**F21RP Report**

**Research Methods and Project Planning**

**An Innovative Approach on Spam Reviews**

**Instructor**

**Dr. Mehran Sharghi**

**Author**

**Milad Roshanaei Zadeh**

**MSc Software Engineering student at Heriot-Watt University**

**Edinburgh**

**Spring 2024**

**Chapter One**

**Introduction**

**Introduction:**

In recent years, opinion-sharing platforms have been increasing exponentially and many websites allow users to share their own experiences, emotions, attitudes, and feelings to help future customers who want to get a service or product already tested and approved. Consequently, posting reviews affects significantly consumers' buying decisions. Unfortunately, as anyone can write anything and get away with it, a rise in the number of opinion spams has been witnessed [S].

In some cases, Companies use a “Water Army” to share positive perspectives about their own products, or even share deceptive opinions about their competitors’ products. These deceptive opinions, spam opinions, or spam reviews, may harm a company's brand name and result in financial loss for retailers and service providers. Unfortunately, there are no restrictions on sharing reviews on social media and everyone is allowed to share their opinions without any limits. So, it is important to detect these reviews as soon as possible without restricting users’ authorities.

**Motivations:**

These days, due to the lack of time, people prefer to shop online. Whenever it comes to using platforms like Amazon, Alibaba, or eBay, users usually find themselves in a dilemma. Since there are many similar products, users need to refer to some qualitative and quantitative attributes to make the final decision. But what if these attributes are manipulated by some profit-seeking spammers to cloud users’ judgement?

As a person who shops online most of the times, I have always been struggling with making the correct decision. I wholeheartedly want to make a safe shopping environment for users to feel comfortable reading a product reviews and judge based on product’s rating.

**Problem definition:**

Based on Lim [S], there are 4 models to detect spam reviews according to the rating behaviour of reviews:

1. Publish deceptive spam reviews targeting a certain product: The more times a reviewer repeatedly rates the same product, and the greater the difference between the ratings each time, the higher the similarity of the review content, the reviewer is more likely to be spammer.
2. The more times a reviewer repeatedly rates the same product: and the greater the difference between the ratings each time, the higher the similarity of the review content, the reviewer is more likely to be spammer.
3. General rating deviation: The deviation between a reviewer’s rating of a product and the average of all reviewers’ ratings. The larger the deviation, the more likely it is a spammer.
4. Early rating deviation: Like general rating deviation, early rating deviation is related to the chronological order of ratings.

But what if users want to post the first reviews of a product?

Basically, the most difficult and important part of spam review detection is on the early stage of product deployment. In the early stages, there are too few reviews, so this difficulty lies in the lack of behavioural characteristics of new reviewers. This problem is called Cold Start Problem.

One significant challenge in the field of fake review detection is the absence of large, standardized, and diverse datasets that accurately represent the complexity of fake review patterns across various domains and platforms. Indeed, most studies widely used datasets such as Yelp or OTT datasets, which may not fully encapsulate the spectrum of fake review characteristics found in other contexts. In addition, the choice of dataset has a substantial impact on experimental outcomes and the reported performance of detection models. The specific attributes of the dataset, such as the labelling of genuine and fake reviews, the variability in writing styles, and the intricacies of domain-specific language, can significantly influence the efficacy of machine learning algorithms. Consequently, the reported precision, recall, and other metrics may vary extensively based on the unique attributes of the chosen dataset.

Difficulties in obtaining accurately labelled fake review datasets result in many problems. First, supervised approaches are incompatible with the design of fake review detection models. Therefore, unsupervised machine learning techniques are frequently adopted to spot fake reviews and/or reviewers based on some assumptions given in advance. Secondly, due to the lack of ground-truth data, it is hard to evaluate the “real” performance of an unsupervised machine learning model. As a result, it’s challenging to select the appropriate models for detecting fake reviews in each review dataset. Lastly, due to the diversity of review spam strategies, no single detection model can stop all the spam. Therefore, unsupervised models based on different assumptions of spam characteristics must be combined to improve the detection performance.

**Spam detection and Spam reviews detection**:

Although these two concepts aim for different purposes, there are many similarities between them:

1. Text Analysis Techniques:

Both spam detection and spam reviews detection systems use text analysis techniques such as Natural Language Processing (NLP). NLP helps in understanding the semantics and context, which are necessary for spam identification.

1. Feature Extraction:

Features such as word frequency, n-grams, and syntactic patterns are used to train machine-learning models for spam and spam reviews detection.

1. Machine Learning Model:

Models such as supervised classifiers like Naïve Bayes and Random Forests are used for both spam and spam reviews detection.

1. Data Processing:

Processing steps like tokenization and spell checking are important for cleaning data before feeding it into models. This process helps with accuracy and reliability of detection systems.

1. Adaption to Evolving Patterns:

Spams and Spam reviews evolve overtime, which means spammers change their approaches after a while. So, adaptive techniques are needed for maintaining effectiveness of detection system.

**Chapter Two**

**Literature Review**

**Related Approaches:**

1. **Collaborative Filtering:**

Collaborative Filtering is one of the most popular techniques used in recommender system. The underline intuition of Collaborative Filter is simple. If user X and user Y like the same product A, then it is likely X and Y will share the same interest on a different item.

Collaborative filtering (CF) is one of the most popular techniques used in recommender system. CF is based on mutual interests, meaning that if user X and Y like product A and user X also likes product B it is likely that user Y will like product B as well. This technique helps detection systems to be able to extract some semi-legit data for training from people that are likely to use the product.

1. **Graph based recommender system:**

Knowledge graphs have been proven to be useful in recommender systems. Having a multi-dimensional vector for each item, can help us to easily calculate the similarity between two items.

Network structure for Cold Start problem can cover both collaborative and content information.

1. **Abnormal detection using information graph:**

Abnormal detection can help detection systems to acquire a graph that can be treated as a time-evolving dynamic network. Abnormal can be detected using nodes, edges, and sub-graphs.

As spammers usually go to opposite direction of the crowd, detecting abnormal can be very valuable.

**Cold Start Problem Overview**:

In nutshell, Cold Start Problem is referred to the challenge of spamming identification in early stages. For example, when spammer has set up a new account, or a new product has been released online, there are limited or no historical data to train the machine-learning model.

Here are some key points that can cause cold start problem:

1. Lack of Data:

In early stages of detection process (e.g. when user has just created an account), systems may lack sufficient historical data on user behaviour to train the machine-learning model.

1. Lack of Features:

Spam detection models may have limited features or patterns to identify spam messages, while there are no historical data. Old features that work with sender’s reputation or content analysis might not be available.

1. Evolving Spamming Strategies:

Merging some of techniques, or strategy evolutions, can cause spam bypass and higher rates of false detection.

**Formulise Cold Start Problem**:

For finding the most appropriate approach, evaluating purposes, and looking closer to the problem, we need to give the problem a mathematical appearance. Here is the definition of the problem in mathematical language:

1. Let be the set of messages that are going to be fed into our model and be set of labels that defines a message as ham or spam. So, our training data () will be like:
2. Now we can define new data as that is not labelled yet.
3. Extracted features will be represented as feature vectors which is the feature vector that is extracted from . By using bag-of-word or TF-IDF techniques we can extract these features. We can define as below:

So, for new message there is a new feature vector called

1. Now that we have the features for each message, we need classification model to make the final decision for the detection system.
2. Cold Start Scenario: The most challenging issue in Cold Start problem is that for there is no accurate available for labelling training data as new spamming pattern shows up.

**A Standard Approach for Spam Detection**:

Spam (in a form that we know today) detection has been an issue for users since mid-1990s. Since the foundation of machine-learning and deep-learning one approach has brought the highest accuracy in comparison with other methods. We call this approach “The Standard Approach” and we try to evolve it to reach our desired model. Here are steps of the standard approach:

1. Gather and choose datasets.
2. Feature extraction: Afterwards, feed the feature vectors into machine learning model.
3. Classifier Decision: Afterwards, give the input to our system.
4. Label the input.

A diagram of a flowchart

Description automatically generated

For different issues, these steps might change but the foundation of the solution will remain untouched.

**Common Strategies**:

1. Use Data from other sources:

Using external data or pre-trained models can provide initial training data and patterns until enough user data is accessible.

1. Active Learning:

Using the most informative data and label them for training spam detection model.

1. Define Rules:

Based on previous spams, we can define some rules for our system to be more sensitive on some common spam keywords or suspicious domains. This technique can make the system more rigorous in early stages to prevent false detection.

1. Semi-supervised Learning:

This technique is a combination of unsupervised learning with limited labelled data to improve spam detection while more data becomes available.

**Proposed System**:

1. Design and Methodology:

Design of this system is based on standard approach, although extra steps are added for more accuracy and reliability. As the first necessary step, we need a preprocess on input dataset to remove or repair noisy data. Afterwards, we select our machine-learning model and feature extraction technique. Finally, we train our system and evaluate its decisions. System design is presented below.

A diagram of a process

Description automatically generated

This is the design for spam reviews detection, to address cold start problem we decided to add some rules to the system as well as the knowledge than it gains through training phase. The reason behind rules definition is that we need to minimise the error rate in early stages of a product release. So, the complete design map of our system will be like:

A diagram of a system

Description automatically generated

1. Description and Definition:

In this section, we define each step of system, and give a brief explanation about them.

2.1. Input Dataset:

For training purposes we used an Amazon product review dataset that could be found in: <https://www.kaggle.com/datasets/naveedhn/amazon-product-review-spam-and-non-spam/data>

2.2. Tokenisation:

Tokenization in the process of breaking a sentence into multiple words. These words are called tokens. Breaking messages into tokens can provide a more digestible and accurate input for our machine-learning model.

2.3. Stemming:

Stemming is the process of converting variations of a word to the root. For instance, the words “Chocolate”, “Choco”, and “Chocolatey” will be converted to the word “Chocolate”. Stemming will help us to get the original dictionary for algorithm.

2.4. “Stop Words” Removal:

Stop words are repetitive words like “the”, “a”, “an”, or “at” that are commonly used in all reviews. Stop words do not provide any useful information for our system as we can find them in both ham and spam messages. So, we need to remove them before beginning of training phase.

2.5. POS Tagging:

POS or Part-Of-Speech tagging is the process of tagging a word with its grammatical position in a sentence. For instance, a word might be a “noun”, “adjective”, or “verb”.

2.6. Feature Extraction:

Sentiment analysis is the process of determining a sentence expression. For example, whether a sentence is positive, negative, or neutral.

2.7. Rules definition:

To increase the accuracy and reliability of our system in early stages, we decided to define some rules using acquired results from similar situations. These rules are sensitive to words that increase the spam likelihood by a specific threshold. For example, if there are “win” and “prize” words in a sentence, the spam probability of the input will be calculated. If the likelihood reach a specific amount, the data will be labelled as spam. So, we can hard-label this input without training the algorithm and use this input for training purposes.

2.8. Machine-learning Classification:

A Classifier is usually an algorithm that is responsible to label the input data. In our case, labels are either ham or spam. This means we need a binary classifier. We use Naïve Bayesian Classifier which is one of the most accurate and reliable classifiers in spam detection problems.

2.9. Detection Phase:

In this phase, we need to calculate the likelihood of input and make label it.

2.10. Clustering:

Clustering is used to detect spammers community.

2.11. Graph Evaluation:

To compare and measure performance graphs using factors like precision, F1-score, accuracy, and time parameters.

3. Why Rule Definition?

Defining rules gives us 3 privileges:

3.1. Rules are clear:

Rules are less complex than other methods like CF and recommender systems.

3.2. Rules are fast:

Rules provide training data for our system in early stages with least possible calculation.

3.3. Rules are easy to use:

In comparison with other methods like semi-supervised approach, rules definition is easier to use.

1. Rules Definition:

To have the most reliable and accurate rules for the system, we need to define a dictionary that maps words to a reasonable score. To define a proper dictionary, we gathered 1000 spam related words with scores between 0 to 100. Then we search in the message context for these words. But how are we going to calculate the probability? What is a proper threshold for this problem?

4.1. Bayes’ Law:

Bayes law describes the probability of an even based on prior knowledge and conditions that might be related to the event. In mathematical language, when we have two events and the probability of (Read A if B happens) will be stated as:

Let be the spam probability of the message, and be the probability of extracted features. We will have:

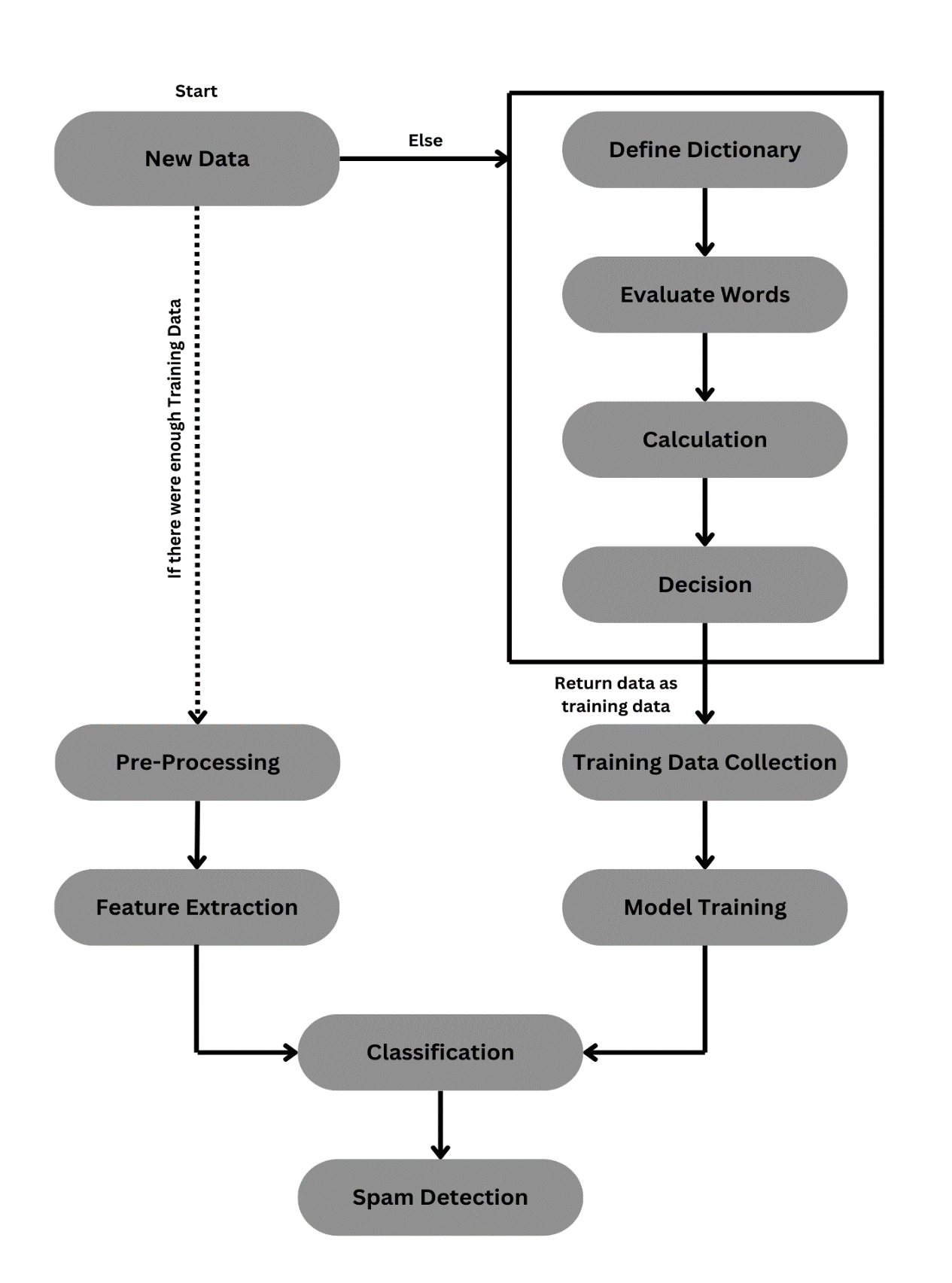
So, we need to calculate and compare it with our defined threshold.

4.2. Threshold:

Naïve Bayes classifier gets 0.5 or 50% as the reasonable threshold for making a labelling decision. We will use the same value for our rules system.

1. System’s Design:

Now that we have defined all required tools and reasons for our system, it is time to have a complete flowchart of our systems functionality.



1. Pseudocode:

**Evaluation:**

1. Pros:

1.1. Transparency:

Rule-based approaches are easy to understand as they are based on explicit rules rather than complex algorithms.

1.2. Flexibility:

Rules can be modified based on our needs and the cost of adjustment is by far lower than other approaches.

1.3. Robustness:

Rule-based approach preforms well with limited data and that makes it an excellent solution for cold start problem.

1. Cons:

2.1. Limited Scalability:

Having a high number of rules can be challenging as we need to modify and control them.

2.2. Depending on rules quality:

The effectiveness of our system heavily depends on quality of rules that are defined by domain experts.

2.3. Unable to manage specific relationships:

Rules cannot realise specific needs of users. That will lead our system to not be able to recommend best recommendations.