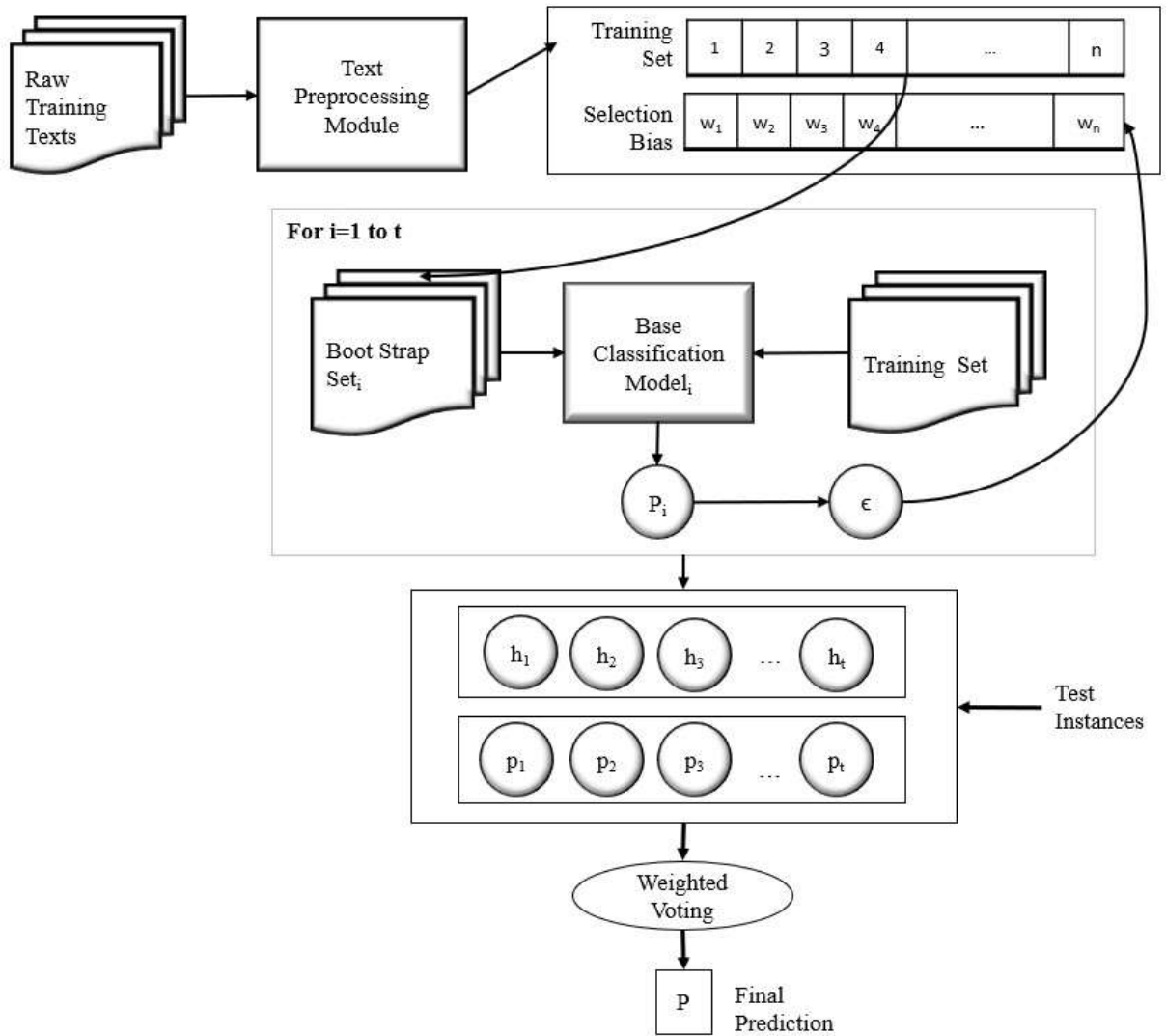


Figure 3.1: Block diagram of the proposed method

3.3 Text Preprocessing

Preprocessing is an important task and critical step in text mining. It is the process of preparing and cleaning the raw text for classification. Our preprocessing module has two components: Feature Generation and Feature Selection as shown in figure 3.2. In feature generation, we employ dependency parsing along with BOW to convert raw text into a set of insightful features that we can feed the classification model. Then in feature selection, we rank features using information gain and select a number of top features to discard noisy features and to reduce dimensionality. Both components are described in detailed in the following subsections.

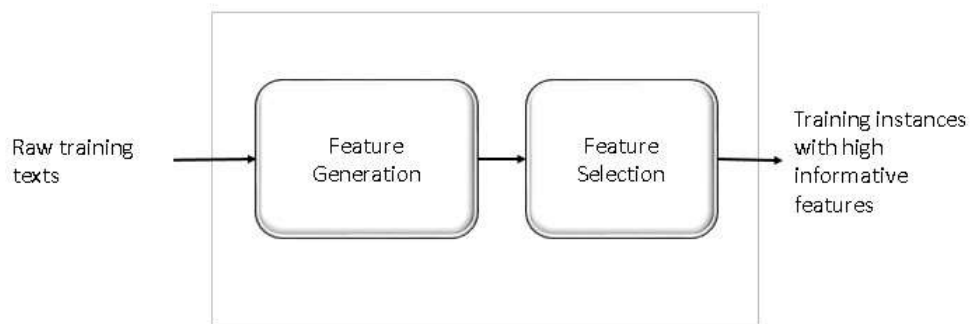


Figure 3.2: Text preprocessing module

3.3.1 Feature Generation

There is no specific rules of constituting spoilers. What constitutes spoiler depends on the context of the art of work. For example, in case of movies or books, the storyline defines which information contains spoilers and which does not. That's why, it is important that the features generated from the raw text represent the concepts of the text effectively. As mentioned in 2.2, the BOW and the n -gram model are unable to capture the gist of the text properly. There are alternative ways to represent the sentences of a text, such as typed dependencies and phrase structures. In our model, we use typed dependencies to extract features from text.

A typed dependency parse represents dependencies between individual words and also labels dependencies with grammatical relations such as subject or indirect object. In our experiments, we use Stanford typed dependency parser [49] to generate syntactically related word pairs. A dependency parser analyzes the grammatical structure of a sentence, establishing relationships between “head” words and words which modify those heads. The figure 3.3 shows a dependency parse of a short sentence. The arrow from the word *moving* to the word *faster* indicates that *faster* modifies *moving*, and the label *advmod* assigned to the arrow describes the exact nature of the dependency.

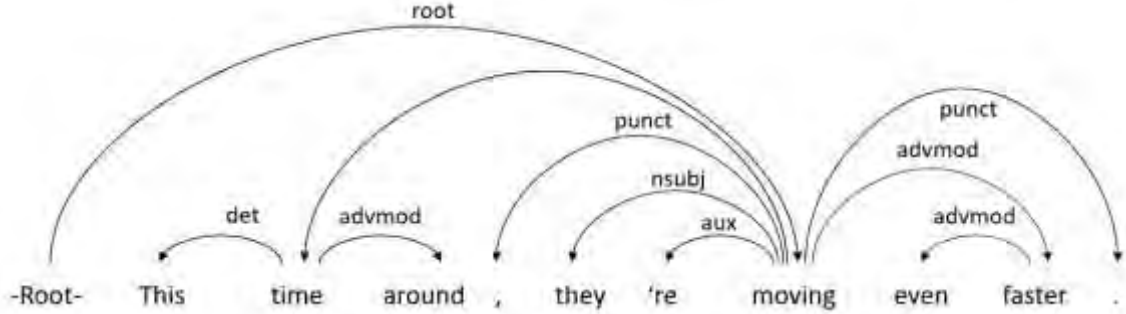


Figure 3.3: Dependency parse of a sample sentence

We use dependency pairs as our features. A dependency pair is a pair of grammatically related words. We exemplify this through a parse tree generated (shown in figure 3.4) by nlp standard parser for the following sample sentence.

“David Dunn is the sole survivor of this terrible disaster”

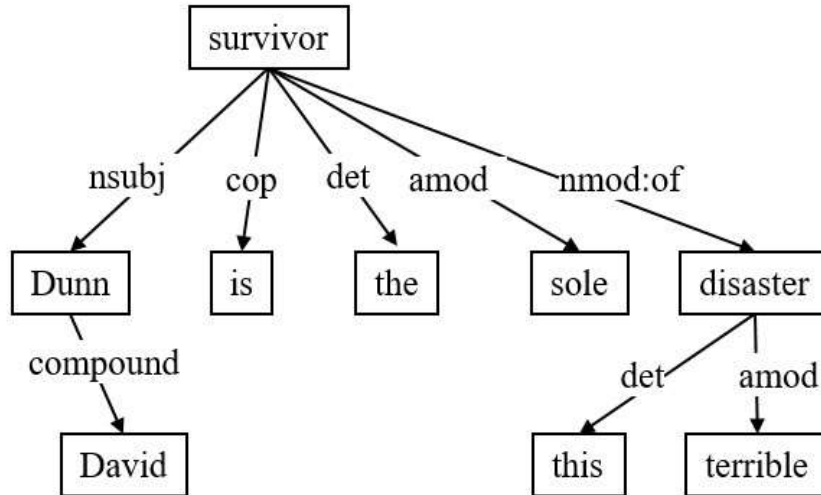


Figure 3.4: Parse-tree generated from a sample sentence

The dependency pairs, along with their parts-of-speech (POS) tags, extracted from the parsetree in figure 3.4 are provided in table 3.1.

Table 3.1: Dependency pairs generated from a sample sentence with POS tags

<nsubj>survivor, Dunn
<cop>survive, is
<det>survivor, the
<amod>survivor, sole
<nmod:of>survivor, disaster
<compound>Dunn, David
<det>disaster, this
<amod>disaster, terrible

The motivation behind using syntactically related word pairs instead of n -grams is to capture the concept of the text more effectively, and to introduce minimum noise. Consider the phrase “black large bear”. From the bigram model, we get two features: “black large” and “large bear”. Here, the feature “black large” is noisy, as these two words are not syntactically or semantically connected. On the other hand, dependency parser generates “black bear” and “large bear” phrases. These word pairs represents the text more appropriately.

We can also integrate plot related relations into our model using dependency parsing. Consider the following five sample sentences. They all contain two words: “Dunn” and “survivor”. The grammatical relation between these two words is given in beginning of each sentence in bold letter separated by a colon. These sentences are of various length and constructed differently, but they hold similar meaning. Another significant thing is that we can capture the semantic connections between these words, though they are physically separated by different length in different sentences. In the third sentence, the word gap between the two words is even 15.

nsubj: “David Dunn is the sole survivor of this terrible disaster”

appos: “David Dunn (Bruce Willis) is the only survivor in a horrific train crash”

nsubj: “David Dunn, a man caught in what appears to be a loveless, deteriorating marriage,

is the sole survivor of a Philadelphia train wreck”

appos: “In this Bruce Willis plays David Dunn, the sole survivor of a passenger train accident ”

acl:relcl: “Then the story moves to security guard David Dunn (Bruce Willis) miraculously being the lone survivor of a mile long train crash (that you find out later was not accidental), and with no injuries what-so-ever ”

The grammatical relations between the two words vary in the five sentences. In the fourth sentence, “survivor” serves as an appositional modifier of the term “Dunn”, whereas in other sentences, “Dunn” serves as the nominal subject of “survivor”. But the dependency pair, “Dunn, survivor” refer to the same individual. So we eliminate the pos tag from the generated parse so that we can treat the same dependency pairs with different pos tags as a single feature.

Table 3.2: Dependency pairs after removing POS tags generated from a sample sentence

survivor, Dunn
survive, is
survivor, the
survivor, sole
survivor, disaster
Dunn, David
disaster, this
disaster, terrible

We also eliminate the pairs in which one of the elements is a stop word, such as, determiners, conjunctions, prepositions etc. In this example, such dependency pairs are “survivor, is”, “survivor, the”, “disaster, this”. We remove these pairs as they do not essentially contribute to represent the concept of the sentence. We also lowercase every words of the sentence. So the final features extracted from the sample sentence are given in table 3.3

Table 3.3: Final features generated from a sample sentence after lowercasing words and removing noisy wordpairs

survivor, dunn
survivor, sole
survivor, disaster
dunn, david
disaster, terrible

A comparison of the features generated by BOW, bigrams and our proposed extraction technique from the sample sentence is given in table 3.4. It is quite clear from the comparison that the features, generated by dependency parsing, represent the concept of the sentence more effectively than the other methods.

Table 3.4: A comparison of the features generated by BOW, bigrams and our proposed extraction technique from the sample sentence

BOW	Bigrams	Proposed dependency pairs
David	David Dunn	survivor, Dunn
Dunn	Dunn is	survivor, sole
is	is the	survivor,disaster
the	the sole	disaster, terrible
survivor	sole survivor	
of	survivor of	
this	of this	
terrible	this terrible	
disaster	terrible disaster	

In our experiment, we employ BOW and our proposed dependency pairs both individually and collectively to compare their effectiveness.

3.3.2 Feature Selection

Text classification usually deals with staggering number of features. When classification model has hundreds or thousands of features, there is a good chance, many of these features are noisy. That means these features are common across all classes, but contribute little information to the classifier. Individually they are harmless, but in aggregate, low information features can decrease performance. The purpose of feature selection is to discard these irrelevant or redundant features from the given feature set and to expedite the whole process. It can save the model from overfitting and the curse of dimensionality.

Our feature selection method is a combination of a search technique for selecting the feature subset, along with a evaluation measure. We rank the features by their individual evaluations using information gain (IG) and select the top n features to use further.

Information gain is the amount of information that is achieved by knowing the value of the attribute. The key measure of information gain is entropy. It characterizes the

purity of an arbitrary collection of instances. The entropy measure is considered as a measure of dataset's unpredictability. The entropy of an entire dataset, X , is

$$H(X) = - \sum_{c \in C} p(c) \log_2(p(c)) \quad (3.1)$$

where $p(c)$ is the probability of class c that is the proportion of the number of instances in class c to the number of instances in X . $H(X) = 0$, when all instances belong to the same class. It means minimum impurity. $H(X) = 1$, when instances are equally distributed in all classes.

Information Gain is calculated by the difference in entropy from before to after the data set is split on an feature F . In other words, it measures how much uncertainty in X was reduced after splitting dataset on feature F .

$$IG(F, X) = H(X) - \sum_{t \in T} p(t) H(t) \quad (3.2)$$

where T is the subsets of instances created from splitting X by feature F such that $X = \cup_{t \in T} t$, $H(t)$ is the entropy of the subset t and $p(t)$ is the proportion of the number of instances in t to the number of instances in X .

We take each feature and calculate its information gain that represents its significance and relevance to the classes. Then we rank all the features according to their IG in the descending order and select the top n features for our classification model.

3.4 Base Classification Model

The base classification model comprises a base classifier and a minority overampling technique. Typically classifiers are more sensitive to the majority class and less sensitive to the minority class. In our case, instances in positive class are the relevant instances. As positive class is also the minority class, if we do not take care of the data imbalance distribution, the classification output will be biased. In most of the cases, it will always predict the negative class. This is the motivation behind incorporating an oversampling technique in our proposed method. Sampling method

usually carried on before building the classifier. However, we integrate our proposed distribution-based amalgam minority oversampling technique (DAMOT) into the base classifier. In this section, we describe functionality of our base classification model as well as the proposed novel oversampling technique in details.

The base classification model has mainly three steps:

1. Build the classifier and extract the distribution of positive instances over positive class (ρ)
2. Make a set (M_o) of positive instances from the proposed oversampling technique (DAMOT)
3. Update the classifier with M_o .

We use *naive bayes* as our base classifier. Naive bayes is described in detail in section 2.2.2. We first build the naive bayes classifier to calculate the distribution of positive instances over positive class (ρ). We employ this distribution in our proposed oversampling method. DAMOT consists of a simple distribution-based minority oversampling (SimDMO) and a synthetic distribution-based minority oversampling(SynDMO) method. We get a subset of original positive instances from SimDMO and a set of synthetic instances from SynDMO. Then the base classifier is updated by this newly generated mixture set of original and synthetic instances. We discuss DAMOT in depth in the next subsection.

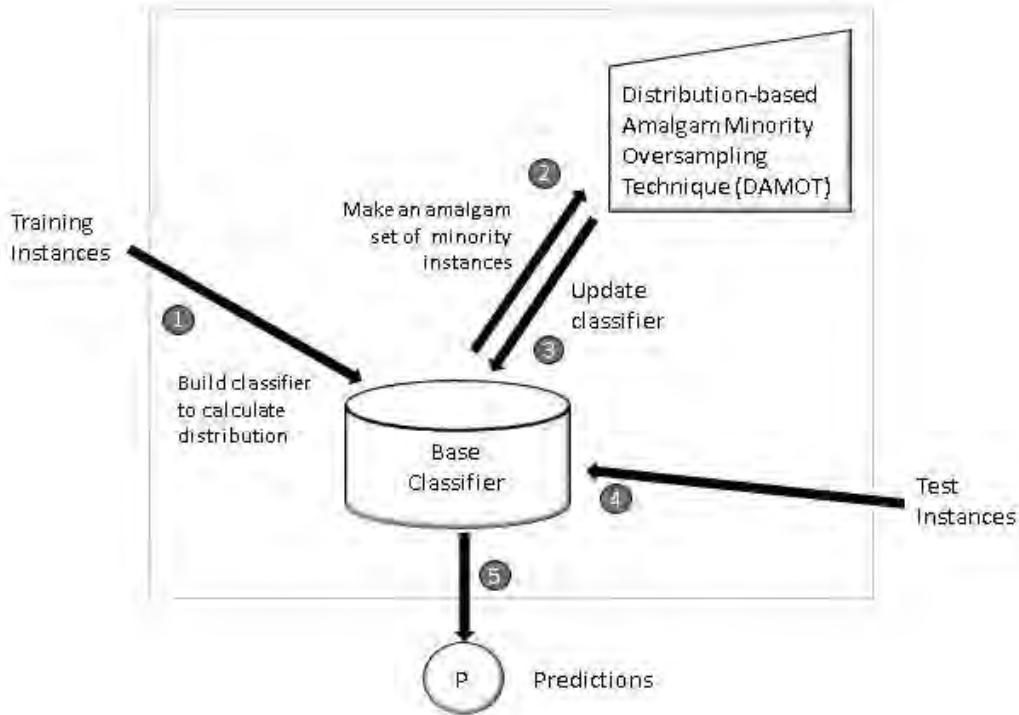


Figure 3.5: Base classification model

3.4.1 Distribution-based Amalgam Minority Oversampling Technique

The proposed oversampling technique, DAMOT, takes the difference between the number of positive and negative instances, D , as the required number of positive instances that should be oversampled. The percentages of D generated by SimDMO and SynDMO, are represented by R and S respectively. We can control the number of duplicated positive instances and that of synthetic instances for oversampling with the value of R and S .

The proposed oversampling method is described in algorithm 1. The algorithm starts with sorting the instances according to ρ in descending order (line 1). Then SimDMO begins with calculating the number (R) of instances that will be oversampled by itself (line 2). Then in line 3 - 6, a loop iterates R times. Each time, it simply duplicates the instance x_i (line 4) and add to the set M_o (line 5).

The rest of algorithm covers SynDMO (line 7 - 22). This is basically smote [9] algorithm except it prioritizes the positive instances with higher distribution over positive class. As the instances are sorted according to ρ , the instances with higher distribution are selected

first. In line 7, the number of required synthetic instances is calculated (S). We dynami-

cally calculate how many instances (\bar{S}) will be used to generate synthetic instances at what

percentage (N) (line 8-11). The values of \bar{S} and N depend on the number of positive instances and that of required synthetic instances. The main loop of SynDMO iterates through

\bar{S} times. At each iteration i , \bar{N} synthetic instances are generated from x_i . \bar{N} neighbors from

the k nearest neighbors are randomly chosen (line 16). If \bar{N} is 2, then two neighbors from the k nearest neighbors are chosen and one synthetic instance is generated in the direction of each (line 15 - 21). Synthetic instances are generated in the following way. A random number is taken between 0 and 1 (line 17). The random number is multiplied by the difference between vector x_i and vector x , then it is added to the feature vector x_i (line 18). This is how a random point (x) is selected along the line segment between two specific features. Then the new synthetic instance x is appended to M_o (line 19).

The motivation behind this amalgam is to avoid overfitting. We have analyzed the imdb reviews on several movies and observed that less than 30% of the overall reviews are considered as spoilers. To balance the dataset, we need to oversample nearly 200% of original minority instances. That is why, if we exclusively apply random oversampling method or synthetic minority oversampling method, it will cause overfitting.

We also utilize the distribution initially generated by the classifier to sort out the instances that have higher distribution over the positive class. We consider them as more spoilers than others. Some reviews are considered mild spoiler as they may not contain enough critical information to ruin suspense. When oversampling, we want to avoid these mild spoilers by giving priorities to the instances with higher distribution. At the same time we avoid overfitting and maintain diversity by oversampling with both original and synthetic instances.

Algorithm 1: Distribution-based Amalgam Minority Oversampling Technique (DAMOT)

Input: Minority Data $M^{(l)} = \{x_i \in X\}$ where $i = 1, 2, \dots, T$, Number of minority instances (T), Distribution of minority data over minority class (ρ), difference between the number of minor class and major class (D), SimDMO percentage (R), SynDMO Percentage (S)

- 1 Sort the minority instances in descending order according to ρ ;
- 2 Calculate the number for simple distribution-based oversampling instances,

$$\bar{R} = D * R / 100;$$

- 3 for $i = 1, 2, \dots, \bar{R}$ do
- 4 Copy the instance x_i , call this \bar{x}_i ;
- 5 Append \bar{x}_i to M_o ;

6 end

- 7 Calculate the number of synthetic instances, $S = D * S / 100$;
- 8 Calculate smote percentage, $N = (S / T) * 100$;
- 9 if $S > T$ then
- 10 $S = T$;

11 end

- 12 for $i = 1, 2, \dots, S$ do
- 13 Find the k nearest (minority class) neighbour of x_i ;
- 14 $N = \lceil N / 100 \rceil$;

15 while $N \neq 0$ do

- 16 Select one of the k nearest neighbours randomly, call this \bar{x} ;
- 17 Select a random number $\alpha \in [0, 1]$;
- 18 $\bar{x} = x_i + \alpha(x - x_i)$;
- 19 Append \bar{x} to M_o ;
- 20 $N = N - 1$;

21 end

22 end

3.5 Boosted Classification Model with Integrated Oversampling Technique

In conjunction with our base classification model, we employ a boosting algorithm, *AdaBoost*, to achieve higher performance. *AdaBoost* is short for “Adaptive Boosting”. It combines the outputs of the base classifiers into a weighted sum and presents it as the final output. It is called adaptive as the instances misclassified by the preceding base classifiers are prioritized in the following ones. We reconstruct the existing *adaBoost* algorithm to accommodate our proposed base classification model as its base classifier. We call this model, a Boosted Classification Model with Integrated Oversampling Technique.

The pseudocode for this model is given in algorithm 2. The inputs are the number of iterations, T and the given N labeled training instances, $\{(x_1, y_1), \dots, (x_N, y_N)\}$, where the labels $y_i \in \{-1, +1\}$ correspond to our negative and positive class respectively.

The algorithm begins with distributing equal weights to all the instances. The number of iteration, T , is the number of our base classifier modules used in the ensemble method. On each iteration t , a naive bayes classifier is trained with the weighted training set. Then we call DAMOT to generate a set M_o of positive instances made of duplicate and synthetic instances. Then the naive bayes classifier is updated with these newly generated positive instances. DAMOT is described detailed in 3.4. Once hypothesis h_t has been received, the error ϵ_t of classifier t is calculated. A parameter, α , is chosen from the error ϵ_t .

$$\alpha_t \leftarrow -\log\left(\frac{1 - \epsilon_t}{2 \epsilon_t}\right) \quad (3.3)$$

α measures the importance that is assigned to h_t . Some significant properties of α can be drawn from the equation [3.3]. α grows exponentially as the error approaches to 0. Better classifiers are given exponentially higher weight. α is zero if the error rate is 0.5. A classifier with 50% accuracy is no better than random guessing, so *adaBoost* ignores it. α grows exponentially negative as the error approaches to 1. So

negative weights are given to classifiers with accuracy lower than 50%. A plot of the values of α_t for classifiers with different error rates is given in figure 3.6.

Then the distribution, d_t is updated to increase the weight of instances misclassified by h_t , and to decrease the weight of correctly classified instances. Thus, the next classifier tends to

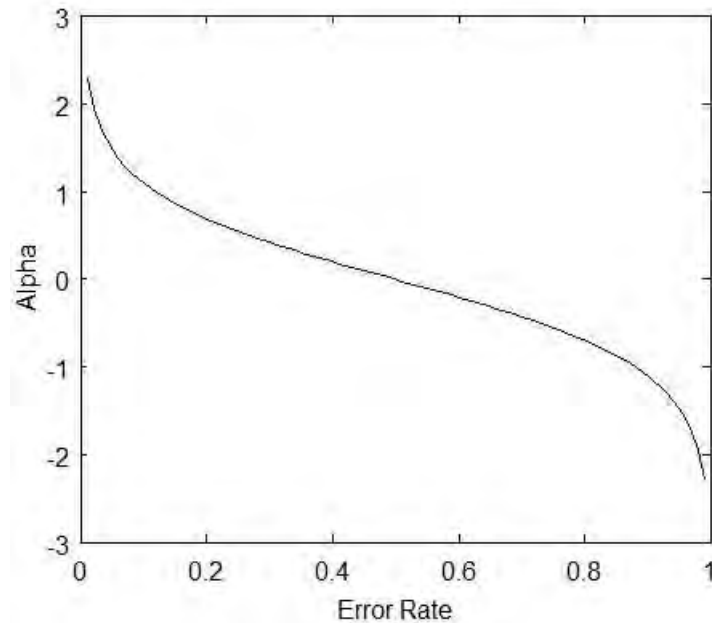


Figure 3.6: A plot of classifier weight against classifier error

concentrate on “hard” instances. The distribution is updated by the following rule,

$$d_n^{t+1} \leftarrow d_n^t \exp(\alpha_t |h_t(x_i) - y_i|) / Z_t \quad (3.4)$$

Here Z_t is a normalization constant, $Z_t = \sum_{n=1}^N d_n^{t+1} = 1$ The final hypothesis H is a weighted majority vote of the T hypotheses where α_t is the weight assigned to h_t .

$$H(x) = \sum_{t=1}^T \alpha_t h_t(x) \quad (3.5)$$

The key reason behind employing adaBoost algorithm because of its ability to reduce the training error. Consider the error as ϵ_t of h_t as $\epsilon_t - \gamma$. For binary problems, classifiers that predict instances at random has error rate of $\frac{1}{2}$, so here γ represents the measure that how much h_t performs better than a random classifier. Freund and Schapire [50] prove that the training error of the final hypothesis H is at most,

$$\prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp(-2\sum_t \gamma_t^2) \quad (3.6)$$

Thus, if each weak hypothesis is slightly better than random classifier, so that $\gamma_t \geq \gamma$ for some $\gamma > 0$, then the training error drops exponentially fast. AdaBoost adapts to the error rates of the individual classifiers. Another significant property of adaBoost is that it can identify outliers, i.e., instances that are either mislabeled in the training data, or which are inherently ambiguous and hard to categorize. Because adaBoost focuses its weight on the hardest examples, the examples with the highest weight often turn out to be outliers.

To compare the effectiveness of adaboost, we first apply it with only naive bayes, then in conjunction with our proposed DAMOT as well as random resampling. We also experiment each of the mentioned methods for both BOW and dependency parses exclusively and collectively.

Algorithm 2: Boosted Classification Model with Integrated Oversampling Technique

Input: An integer T specifying number of iterations, and N labeled training data

$$\{(x_1, y_1), \dots, (x_N, y_N)\}$$

1 Initialize weights

$$d_n = \frac{1}{N}$$

for all $n = 1, \dots, N$;

2 for $t \leftarrow 1, 2, \dots, T$ do

- (a) Train *Naive Bayes* with the weighted training set
- (b) Extract instances from positive class, M
- (c) Get the distribution of M over the positive class, ρ
- (d) Get the difference between the number of negative class and positive class, D
- (e) Call $DAMOT(M, \rho, D, R, S)$ to get the set of minor oversampling instances, M_o
- (f) Update *Naive Bayes* with M_o and obtain hypothesis $h_t: X \rightarrow \pm 1$
- (g) Calculate the weighted training error of h_t :

$$\epsilon_t = \sum_{i=1}^N d_i^t |h_t(x_i) - y_i|$$

(h) Set

$$\alpha_t \leftarrow \frac{1 - \epsilon_t}{2 \epsilon_t}$$

(i) Update weights

$$d_n^{t+1} \leftarrow d_n^t \exp(\alpha_t |h_t(x_n) - y_n|) / Z_t$$

where Z_i is a normalization constant, such that $\sum_{n=1}^N d_n^{t+1} = 1$

3 end

Output: $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

3.6 Divergence between the Proposed method and the Previous Approaches

There are mainly three machine learning approaches [43] [24] [25] on spoiler detection so far. We specify the key differences between those and our proposed method in this section.

Other than us, Guo and Ramakrishnan [43] have used dependency parsing in their feature extraction process. However, they use collapsed typed dependencies to form the features where part-of-speech tags are included in the dependencies. In our study, we exclude the POS tags from dependency parses to extract only the grammatically related word pairs. We also remove the word pairs which include auxiliary verbs, determiners, conjunctions and prepositions. Moreover, none of the previous studies take feature selection into consideration although it is an essential step in text mining. Due to the enormous number of features involved in text classification, classifiers suffer from overfitting and curse of dimensionality. We include feature selection in our preprocessing module which combines a search technique to select features along with information gain to measure the features.

To the best of our knowledge, no other previous studies employed ensemble based classification model to detect spoilers. J. Boyd-Graber et al. [24] and S. Jeon et al. [25] both have implemented SVM whereas Guo and Ramakrishnan [43] have built a topic model based on LDA. We use Naive Bayes with integrated DAMOT as our base classification model and enclose a bundle of base classification models with adaboost ensemble algorithm.

Though data imbalance distribution is an indigenous characteristic of this particular problem, none of the previous studies deal with it. We incorporate a novel minority oversampling technique, DAMOT, to handle imbalance in data distribution in our proposed method. Our proposed technique also handle overfitting by mixing original

and synthetic instances. It also avoid oversampling by mild spoilers by prioritizing instances with higher distribution.

3.7 Summary

We take each phase of text classification in consideration and design each component in favor for this specific problem of detecting spoiler. We presented the details architecture of our proposed method in this chapter. We utilize dependency parses in conjunction with

BOW for extracting features to represent the concept of the text effectively. Then the process of cleaning and selecting high informative features is also conducted in text preprocessing unit. Our major contribution lies in the base classification model where we integrate a new amalgam oversampling technique with the base classifier. We also incorporate the base classification model in adaboost algorithm as its base classifier to boost the classification performance.

Chapter 4

Experimental Studies

4.1 Introduction

In the previous chapter, we have presented our proposed method for detecting spoiler in depth. We have implemented several models using the major components of the proposed architecture individually and collectively. The goal of this chapter is to evaluate these models extensively and find out the one that is most suitable for our problem. Through various comparisons and statistical analysis, we gradually eliminate the models with less performance in order to filter the best ones.

We begin this chapter with a brief introduction of our datasets. We then describe the performance metrics used in the process of evaluation. This follows by the short description of the baseline methods and the experimental setup. Later on, we present our experimental result in details. Then we summarize all the discussions and conclude this chapter.

4.2 Datasets

To assess and compare the performance of our proposed model, we have collected our datasets from IMDb. It is the most popular and authoritative online database of information relating to films, television programs and video games. It includes information regarding the cast, production crew, fictional characters, biographies, plot summaries, trivia and reviews. We have collected reviews from 8 movies. We choose movies of different genres, years, and

number of reviews to take the effects of these parameters into account.

Table 4.1: An overview of dataset collected from eight movies

Movie Title	#Reviews	#Spoilers	Year	Genre
Unbreakable	1367	204 (15%)	2000	Drama, Mystery, Sci-Fi
The Usual Suspects	1000	230 (23%)	1995	Crime, Drama, Mystery
The Prestige	1000	290 (29%)	2006	Drama, Mystery, Sci-Fi
Inception	1000	300 (30%)	2010	Action, Adventure, Sci-Fi
Shutter Island	980	369 (37%)	2010	Mystery, Thriller
Blood Diamond	659	136 (21%)	2006	Adventure, Drama, Thriller
Shooter	342	114 (30%)	2007	Action, Crime, Drama
Role Models	173	50 (29%)	2008	Comedy

A brief description of the datasets is shown in table 4.1. Datasets are arranged in the table by the size of reviews in descending order. Henceforth, we denote the datasets by the movie titles. The first dataset is on a mystery and drama movie titled, Unbreakable. In this movie, a man, named David Dunn, learns that he is extraordinary after being a sole survivor of a horrific train crash. Later, he finds out that the train crash was not actually an accident. Hence, the spoilers are mainly about the origin of the train crash. The Usual Suspects is a mystery around a criminal master mind, called Keyser Soze whose identity reveals at the very end of the movie. In Prestige, two stage magicians engage in a battle to create the ultimate illusion. Near the end of the movie, the secret behind the great illusion of one magician discloses. Inception is a sci-fi movie where a thief uses a dream-sharing technology to steal corporate secrets. In Shutter Island, a U.S. Marshal investigates the disappearance of a murderer, who escaped from a hospital for the criminally insane. Later it turns out that the marshal, himself, is the patient. Blood Diamond is about a journey of a fisherman, a smuggler, and a syndicate of businessmen fighting over the possession of a priceless diamond. In the movie, Shooter, a retired marksman, framed for the murder attempt of the president, runs for his life while trying to find out the real criminal and the reason behind the crime. The last movie, Role Models, is a comedy, where two middle aged men are forced to change their wild lifestyle to become role models for two kids.

The degree at what a spoiler can impair the audience anticipation often depends on the genre of the movie. For instance, there can be crucial spoilers about movies like

Unbreakable, The Usual Suspects, The Prestige and Shutter Island where the entire plot is based on a single mystery. On the other hand, for some movies such as, Inception, Blood Diamond and Shooter, the stories evolve with the timeline. In these cases, spoilers are not as detrimental as the previous ones. Most spoilers are considered as mild in the case of comedy movies like Role Models as humors are the key ingredients in the stories rather than plot twists.

4.3 Performance Metrics

In this section, we describe the measures that are used to evaluate the predictive performance of our baseline and proposed models. We employ performance metrics such as kappa, accuracy, precision, recall, and f-measure. However, it is required to understand the concept of confusion matrix first to comprehend these metrics.

Confusion matrix: A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

For example, table 4.2 shows an example of confusion matrix for a binary classifier.

Table 4.2: Confusion matrix

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

The basic terms regarding the confusion matrix are as follows:

True positives (*TP*): The number of positive instances that are correctly classified.

True negatives (*TN*): The number of negative instances that are correctly classified.

False positives (*FP*): The number of negative instances that are misclassified as posi-

tive.

False negatives (*FN*): The number of positive instances that are misclassified as negative.

The total number of actual positive and negative instances are $TP+FN$ and $FP+TN$ respectively. On the other hand, the total number of predicted positive and negative instances are $TP+FP$ and $TN +FN$ respectively.

Accuracy: Accuracy is the most intuitive performance measure and it is simply the ratio of correctly predicted instances to the total instances. Accuracy is a great measure when datasets are symmetric. In asymmetric datasets like ours, other parameters are needed to be analyzed to evaluate the performance of the model.

$$\frac{TP+TN}{TP+FP+FN +TN}$$

$$Accuracy = \frac{TP+TN}{TP+FP+FN +TN} \quad (4.1)$$

Precision: Precision is the ratio of correctly predicted instances of a class to the total predicted instances of the corresponding class. Higher precision relates to the lower false positive rate.

$$\frac{TP}{TP+FP}$$

$$Precision = \frac{TP}{TP+FP} \quad (4.2)$$

Recall: Recall is the ratio of correctly predicted instances of a class to all the actual instances in the corresponding class. Higher recall means the rate of correctly classified instances of the concerning class is higher.

$$\frac{TP}{TP+FN}$$

$$Recall = \frac{TP}{TP+FN} \quad (4.3)$$

F-measure: F-measure is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. It reaches its best value at 1 which means perfect precision and recall, and worst at 0.

$$\frac{2*(Recall *Precision)}{Recall +Precision}$$

$$F -measure = \frac{2*(Recall *Precision)}{Recall +Precision} \quad (4.4)$$

Intuitively it is not as easy to comprehend as accuracy, but f-measure is usually more useful than accuracy, especially in an uneven class distribution. Accuracy works best when misclassifying both classes has similar cost. If the cost of misclassifying one class is higher than others, it is better to look at both precision and recall, hence f-measure.

Cohen’s kappa coefficient: Cohen’s kappa measures the agreement between two raters who each classify N items into set of mutually exclusive categories. It is generally more robust measure than simple percent agreement calculation as κ takes the possibility of the agreement occurring by chance into account.

$$\kappa = \frac{p_o - p_e}{1 - p_e} \quad (4.5)$$

where, P_o = the relative observed agreement among raters. P_e = the hypothetical probability of chance agreement. The value of kappa may range from -1 to +1, where 0 represents the amount of agreement that can be expected from random chance, and 1 represents perfect agreement between the raters. While kappa values below 0 are possible, but they are unlikely in practice. There is not a standardized interpretation of the kappa statistic. Acceptable kappa statistic values vary on the context of the problem. Landis and Koch considers 0-0.20 as slight, 0.21-0.40 as fair, 0.41-0.60 as moderate, 0.61-0.80 as substantial, and 0.81-1 as almost perfect [51].

4.4 Baseline Methods

To the best of our knowledge, there are only three machine learning approaches [43] [24] [25] conducted on spoiler detection so far. Guo and Ramakrishnan [43] ranked the reviews such that major spoilers are ranked to be higher than the mild spoilers. They considered first n reviews as spoilers. Depending on the value of n , the performance metrics varied. In our datasets, we include the reviews of the four movies they experimented on. The later approaches [24] [25] proposed feature-base

machine learning methods. Both of these methods used manually collected features such as “Frequently Used Verb”, “Genre”, “First Aired”, “Country”, “Episodes”. Moreover, they conducted their experiments on short text. So there is no direct way to compare our model to the existing approaches.

For baseline methods, we combine two feature extraction technique: BOW and dependency pairs, with random oversampling technique (ROS) and synthetic minority oversampling technique (SMOTE). We use *naive bayes* as base classifier in every case. From now on, we denote our baseline models as follows:

- i. BOW : using BOW features only.
- ii. DP : using only dependency pairs (DP) as features.
- iii. Mix : using both BOW and DP as features (Mix features).
- iv. ROS BOW : using ROS with only BOW features.
- v. ROS DP : using ROS with only DP as features.
- vi. ROS Mix : using ROS with mix features.
- vii. SMOTE BOW : using SMOTE with only BOW features.
- viii. SMOTE DP : using SMOTE with only DP as features.
- ix. SMOTE Mix : using SMOTE with mix features.

4.5 Experimental Setup

We have built several models exploring each of our major components, such as the feature generation techniques, our proposed oversampling technique, DAMOT, and boosting algorithm individually and conjointly. The following list provides the denotements and short descriptions of these models.

- i. DAMOT BOW : using DAMOT with BOW only.
- ii. DAMOT DP : using DAMOT with only DP as features
- iii. DAMOT Mix : using DAMOT with mix features.
- iv. Boosted BOW : using adaboost with BOW only.
- v. Boosted DP : using adaboost with only DP as features
- vi. Boosted Mix : using adaboost with mix features.
- vii. Boosted ROS BOW : using adaboost and ROS with BOW features.
- viii. Boosted ROS DP : using adaboost and ROS with only DP as features.
- ix. Boosted ROS Mix : using adaboost and ROS with mix features.
- x. Boosted SMOTE BOW : using adaboost and SMOTE with BOW features.
- xi. Boosted SMOTE DP : using adaboost and SMOTE with only DP as features.
- xii. Boosted SMOTE Mix : using adaboost and SMOTE with mix features.
- xiii. Boosted DAMOT BOW : using adaboost and DAMOT with only BOW features.
- xiv. Boosted DAMOT DP : using adaboost and DAMOT with only DP as features.
- xv. Boosted DAMOT Mix : using adaboost and DAMOT with mix features.

We consider top 2000 features ranked by information gain as relevant. While oversampling positive instances, we oversample 60% of the required instances by duplicating the original instances and rest by generating synthetic instances.

All the models have been developed in JAVA. We use standford parser [49] for generating dependency parses and WEKA library [52] for existing methods. All the experiments are conducted on a PC with a Intel(R) Core(TM) i7-7500U CPU running at 2.70 GHz with 12GB RAM.

We evaluate each of our model using k -fold cross validation technique. Cross Validation is used to assess the effectiveness of model, particularly in cases where overfitting is needed to be mitigated. In the following section, we briefly describe k -fold cross validation method.

4.5.1 k -fold Cross Validation Method

Simple cross validation technique, also known as the holdout method, removes a part of the training data to get predictions from the model after training it on the rest of the data. The error estimation then shows how our model is doing on unseen data or the test set. This method suffers from high variance as it is not certain which instances will end up in the test set and the result might be entirely different for different sets. It may also risk losing important information by reducing training data, which in turn increases error induced by bias. A variant of cross validation technique, called k -fold cross validation, resolves these limitations.

In k -fold cross validation, the data is divided into k subsets. Then the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of the model.

The advantage of using k -cross validation model is that every instance gets to be in a test set exactly once, and gets to be in a training set $k-1$ times. Interchanging the training and test sets also adds to the effectiveness of this method.

A slight variation in this technique is made to maintain balance in training and test sets. This variation is known as stratified k -fold. In this, approximately the same percentage of instances of each target class is given to each fold as a complete set. In case of prediction problems, the mean response value is approximately equal in all the folds.

We use 5-fold stratified cross validation to evaluate each of our model.

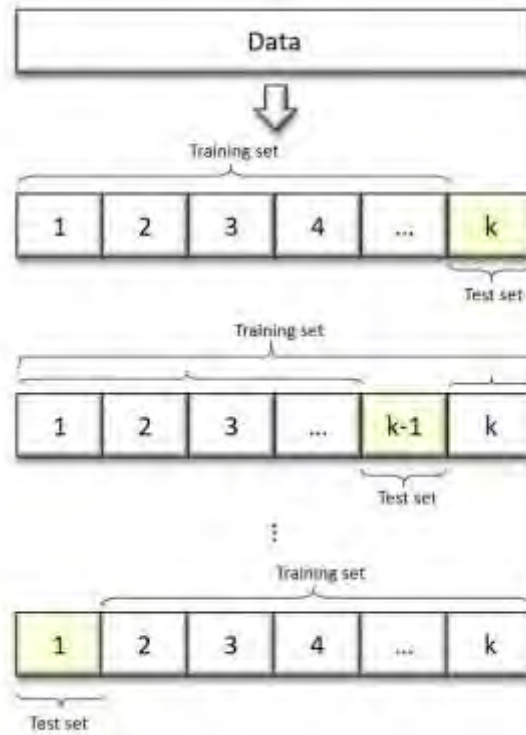


Figure 4.1: k -fold cross validation model

4.6 Results

We tested each of our baseline and proposed models with each dataset and generated the performance metrics. Table 4.3 and 4.4 show the average of the performance metrics of baseline and proposed methods for the eight datasets respectively. To present and conceive the performances of the models clearly, we highlight the top two values in each metrics. The cells with the highest values are underlined and embolden while the cells with the second highest values are just embolden.

We observe from table 4.3 and 4.4 that when a model achieves higher accuracy and negative recall, its positive recall decreases significantly and vice versa. For instance, Boosted BOW and Boosted Mix obtain higher accuracy, 71.077 and 71.015, but their positive recalls fall down to 0.467 and 0.446. Boosted DP and Boosted SMOTE Mix have higher negative recall, 0.823 and 0.844, with positive recall of 0.332 and 0.446 respectively. On the other hand, Boosted ROS Mix and

DAMOT Mix attain higher positive recall, 0.582 and 0.575, with accuracy of 67.592 and 64.161 respectively. This imbalance of models' performances in different metrics is expected as the number of positive instances is far less than that of negative instances. Because of the poor number of positive instances, it is not only more difficult for the models to classify the positive instances correctly but also classifying positive instances accurately has less effect on the overall accuracy.

The effect of misclassifying a positive instance is far more detrimental than that of misclassifying a negative instance. When a spoiler is marked as a spoiler free review, one can be exposed to it unintentionally which may essentially cause impairment to his or her enjoyment. On the contrary, if a spoiler free review is marked as a spoiler, it would not essentially have any significant effect on the audience. That's why, more importance is given to positive recall among other performance metrics when assessing the models. Our goal is to have higher positive recall without penalizing other metrics as much as possible.

Among the baseline methods, BOW and Mix have good performance with kappa 0.250 and 0.257, accuracy 68.446 and 69.358, positive recall 0.549 and 0.531, and negative recall 0.738 and 0.756 respectively. Besides these models, DAMOT BOW has also good performance with highest kappa 0.258. The models mentioned above in this section have shown prominent performance according to one or more metrics. Among these models, we select those models with minimum 0.4 positive recall for further analysis. These models are embolden in table and 4.3 and 4.4.

4.6.1 Effect of the Size of the Datasets

The size of the datasets plays an important role in the performances of the models. The average performances of the models are quite poor considering the value of kappa and positive recall. The highest average kappa and positive recall are only 0.258 and 0.582. To analyze the effect of the size of the datasets, we plot the performance metrics of the different models against the individual dataset (Figure 4.2).

All the models show similar performance trend for all the datasets. These models have decent performances for the first five datasets that have more than 900 reviews.

The performance starts to deteriorate from the movie, *Blood Diamond*, that has 659 reviews. For this movie, *Boosted Mix*, has even negative κ value meaning it has lower performance than random choice. Positive recall and kappa drastically decrease for *Role Models* that has only 173 reviews.

We also plot positive recall of the models, DAMOT BOW, DAMOT Mix and Boosted ROS Mix, against number of reviews from 100 to 1000 on an interval of 100 for individual dataset (Figure 4.3). The positive recall of DAMOT BOW and DAMOT Mix increases with the number of reviews for all datasets except The Usual Suspects. For this particular dataset, positive recall for first 100 reviews is quite high, then the positive recall drops little upto 600 reviews. Then it again starts increasing from 800 reviews. So along with the number of reviews, the performance also depends on other factors, such as data distribution and the degree of spoilers in the particular reviews. Another observation from the plots is that the model, Boosted ROS Mix has achieved higher recall for fewer number of reviews compared to the other two models.

Table 4.5 presents the average Kappa, accuracy, positive recall and negative recall for the five datasets with more than 900 reviews. Kappa value increases up to 0.341 which is moderate agreement between the predictions of a model and actual classes. The highest average positive recall also escalates significantly, from 0.582 to 0.676. The change in performance with the size of the datasets is expected as the models learn more effectively with more training data

Table 4.5: The average performance of models for datasets with more than 900 reviews

	Kappa	Accuracy	Positive Recall	Negative Recall
BOW	0.323	70.748	0.642	0.722
Mix	<u>0.341</u>	72.033	0.632	0.747
DHOS BOW	0.321	69.806	<u>0.676</u>	0.693
DHOS Mix	0.328	70.470	0.669	0.704
Boosted BOW	0.335	<u>73.389</u>	0.546	0.785
Boosted Mix	0.329	73.060	0.553	<u>0.799</u>
Boosted ROS Mix	0.267	66.283	0.673	0.654

4.6.2 Effect of the Components

In this subsection, we take each of our proposed components into consideration and analyze its effects on the outcome. Figure 4.6 shows how models perform on each dataset on the basis of kappa, accuracy, positive recall and negative recall.

Using dependency pairs along with BOW features, we are able to improve kappa, accuracy and negative recall, but unable to retain positive recall in most of the cases. Adaboost works best with random oversampling technique and BOW features. On the other cases, it boosts accuracy and negative recall highly with the cost of penalizing positive recall on a great scale. From the figure 4.6, it is quite visible that the models that bring balance in all the performance metrics are DAMOT BOW and DAMOT Mix.

We analyze the effect of injecting the synthetic instances and prioritizing instances by their distribution along with simple random oversampling (ROS) technique. To do so, we convert our DAMOT oversampling technique into ROS step by step. First, we eliminate the prioritization. Then, we gradually reduce the percentage of synthetic instances. When the percentage of synthetic instances becomes zero, the oversampling will be a simple ROS technique. For each change, we will analyze the effects of these changes on the positive recalls of our datasets. Here, we use mix features in all cases.

We mix a distribution based simple oversampling and a distribution based synthetic oversampling in our proposed oversampling technique, DAMOT. Here we oversample 60% of required positive instances by simple oversampling and 40% by synthetic oversampling.

While oversampling, we also prioritize instances with higher distribution. First we eliminate the prioritization. That brings us the model ROS(60%)+SMOTE(40%) Mix . In the figure 4.4, the first model is DAMOT Mix. In the following models, we exclude the prioritization of instances by their distributions. That makes the oversampling a mixture of ROS and SMOTE. The percentages in the title represent the percentage of ROS and SMOTE used in the corresponding model. For example,

ROS(60%)+SMOTE(40%) Mix means 60% of the required instances is oversampled by ROS and 40% is by SMOTE. We gradually reduce the percentage of SMOTE to analyze the effect of the injection of synthetic instances. The model ROS(100%)+SMOTE(0%) Mix means simply the ROS oversampling. In the figure 4.4, we plot the recall of the these models against individual movie dataset. We can observe from this plot is that prioritizing instances by their distribution as well as injecting SMOTE with ROS improve the positive recall. In all these cases, we use mix features and employed oversampling technique after feature selection.

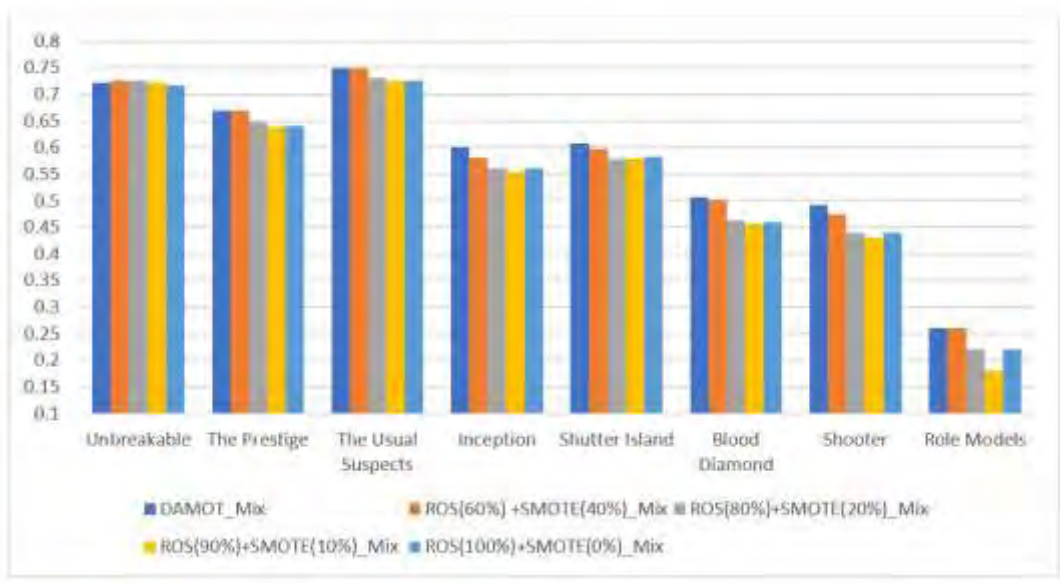


Figure 4.4: Effect of injecting synthetic instances in random oversampling

The performance of the models also relies on the time of the feature selection. When oversampling technique is employed before feature selection, it also affects the selection of the features along with their distributions. We applied random oversampling (ROS) before feature selection. On the other hand, smote is carried out after feature selection as it takes higher computational effort to generate synthetic instances from huge number of features. As DAMOT comprises of both ROS and SMOTE, we also employed it after feature selection.

In the figure 4.5, the baseline model ROS Mix has been added with the models in the previous experiment. ROS(100%)+SMOTE (0%) Mix model and ROS Mix model are same except in the first one the oversampling technique has been carried out after the feature selection where in the latter one, it has been carried out before

feature selection. The latter one results higher positive recall than the first one. We observe from figure 4.5 that DAMOT outperforms ROS in every dataset except The Usual Suspects and Shooter. This is because the oversampling alter the feature selection drastically for these two datasets. So the variation in the performance is the result of the timing of the feature selection process.

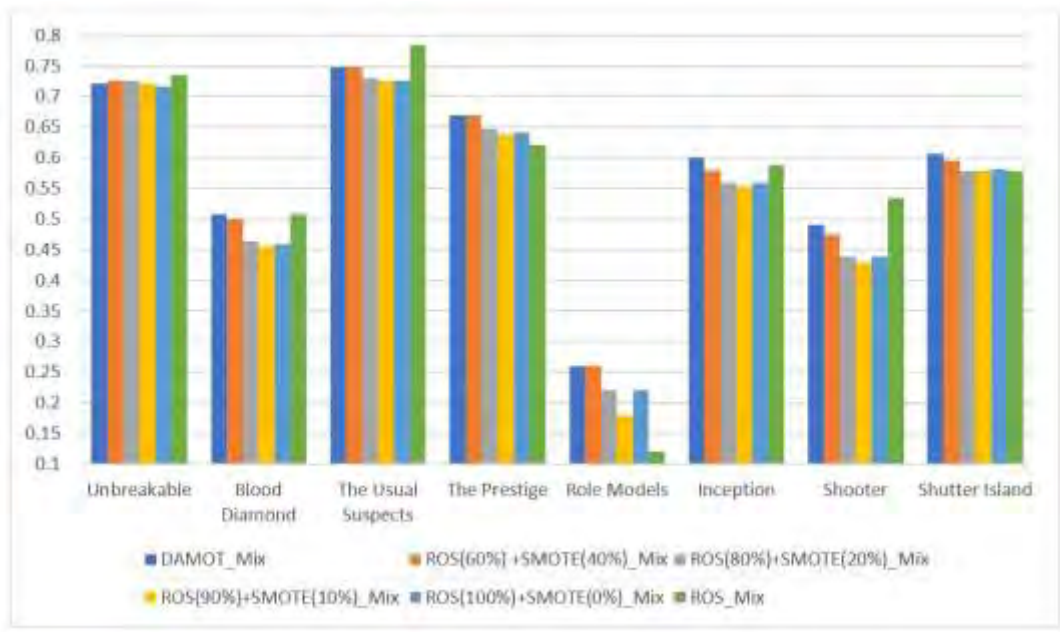


Figure 4.5: Effect of the time of feature selection

Another observation is that the recall is comparatively higher for the movies of mystery genre such as Usual Suspects, Unbreakable, Prestige. On the other hand, movies in adventure or comedy genre such as, Blood Diamond, Shooter and Role Models have lower recall. This is because the amount of spoiling material in adventure and comedy movies is comparatively less than movies of mystery genre. The spoilers are not as detrimental for these movies as well.

We also plot the deviations of the models from the top value in different performance metrics for each movie (figure 4.7). These plots illustrate the relative performance of the models. The lower the points reside for a model, the smaller the differences between its performance and the highest performance for individual movie dataset. This also supports the

intuition about the models obtained from the previous comparisons. For DAMOT BOW and DAMOT Mix, the points reside in the lowest region for positive recall except for the movie, Role Models. As the size of the dataset for this movie is very small, only 173 reviews, we can consider it as an irregular behavior. For other performance metrics, the points also stay in the lower region in the plots for these models. So, we can deduce from these comparisons is that our proposed new amalgam oversampling method, DAMOT, performs effectively to boost the positive recall with moderate accuracy, negative recall and kappa.

4.6.3 Statistical Analysis

We employ the Wilcoxon signed-rank test in a pairwise manner to evaluate the statistical significance between models. We conduct this test under a significance level of 0.10 for these performance metrics: accuracy, positive recall, negative recall and kappa. We compare DAMOT BOW, DAMOT Mix and Boosted ROS Mix models with others as these models has leading positive recall. We refer these models as candidate models and others, that are compared to them, as challenged models. We give positive ranks to candidate model and negative ranks to the challenged model. R^+ and R^- represent the sum of all positive and negative ranks, respectively. The test statistic, T_{value} , is the minimum of R^+ and R^- . T_{value} should be less than or equal to the critical value to indicate a significantly different result. Critical values for 6, 7 and 8 number of pairs are 2, 3 and 5 respectively. For better understanding, we embolden the metrics that has significant difference in favor for the candidate models and underlined the metrics having significant difference in favor for the challenged models. When comparing, if a model beats another as per positive recall, other metrics will be avoided. Other metrics will be taken into consideration when there is no significant difference in positive recall between two models. The reason behind this is explained in the beginning of this section.

Table 4.6 provides the detailed result of the wilcoxon signed-rank tests between DAMOT BOW and other models. According to positive recall, this model beats BOW, Boosted BOW, and Boosted Mix without having any significant difference with Mix, DAMOT Mix, and Boosted ROS Mix. Though this model defeats Boosted ROS Mix

by significant difference in kappa, but the baseline model, Mix, beats it according to accuracy, and negative recall.

The statistical comparisons between DAMOT Mix and other models are provided in table

4.7. The comparison with DAMOT BOW is not presented in this table as it is already provided in table 4.6. As per positive recall, it defeats all models except DAMOT BOW and Boosted ROS Mix. However, it beats Boosted ROS Mix according to Kappa.

Table 4.8 presents the result of wilcoxon signed rank tests between Boosted ROS Mix model and other model. The comparisons with DAMOT BOW and DAMOT Mix are ex-

cluded from this table as they are already provided in table 4.6 and table 4.7. Boosted ROS Mix has no significance lead in positive recall with Mix, Boosted BOW, Boosted DP, and Boosted Mix. It is defeated by both DAMOT BOW and DAMOT Mix in accordance with kappa. More-

over, baseline model, BOW, beats it as per accuracy, negative recall and kappa.

Summarizing table 4.6, 4.7, and 4.8, we observe that model DAMOT Mix defeats all the models except DAMOT BOW. It should be noteworthy that DAMOT BOW failed to defeat the baseline model, Mix, whereas DAMOT Mix has significantly higher recall than Mix model. However, we run wilcoxon signe-rank test for all performance metrics between DAMOT Mix and DAMOT BOW for the datasets with more than 500 reviews. This is because we want to consider only the moderate-sized datasets to avoid irregular performances obtained from small-sized datasets. The result of these tests is presented in table 4.9. DAMOT Mix succeeds to obtain notably higher performance than DAMOT BOW in accuracy, positive precision, negative f-measure and negative recall.

Table 3: Average performance of the baseline models for the eight movies

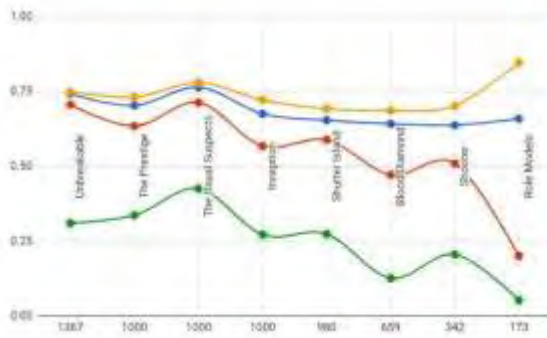
Negative classes	Recall	0.73	0.76	0.75	0.70	0.66	0.72	0.66	0.80	0.78
	Precision	0.81	0.78	0.81	0.81	0.79	0.81	0.79	0.76	0.79
	F-measure	0.77	0.77	0.78	0.75	0.71	0.76	0.73	0.76	0.77
Positive classes	Recall	0.54	0.42	0.53	0.56	0.53	0.55	0.51	0.29	0.41
	Precision	0.42	0.39	0.43	0.40	0.36	0.41	0.40	0.46	0.48
	F-measure	0.46	0.40	0.46	0.46	0.39	0.45	0.44	0.28	0.41
Accuracy		0.6846	0.6758	0.6958	0.6655	0.6212	0.6763	0.6698	0.6877	0.7047
	Kappa	0.25	0.17	0.25	0.23	0.15	0.24	0.20	0.12	0.22
		BO	DP	Mix	BO	DP	Mix	BO	DP	Mix
		OS	OS	OS	OS	OS	OS	OS	OS	OS

Table 4: Average performance of the proposed models for the eight movies

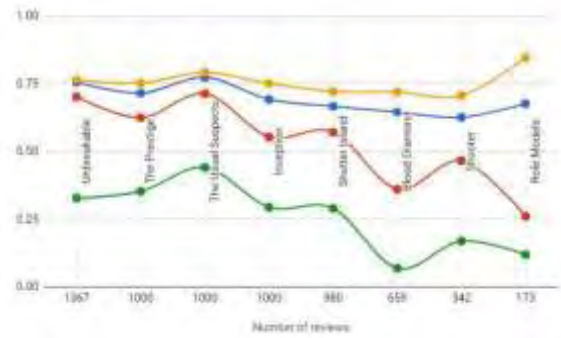
Negative Class	Recall	0.71	0.65	0.71	0.79	0.82	0.80	0.70	0.73	0.66	0.79	0.84	0.82	0.71	0.74	0.72
	Precision	0.81	0.78	0.81	0.79	0.76	0.79	0.80	0.77	0.79	0.76	0.76	0.77	0.80	0.77	0.80
	F-measure	0.76	0.70	0.76	0.79	0.79	0.80	0.74	0.74	0.72	0.77	0.78	0.78	0.76	0.76	0.76
Positive Class	Recall	0.57	0.52	0.57	0.46	0.33	0.44	0.55	0.45	0.58	0.36	0.25	0.35	0.51	0.43	0.53
	Precision	0.41	0.34	0.46	0.45	0.41	0.44	0.39	0.36	0.38	0.44	0.48	0.49	0.40	0.38	0.40
	F-measure	0.46	0.40	0.47	0.46	0.36	0.43	0.45	0.40	0.45	0.37	0.26	0.33	0.45	0.40	0.45
Accuracy	0.6799	0.6195	0.6792	0.7177	0.6906	0.7115	0.6610	0.6462	0.6461	0.6865	0.6931	0.7013	0.6769	0.6646	0.6746	
Kappa	0.24	0.13	0.24	0.25	0.16	0.24	0.21	0.15	0.19	0.16	0.11	0.16	0.22	0.16	0.22	

		B O D _r A M O _T	DP D _i A M O _T	Mi D _i A M O _T	B O B _o os te ↵	DP B _o os te ·	Mi B _o os te ↵	B O R _r OS B _o os te ·	DP R _r OS B _o os te ·	Mi x R _r OS B _o os te ↵	B O S _r M O _T B _o os te ·	DP S _i M O _T B _o os te ·	Mi S _i M O _T B _o os te ·	B O D _r A M B _o os te ·	DP D _i A M B _o os te ·	Mi D _i A M B _o os te ·
--	--	--	--	--	---	---------------------------------------	---------------------------------------	---	---	--	--	--	--	---	---	---

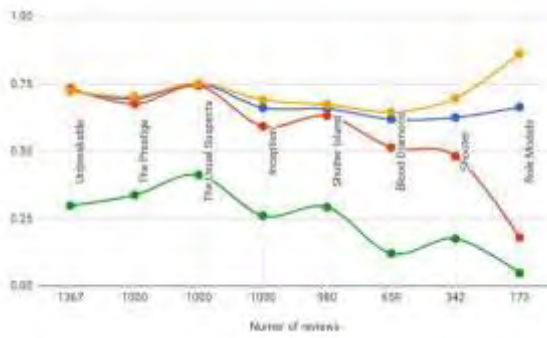
● Accuracy
 ● Pos Recall
 ● Neg Recall
 ● Kappa



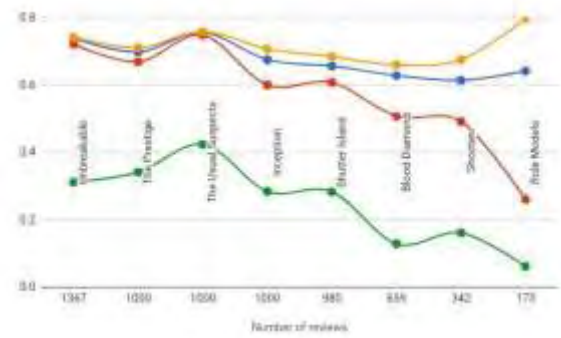
(a) BOW



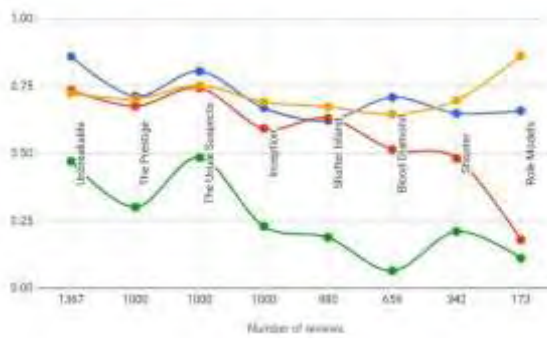
(b) Mix



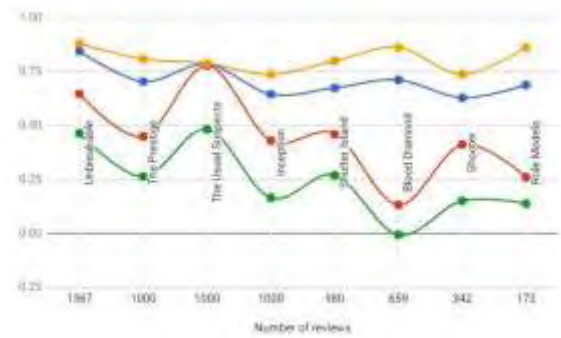
(c) DAMOT BOW



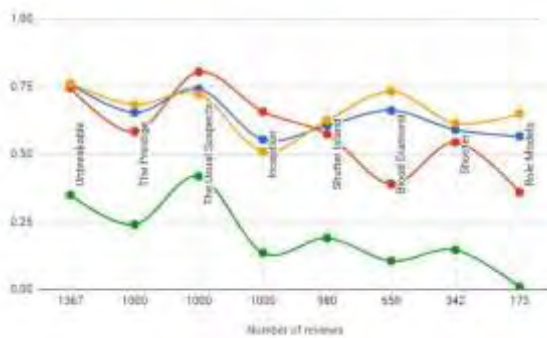
(d) DAMOT Mix



(e) Boosted BOW

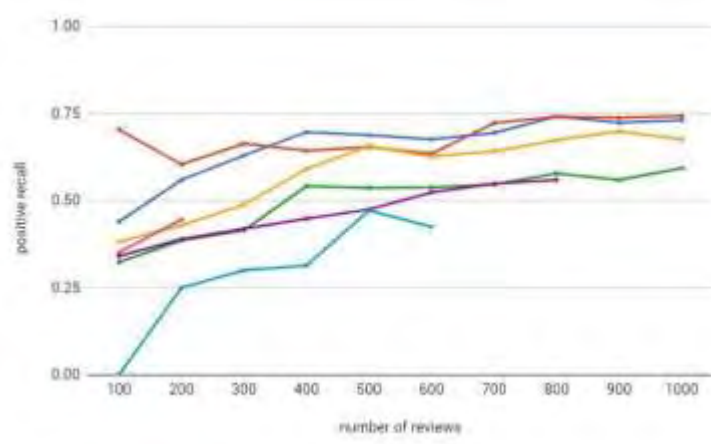


(f) Boosted Mix

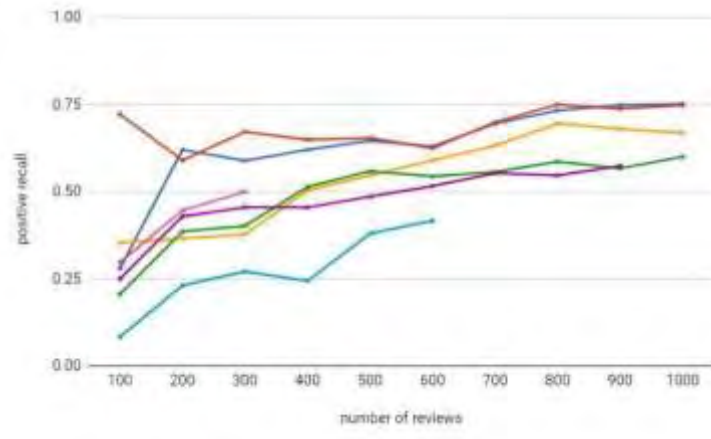


(g) Boosted ROS Mix

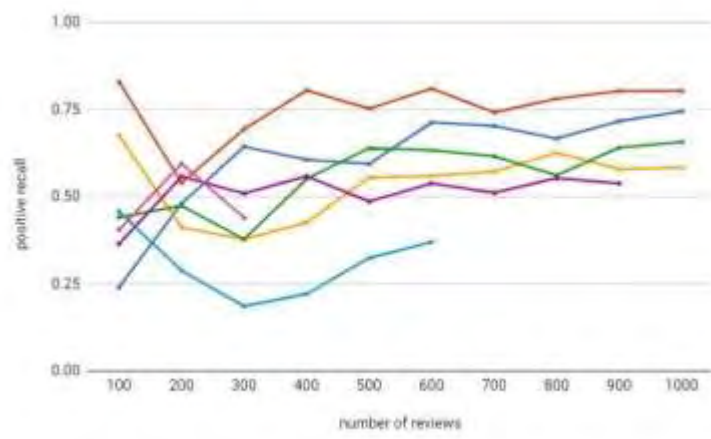
Figure 4.2: The performances of the models for individual dataset



(a) DAMOT BOW



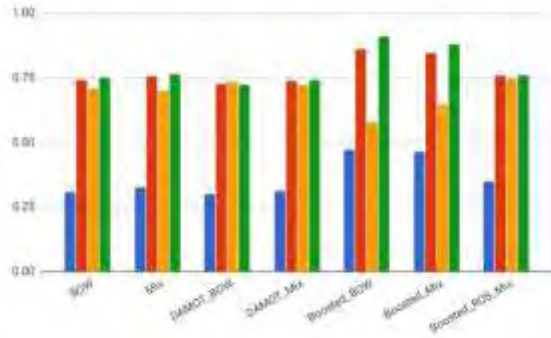
(b) DAMOT Mix



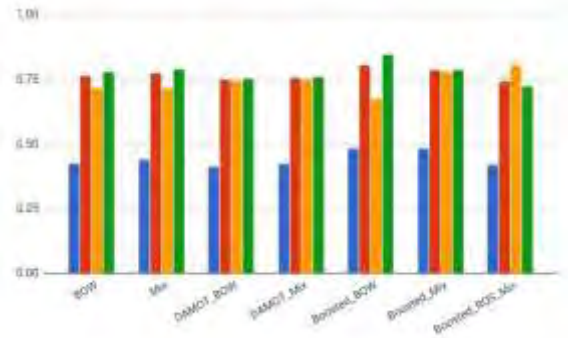
(c) Boosted ROS Mix

Figure 4.3: The change in performances of the models with the size of the datasets

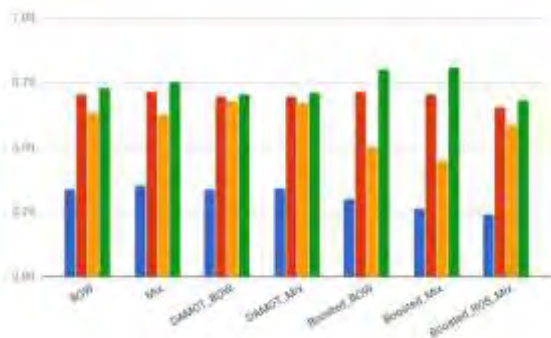
● Kappa
 ● Accuracy
 ● Positive Recall
 ● Negative Recall



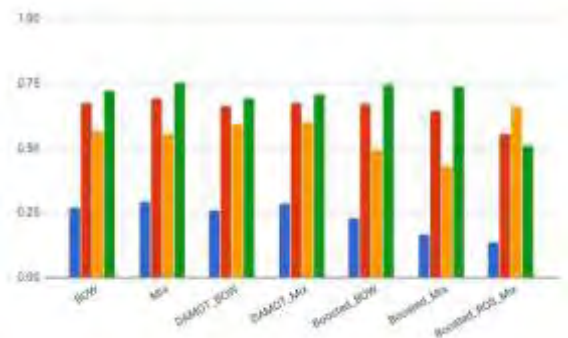
(a) Unbreakable



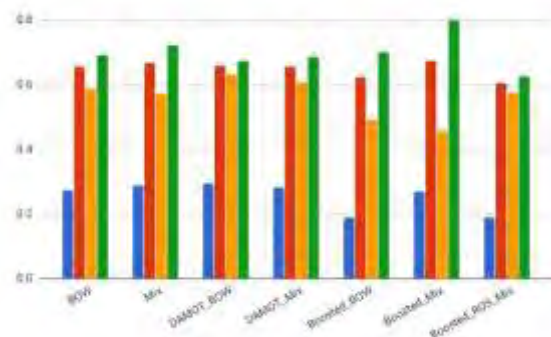
(b) The Usual Suspects



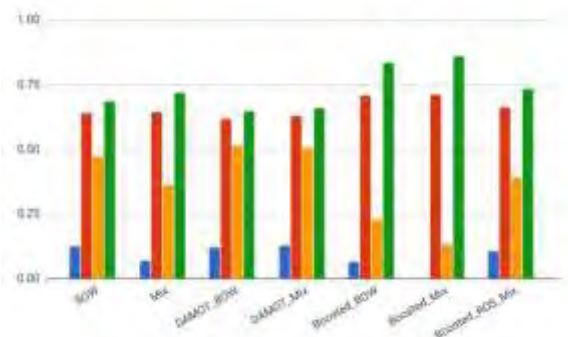
(c) The Prestige



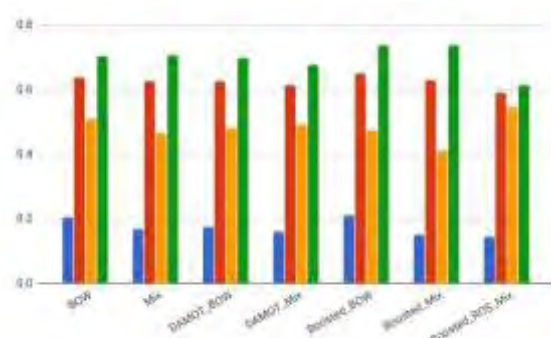
(d) Inception



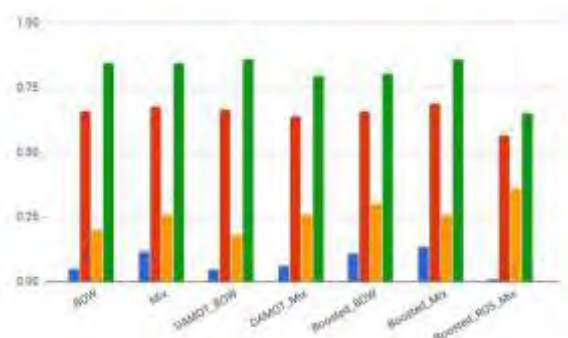
(e) Shutter Island



(f) Blood Diamond



(g) Shooter



(h) Role Models

Figure 4.6: Performance of the models

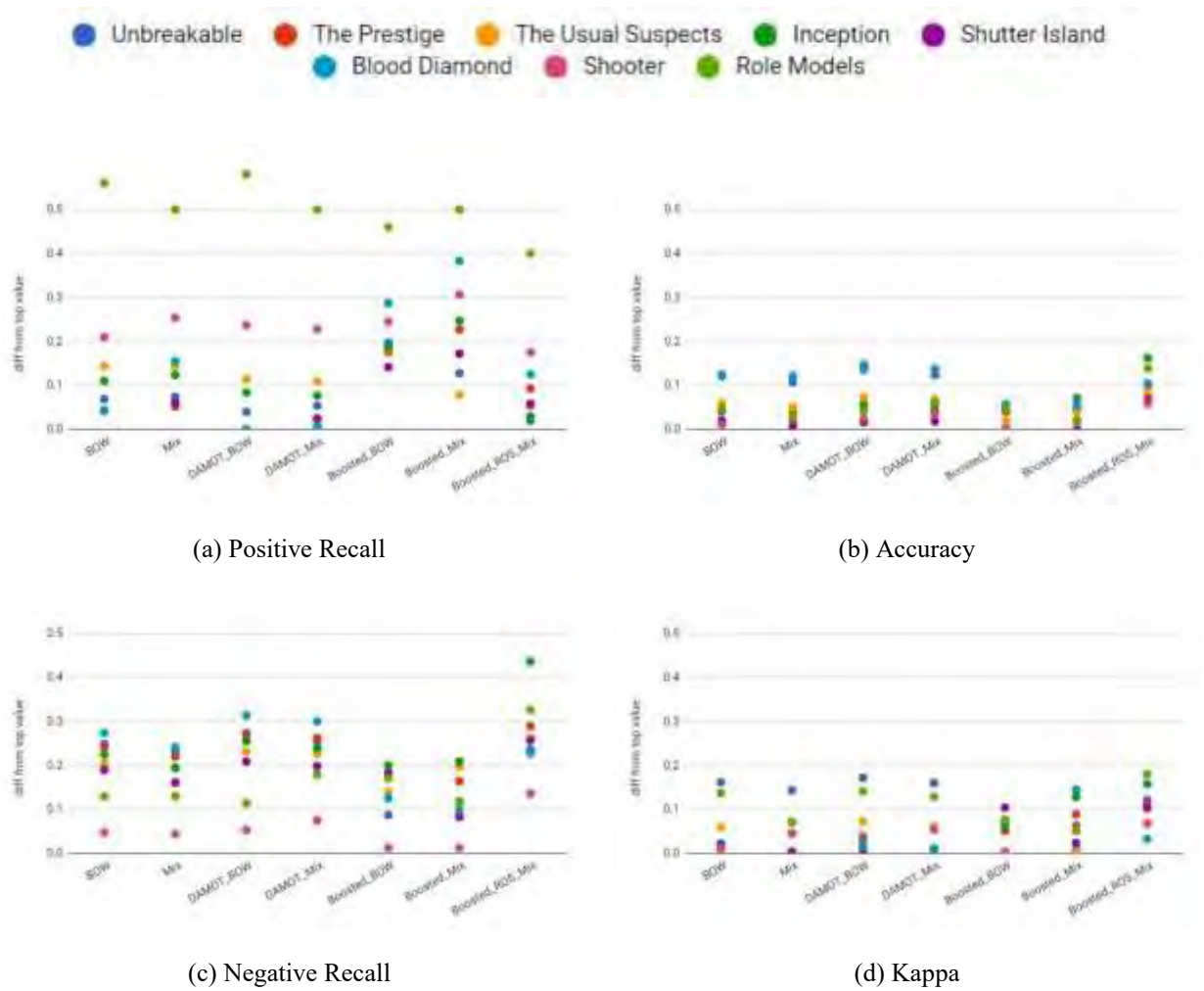


Figure 4.7: Deviation of the models from the top value in each performance metric for individual movies

Table 4.6: Result of Wilcoxon signed-rank test between DAMOT BOW and other models

Performance Metrics	N	R+	R-	T _{value}	Hypothesis
BOW					
Positive Recall	8	31	4	4	Reject for DAMOT BOW
Accuracy	8	3	33	3	Reject for BOW
Negative Recall	8	2	34	2	Reject for BOW
Kappa	8	8	28	8	Accept Null Hypothesis
Mix					
Positive Recall	8	29	7	7	Accept Null Hypothesis

Accuracy	7	0	28	0	Reject for Mix
Negative Recall	8	2	34	2	Reject for Mix
Kappa	8	10	26	8	Accept Null Hypothesis
DAMOT Mix					
Positive Recall	8	19.5	16.5	16.5	Accept Null Hypothesis
Accuracy	8	14	22	14	Accept Null Hypothesis
Negative Recall	8	15	21	0	Accept Null Hypothesis
Kappa	8	26	8	10	Accept Null Hypothesis
Boosted BOW					
Positive Recall	8	32	4	4	Reject for DAMOT BOW
Accuracy	8	6	30	6	Reject for Boosted BOW
Negative Recall	8	4	32	4	Reject for Boosted BOW
Kappa	8	15	21	15	Accept Null Hypothesis
Boosted Mix					
Positive Recall	8	32	4	4	Reject for DAMOT BOW
Accuracy	8	4	32	4	Reject for Boosted Mix
Negative Recall	7	0	28	0	Reject for Boosted Mix
Kappa	8	20	16	16	Accept Null Hypothesis
Boosted ROS Mix					
Positive Recall	8	15	21	15	Accept Null Hypothesis
Accuracy	8	29	7	7	Accept Null Hypothesis
Negative Recall	8	27	9	9	Accept Null Hypothesis
Kappa	8	30	6	4	Reject for DAMOT BOW

Table 4.7: Result of Wilcoxon signed-rank test between DAMOT Mix and other models.

The comparison with DAMOT BOW is already provided in table 4.6

Performance Metrics	N	R+	R-	T _{value}	Hypothesis
BOW					
Positive Recall	8	34	2	2	Reject for DAMOT Mix
Accuracy	7	1	27	0	Reject for BOW

Negative Recall	8	0	36	0	Reject for BOW
Kappa	8	27	9	9	Accept Null Hypothesis
Mix					
Positive Recall	7	28	0	0	Reject for DAMOT Mix
Accuracy	8	0	36	0	Reject for Mix
Negative Recall	8	0	36	0	Reject for Mix
Kappa	8	8	28	8	Accept Null Hypothesis
Boosted BOW					
Positive Recall	8	34	2	2	Reject for DAMOT Mix
Accuracy	8	5	31	5	Reject for Boosted BOW
Negative Recall	8	0	36	0	Reject for Boosted BOW
Kappa	8	18	18	18	Accept Null Hypothesis
Boosted Mix					
Positive Recall	8	33	2	2	Reject for DAMOT Mix
Accuracy	8	5	31	5	Reject for Boosted Mix
Negative Recall	8	0	36	0	Reject for Boosted Mix
Kappa	8	21	15	15	Accept Null Hypothesis
Boosted ROS Mix					
Positive Recall	8	16	20	16	Accept Null Hypothesis
Accuracy	8	30	6	6	Accept Null Hypothesis
Negative Recall	8	29	7	7	Accept Null Hypothesis
Kappa	8	32	4	4	Reject for DAMOT Mix

Table 4.8: Result of Wilcoxon signed-rank test between Boosted ROS Mix and other models. The comparison with DAMOT BOW and DAMOT Mix is already provided in table 4.6 and 4.7

Performance Metrics	N	R+	R-	T _{value}	Hypothesis
BOW					
Positive Recall	8	26	10	10	Accept Null Hypothesis
Accuracy	8	3	33	3	Reject for BOW
Negative Recall	8	3	33	3	Reject for BOW
Kappa	8	3	33	8	Reject for BOW
Mix					
Positive Recall	8	33	3	3	Reject for Boosted ROS Mix
Accuracy	8	3	33	3	Reject for Mix
Negative Recall	8	2	34	2	Reject for Mix
Kappa	8	6	30	6	Accept Null Hypothesis
Boosted BOW					
Positive Recall	8	36	0	0	Reject for Boosted ROS Mix
Accuracy	8	0	36	0	Reject for Boosted BOW
Negative Recall	8	0	36	0	Reject for Boosted BOW
Kappa	8	3	33	3	Reject for Boosted BOW
Boosted Mix					
Positive Recall	8	36	0	0	Reject for Boosted ROS Mix
Accuracy	8	0	36	0	Reject for Boosted Mix
Negative Recall	8	0	36	0	Reject for Boosted Mix
Kappa	8	20	6	36	Reject for Boosted Mix

Table 4.9: Result of Wilcoxon signed rank-test between DAMOT Mix and DAMOT BOW for datasets with more than 500 reviews

Performance Metrics	DAMOT BOW				
	N	R+	R-	T _{value}	Hypothesis
Kappa	6	18	3	3	Accept Null Hypothesis
Accuracy	6	20	1	1	Reject for DAMOT Mix
Positive F-measure	6	10	4	4	Accept Null Hypothesis
Positive Precision	6	20	1	1	Reject for DAMOT Mix
Positive Recall	6	3.5	17.5	3.5	Accept Null Hypothesis
Negative F-measure	6	21	0	0	Reject for DAMOT Mix
Negative Precision	5	7	8	7	Accept Null Hypothesis
Negative Recall	6	21	0	0	Reject for DAMOT Mix

4.7 Summary

In this chapter, we presented an empirical analysis of our method and each of its major components on eight IMDb movie datasets. Based on the average performance on the eight datasets, we choose BOW, Mix, DAMOT BOW, DAMOT Mix, Boosted BOW, Boosted Mix, and Boosted ROS Mix models among others to be analyzed further. The reason is that these models outperform others according to one or more performance metrics. Then, we plot four significant performance metrics: kappa, accuracy, positive recall, and negative recall, of these models against each of the individual movie dataset. We also plot these models' deviations in these performance metrics from the top value of that metric for each movie. Both comparisons provide similar intuition about the models. Adaboost, in all models except Boosted ROS Mix, rises negative recall and accuracy on a large scale with a penalty of very low positive recall. On the other hand, DAMOT BOW, DAMOT Mix and Boosted ROS Mix achieve higher positive recall without deviating much from the top accuracy and negative recall. As the cost of misclassifying positive

instance is significantly higher than misclassifying a negative instance, positive recall has been given more priority than other performance metrics. According to Wilcoxon signed-rank test, DAMOT BOW and Boosted ROS Mix fail to obtain leading performance in positive recall than the baseline methods, Mix and BOW, respectively. On the other hand, DAMOT Mix defeats all the baseline methods having significantly leading positive recall. It also beats Boosted ROS Mix in accordance with kappa. Moreover, DAMOT Mix leads significantly in accuracy, positive precision, negative f-measure, and recall in comparison with DAMOT BOW. So we can come to a conclusion that the combination of mix features and our innovative oversampling method, DAMOT, performs better than other models by achieving higher positive recall as well as balancing other metrics.

Chapter 5

Conclusion

5.1 Conclusion

Classifying spoiler in text is a more challenging task than any other typical text classifications. In this thesis, we have addressed several of these challenges and designed the architecture of our method taking each of these into account.

Feature extraction is one of the most critical phases of text classification. We employed syntactically related words, called dependency pairs, along with BOW as features in order to extract the context of the movies effectively. Another property of this problem is imbalanced class distribution. We proposed a novel oversampling technique, DAMOT, to resolve this property. No other existing approaches have addressed data imbalance property for spoiler detection so far. Moreover, we used adaboost to boost the performance of our proposed method even more.

Our models have been tested on eight datasets of IMDb movie reviews. Experimental results have revealed that DAMOT has achieved good result for all datasets according to the performance metrics. It successfully improves positive recall and balances other metrics at the same time. Adaboost, improved accuracy, but impaired positive recall which failed to serve our purpose. Overall, DAMOT with mix features provided consistent performance with statistically significant leading positive recall than others.

5.2 Future Work

Sometimes there are different perspectives of people on whether a certain piece of information has the potential to spoil the enjoyment or not. Moreover, not all the facts or details have the similar level of effect on the audience. Those spoiler that may or may not have any negative effect, are usually called mild spoilers. These are difficult to label and typically the cause of higher inaccuracy. In this thesis, we consider the spoiler detection problem as a binary classification task. A fascinating variation of this task can be converting this binomial task into multinomial task. So

we can break down the two classes into more, such as “critical spoiler”, “moderate spoiler”, “mild spoiler”, “non-spoiler”. This variant will assist to achieve better performance and intuition on the classes.

Our experimental result shows that adaboost does not essentially aid to detect more positive instances. We may tweak the boosting algorithm in such way so that it focuses more on the wrongly classified positive instances to boost the positive recall.

In social networks, such as facebook and various review aggregation websites like IMDb, Rotten Tomatoes etc., the posts regarding movies, books, tv-shows are usually quite long. The whole post or review does not typically contain spoiler material rather a very few sentences do. Labeling the full review as spoiler may deprive audience from non-spoiler important information. So we can target to detect the exact sentences that contain spoilers in a post or a review.

REFERENCES

- [1] D. Zillmann, “Anatomy of suspense,” in *The Entertainment Functions of Television* (P. H. Tannenbaum, ed.), ch. 6, pp. 133–163, Lawrence Erlbaum Associates, 1980.
- [2] D. Zillmann, “The logic of suspense and mystery,” in *Responding to the Screen: Reception and Reaction Processes* (J. Bryant and D. Zillmann, eds.), ch. 12, pp. 281–303, Lawrence Erlbaum Associates, 1991.
- [3] J. D. Leavitt and N. J. S. Christenfeld, “Story spoilers don’t spoil stories,” vol. 22, pp. 1152–1154, 2013.
- [4] J. D. Leavitt and N. J. S. Christenfeld, “The fluency of spoilers: Why giving away endings improves stories,” vol. 3, no. 1, pp. 93–104, 2013.
- [5] B. K. Johnson and J. E. Rosenbaum, “Spoiler alert: Consequences of narrative spoilers for dimensions of enjoyment, appreciation, and transportation,” vol. 42, pp. 1068–1088, 2015.
- [6] W. H. Levine, M. Betzner, and K. S. Autry, “The effect of spoilers on the enjoyment of short stories,” *Discourse Processes*, vol. 53, no. 7, pp. 513–531, 2016.
- [7] R. Ebert, *Critics Have No Right To Play Spoiler*. 2005.
- [8] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 1263–1284, 2009.
- [9] N. Chawla, K. Bowyer, L. O. Hall, and W. Philip Kegelmeyer, “Smote: Synthetic minority over-sampling technique,” *J. Artif. Intell. Res. (JAIR)*, vol. 16, pp. 321–357, 2002.
- [10] H. Han, W. Wang, and B. Mao, “Borderline-smote: A new over-sampling method in imbalanced data sets learning,” in *Proceedings of Advances in Intelligent Computing*, pp. 878–887, 2005.
- [11] H. He, Y. Bai, E. A. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning,” in *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 1322–1328, 2008.
- [12] M. Kubat and S. Matwin, “Addressing the curse of imbalanced training sets: Onesided selection,” in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186, 1997.
- [13] C. Elkan, “The foundations of cost-sensitive learning,” in *Proceedings of the 17th international joint conference on Artificial intelligence*, pp. 973–978, 2001.
- [14] Z.-H. Zhou and X.-Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

- [15] N. V. Chawla, “Editorial: special issue on learning from imbalanced data sets,” *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, vol. 6, pp. 1–6, 2004.
- [16] Y. Sun, M. S. Kamel, and Y. Wang, “Boosting for learning multiple classes with imbalanced class distribution,” in *Proceeding ICDM '06 Proceedings of the Sixth International Conference on Data Mining*, pp. 592–602, 2006.
- [17] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, “Cost-sensitive boosting for classification of imbalanced data,” *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.
- [18] P. Domingos, “Metacost: a general method for making classifiers cost-sensitive,” in *KDD '99 Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 155–164, 1999.
- [19] Z.-H. Zhou and X.-Y. Liu, “On multi-class cost-sensitive learning,” in *Proceedings of the 21st national conference on Artificial intelligence*, pp. 567–572, 2006.
- [20] M. Z. Kukar and I. Kononenko, “Cost-sensitive learning with neural networks,” in *Proceedings of the 13th European Conference on Artificial Intelligence*, pp. 445–456, 1998.
- [21] S. Nakamura and K. Tanaka, “Temporal filtering system to reduce the risk of spoiling a user’s enjoyment,” in *Proceedings of the 12th International Conference on Intelligent User Interfaces*, pp. 345–348, 2007.
- [22] J. Golbeck, “The twitter mute button: A web filtering challenge,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2755–2758, 2012.
- [23] K. Maeda, Y. Hijikata, and S. Nakamura, “A basic study on spoiler detection from review comments using story documents,” in *International Conference on Web Intelligence*, pp. 572–577, 2016.
- [24] J. Boyd-Graber, K. Glasgow, and J. S. Zajac, “Spoiler alert: Machine learning approaches to detect social media posts with revelatory information,” in *Proceedings of the 76th ASIS&T Annual Meeting: Beyond the Cloud: Rethinking Information Boundaries*, ASIST '13, pp. 1–9, 2013.
- [25] S. Jeon, S. Kim, and H. Yu, “Spoiler detection in tv program tweets,” *Information Science*, vol. 329, pp. 220–235, 2016.
- [26] B. Markines, C. Cattuto, and F. Menczer, “Social spam detection,” in *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, pp. 41–48, 2009.

- [27] L. Liu and K. Jia, “Detecting spam in chinese microblogs - a study on sina weibo,” in *Proceedings of the Eighth International Conference on Computational Intelligence and Security*, pp. 578–581, 2012.
- [28] D.-H. Park, E.-A. Cho, and B.-W. On, “Social spam discovery using bayesian network classifiers based on feature extractions,” in *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1808–1811, 2013.
- [29] P.-C. Lin and P.-M. Huang, “A study of effective features for detecting long-surviving twitter spam accounts,” in *Proceedings of the 15th International Conference on Advanced Communications Technology (ICACT)*, pp. 841–846, 2013.
- [30] A. Sureka, “Mining user comment activity for detecting forum spammers in youtube,”
03 2011.
- [31] A. Pak and P. Paroubek, “Twitter as a corpus for sentiment analysis and opinion mining,” in *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, vol. 10, pp. 1320–1326, 2010.
- [32] M. S. Neethu and R. Rajasree, “Sentiment analysis in twitter using machine learning techniques,” in *Proceedings of the Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–5, 2013.
- [33] T. Sahni, C. Chandak, N. R. Chedeti, and M. Singh, “Efficient twitter sentiment classification using subjective distant supervision,” in *Proceedings of the 9th International Conference on Communication Systems and Networks (COMSNETS)*, pp. 548–553, 2017.
- [34] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, “Sentiment analysis of twitter data,” in *Proceedings of the Workshop on Languages in Social Media*, pp. 30–38, 2011.
- [35] P.-W. Liang and B.-R. Dai, “Opinion mining on social media data,” in *Proceedings of the IEEE 14th International Conference on Mobile Data Management*, pp. 91–96, 2013.
- [36] P. Gamallo and M. Garcia, “Citius: A naive-bayes strategy for sentiment analysis on english tweets,” in *Proceedings of the 8th International Workshop on Semantic Evaluation*, pp. 171–175, 01 2014.
- [37] D. Davidov, O. Tsur, and A. Rappoport, “Enhanced sentiment learning using twitter hashtags and smileys,” in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pp. 241–249, 2010.
- [38] J. Kamps, M. Marx, R. Mokken, and M. Rijke de, “Using wordnet to measure semantic orientation of adjectives,” in *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 01 2004.

- [39] “Ensemble of feature sets and classification algorithms for sentiment classification,”
Information Sciences, vol. 181, no. 6, pp. 1138 – 1152, 2011.
- [40] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.
- [41] T. L. Griffiths, M. Steyvers, D. M. Blei, and J. B. Tenenbaum, “Integrating topics and syntax,” in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, pp. 537–544, 2004.
- [42] H. M. Wallach, “Topic modeling: Beyond bag-of-words,” in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 977–984, 2006.
- [43] S. Guo and N. Ramakrishnan, “Finding the storyteller: automatic spoiler tagging using linguistic cues,” in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 412–420, 2010.
- [44] I. Titov and R. McDonald, “Modeling online reviews with multi-grain topic models,” in *Proceedings of the 17th International Conference on World Wide Web*, pp. 111–120, 2008.
- [45] W. X. Zhao, J. Jiang, H. Yan, and X. Li, “Jointly modeling aspects and opinions with a maxent-lda hybrid,” in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 56–65, 2010.
- [46] B. Lu, M. Ott, C. Cardie, and B. K. Tsou, “Multi-aspect sentiment analysis with topic models,” in *Proceedings of the IEEE 11th International Conference on Data Mining Workshops*, pp. 81–88, 2011.
- [47] I. B'iro, J. Szab' o, and A. A. Bencz' ur, “Latent dirichlet allocation in web spam filtering,” in *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web*, pp. 29–32, 2008.
- [48] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics*, vol. 1, no. 6, pp. 80–83, 1945.
- [49] M.-C. de Marnee and C. D. Manning, “Stanford typed dependencies manual,” 2008.
- [50] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.
- [51] J. R. Landis and G. G. Koch, “The measurement of observer agreement for categorical data,” *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.