Amazon Books Review Sentiment Analysis Using SparkML

Team 1:
Nguyễn Bá Hà Anh - 20020273
Vũ Trung Kiên - 20021381
Nguyễn Diệu Quỳnh - 20021424
Trần Thanh Sơn - 20021432
Nguyễn Minh Trung - 20021456

***Abstract*— Today, we live in the Big Data age. Social networks, online shopping, mobile data are main sources generating huge text data by users. This "text data" will provide companies with useful insight on how customers view their brand and encourage them to make business strategies actively in order to maintain their trade. Hence, it is essential for the enterprises to analyse the sentiments of social media big data to make predictions. Sentiment analysis is the process of identifying the opinion or feeling expressed as positive, negative or neutral. Capturing the accurate sentiment of a review is a challenge. In this paper, we use different preprocessing techniques such as tokenization and stop words removal to eliminate noise. The preprocessed data is stored using feature selection techniques such as term frequency-inverse document frequency (TF-IDF). In this experiment we use 2 methods to vectorize: TF-IDF and Word2Vec. At the end of the vectorization stage, the feature vectors are used to train classifiers such as Logistic Regression (LR), Support Vector Machine (SVM) and Naive Bayes (NB) to classify the sentiment of Amazon book reviews. Finally, we present a comparison of the accuracy of the classifiers, and the sentiment of different books on different model sets.**

I. INTRODUCTION

In the era of smart devices, consumers increasingly search and evaluate products and services online. One of the popular and useful sources of information for readers is Amazon book reviews, where they can read reviews of the latest and most popular books from editors, authors, critics and readers. Amazon book reviews also introduce books selected by celebrities, award-winning books, books for book clubs and best books of the month. Amazon book reviews are a valuable source of information for those who love reading and want to discover new and interesting books. However, Amazon book reviews are also an example of big data, a concept that refers to data that are large and complex enough that they cannot be processed by traditional methods. Big data includes millions of reviews from different websites, and the number of reviews is increasing every day. To analyze and extract value from big data, modern technologies and architectures are needed, as well as powerful machine learning methods. One of the computing platforms for big data is Apache Hadoop and Apache Spark, designed to integrate machine learning systems to achieve high performance. Sentimental analysis is used in various places, for example: To analyse the reviews of a product whether they are positive or negative. Sentiment analysis is all about having people's real voice of a particular product, programs, organization, movies, news, events, problems and their characteristics. Social media monitoring apps in businesses rely primarily on sentiment analysis using machine learning to help them gain insights into mentions, brands and goods . Sentiment analysis is recognized as a problem of classification and it can also be solved by the method of machine learning techniques. The rest of this paper is laid out as follows. In Section 2, we begin with previous works to sentiment analysis by Spark ML. The core components of Apache Spark and Spark ML are then introduced in Section 3. In section 4, we provide an overview of the discussed research. The proposed methodology for sentiment analysis through big data analytics using spark ML is implemented in Section 5. Section 6 describes the experimental setup, and results. Finally, future work and conclusions of this paper are presented.

II. RELATED WORK

A. Reviews Sentiment Analysis

Sentiment analysis is the process of automatically identifying and extracting the subjective information conveyed in a text. When applied to products reviews datasets, sentiment analysis can help determine the overall opinion of customers

towards a particular product or service. By analyzing the sentiment expressed in these reviews, companies can gain valuable insights into the strengths and weaknesses of their products, as well as identify areas for improvement. Sentiment analysis algorithms can classify reviews as positive or negative based on the language used, and provide a sentiment score that indicates the degree of positivity or negativity expressed in the text. Many researchers have proposed their own methods to tackle this problems. Srunjan et al. [1] applied five classification methods, after having preprocessed the data using TF-IDF representation model, to classify the dataset into positive or negative class. Their preprocessing phase includes data cleaning, removal of HTML tags and URL, punctuation and special character removal, numbers and white-spaces removal, stop words removal, lowercasing and stemming. Following that, classifiers like K-Nearest Neighbours (KNN), Random Forest (RF), Naive Bayes (NB), Decision Trees and Support Vector Machine (SVM) were utilized to categorize the reviews. Their study shows that RF gives the best results and on average has the highest accuracy, while KNN scores the highest for two out of eight books in the dataset. In regards to the reviews classification problem, R. S. Jagdale et al. [2] also presented a comparison of two classifiers, NB and SVM, for Amazon products reviews dataset. By comparing the accuracy, NB classifier got 98.17% accuracy for Camera reviews and Support Vector Machine got 93.54% accuracy for Camera reviews. On a more aspect-based level, the works of Wang, Kai, et al. [3] and Chen, Xiao, et al. [4] have been done in 2020, both using a variation of graph attention networks. The study by Wang, Kai, et al. shows that the correlation between aspects and opinion words can be better established with R-GAT, and the performance of GAT and BERT are significantly improved consequently. Apart from classification, many authors have utilized sentiment analysis to approach other problems. Aljuhani S.A. and Alghamdi N.S. [5] developed a predictor for consumers' satisfaction on a mobile phone product based on the reviews. They carried out their experiments as follows: preprocessing the data, converting the data from text to vector representation with techniques such as bag-of-words, TF-IDF, Glove and word2vec then applying different machine learning algorithms. In addition, they used Lime technique to provide analytical reasons for the reviews being classified as either positive, negative or neutral. According to the findings, CNN with word2vec as a feature extraction method presents the most favorable outcomes with both balanced and unbalanced dataset. A different problem was approached in a study by Elmurngi E. I. and Gherbi A. [6], which is detecting unfair Amazon reviews. They compared multiple supervised machine learning algorithms, namely NB, Decision Tree (DT-J48), Logistic Regression (LR) and SVM for sentiment classification using three datasets of reviews. The results shows that the LR algorithm performs the best at classifying as well as detecting unfair reviews.

B. Sentiment Analysis with Apache Spark

Apache Spark is an open-source distributed computing system designed for processing and analyzing Big Data. We will provide a deeper understanding of this system in the next section. With the convenient built-in libraries, Apache Spark and especially Spark Machine Learning MLlib have been used to perform sentiment analysis tasks by various researchers. The majority of studies have focused on classifying reviews into positive and negative based on sentiment of the text. As the dataset was huge, Apache Spark Apache Hadoop was used by the authors in [7] to handle this issue. Three techniques, namely Linear SVC, LR, and NB, are utilized with Apache Spark's MLlib, to achieve an accuracy of over 80%. Upon implementing these methods, it was discovered that Linear SVC outperforms NB and LR in terms of efficiency. NB and LR, in addition to SVM, in MLlib were also applied for a large-scale dataset in the work of Ahmed, Hafiz Muhammad, et al. [8]. In their experiments, the SVM classifier has better performance than others. To tackle scalability and availability challenges in sentiment analysis on an e-commerce platform, the authors in [9] have employed PySpark and Spark NLP-based sentiment analysis utilizing resilient distributed datasets (RDDs). The RDD-based Spark is proved to enhance efficiency in terms of scalability, availability, and faster data collection. In all of the above studies, Spark ecosystem has become one of the keys for large-scale data distributed processing and analytics frameworks in the world, it achieves high performance for both batch and streaming data and it has easy-to-use APIs for operating on large datasets.

III. SPARK MACHINE LEARNING - MLLIB

A. Apache Spark

Apache Spark is a high-performance and versatile cluster computing platform.

On the speed side, Spark builds upon the widely used MapReduce model and enhances it to efficiently handle a broader range of computations, including interactive queries

and stream processing. Speed is crucial when working with large datasets as it determines the difference between interactive data exploration and waiting for minutes or hours. Spark's ability to perform computations in memory is one of its key features for achieving high speed. Moreover, it surpasses MapReduce in efficiency for complex applications that operate on disk.

In terms of generality, Spark is designed to address diverse workloads that previously required separate distributed systems. It accommodates batch applications, iterative algorithms, interactive queries, and streaming tasks within a single engine. This simplifies the combination of different processing types, a common requirement in production data analysis pipelines, and reduces the overhead of managing separate tools.

Spark emphasizes accessibility by providing user-friendly APIs in Python, Java, Scala, and SQL, along with extensive built-in libraries. It also seamlessly integrates with other Big Data tools. Notably, Spark can be deployed on Hadoop clusters and seamlessly interact with various Hadoop data sources, including Cassandra.

Apache Spark consists of several key components:

- Spark Core: It provides the foundation for all other Spark components. It includes the basic functionalities of Spark, such as task scheduling, memory management, and fault recovery.
- Spark SQL: This module enables working with structured and semi-structured data using SQL-like queries. It provides a DataFrame API for manipulating structured data and integrates with various data sources.
- Spark Streaming: It allows processing real-time streaming data. Spark Streaming ingests and processes data in mini-batches, enabling near real-time processing and analytics.
- Spark MLlib: MLlib is Spark's machine learning library, offering a wide range of algorithms and utilities for building and deploying machine learning models. It provides high-level APIs for ease of use.
- Spark GraphX: This component is designed for graph processing and analysis. It provides an API for expressing graph computation and includes various graph algorithms.
- Cluster Managers: Spark can integrate with various cluster managers like Apache Mesos, Hadoop YARN, and standalone mode. These cluster managers handle resource allocation and scheduling of Spark tasks on a cluster.

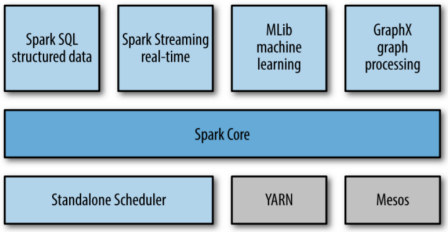

Fig. 1: Apache Spark Stack

These components work together to provide a unified and versatile platform for big data processing, analytics, machine learning, and graph processing. They enable efficient and scalable data processing across various use cases and deployment scenarios.

In summary, Apache Spark excels in delivering fast and versatile cluster computing capabilities. It extends the boundaries of traditional MapReduce, enabling efficient processing of diverse computation types, and streamlines the management of workloads that previously required separate systems. With its user-friendly interfaces and integration capabilities, Spark empowers users to effectively leverage its power in conjunction with other Big Data technologies.

B. Apache Spark Architechture for Big Data

Apache Spark is a powerful platform for analyzing big data, providing an improved alternative to the MapReduce model. Unlike the traditional MapReduce model, Spark avoids the need to write data to disk after every step. Instead, it efficiently collects and stores data in memory. Only when the memory is full, the excess data is spilled to the hard drive. This in-memory processing capability of Spark significantly boosts its processing speed. The architectural layout of Spark is illustrated in Figure 2, showcasing its components and their relationships.

The Spark System follows a master/worker architecture, where the master node oversees the worker nodes. When Spark is deployed on a cluster, the executors, primarily located on the worker nodes, are automatically created, and tasks are executed based on the instructions provided by the cluster manager. In the context of the Spark Session, the driver acts as the interface through which users transmit and receive instructions. The Spark Session serves as the central gateway for all communication. Spark revolves around the concept of RDD (Resilient Distributed Datasets), which enables various

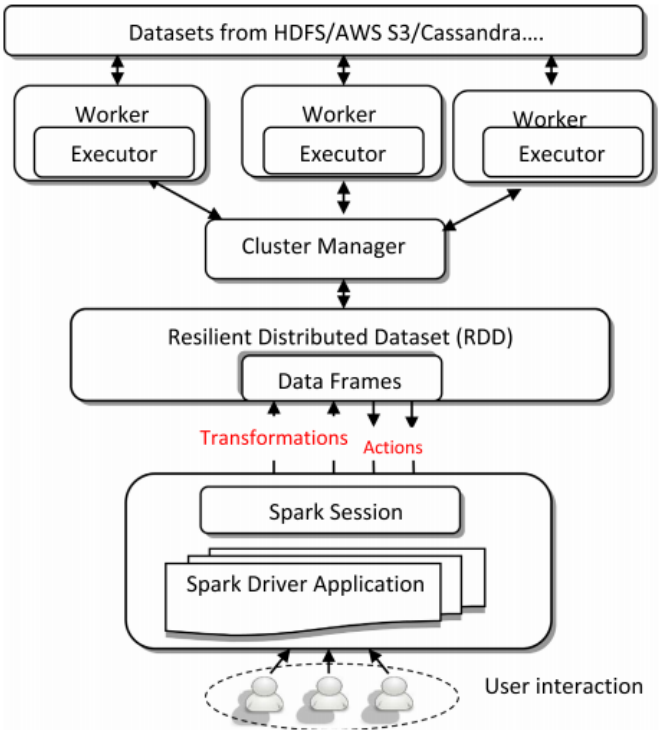

Fig. 2: Apache Spark Architechture

applications such as distributed data processing, support for multiple data sources, fault tolerance, and parallelism. Spark performs two core activities: transformations and actions. Transformations modify the RDDs by applying operations like mapping, joining, and key reduction to transform the input data and generate final outputs, while actions retrieve data from RDDs for further processing.

C. Spark MLlib

Spark MLlib is a powerful machine learning library provided by Apache Spark. It offers a wide range of tools and algorithms for scalable and distributed machine learning tasks. With MLlib, users can efficiently process and analyze large-scale datasets, making it suitable for big data environments. The library provides high-level APIs that are easy to use and enable developers to build and deploy machine learning models with ease.

One of the key features of Spark MLlib is its ability to leverage the distributed computing capabilities of Spark. It allows users to perform parallel and distributed computations across a cluster of machines, enabling faster and more efficient processing of large datasets. MLlib supports various machine learning algorithms, including classification, regression, clustering, collaborative filtering, and more.

MLlib also provides a rich set of feature transformers and utilities for data preprocessing and feature engineering. These transformers allow users to transform raw input data into a format suitable for training machine learning models. Additionally, MLlib supports pipelines, which enable users to easily chain multiple data transformation and modeling stages together, simplifying the development and deployment of complex machine learning workflows.

Another advantage of Spark MLlib is its integration with other components of the Spark ecosystem. It seamlessly works with Spark SQL for data manipulation and querying, Spark Streaming for real-time data processing, and Spark GraphX for graph analytics, providing a unified platform for data processing, analysis, and machine learning.

In summary, Spark MLlib is a comprehensive machine learning library that combines the power of Apache Spark with a wide range of algorithms, utilities, and features for scalable and distributed machine learning. It enables users to process and analyze large-scale datasets, build and deploy machine learning models, and integrate seamlessly with other Spark components for end-to-end data analysis pipelines.

D. Spark MLlib Classifiers

Spark MLlib provides a variety of classifiers that can be used for classification tasks. Here are some commonly used classifiers in Spark MLlib:

- Logistic Regression Classifier: Logistic regression is a popular classifier for binary classification problems. Spark MLlib's logistic regression classifier supports both binary and multinomial logistic regression.
- Decision Tree Classifier: Decision trees are versatile classifiers that can handle both categorical and numerical features. Spark MLlib's decision tree classifier supports binary and multiclass classification.
- Random Forest Classifier: Random forests are ensemble methods that combine multiple decision trees to improve classification accuracy. Spark MLlib's random forest classifier is capable of handling both binary and multiclass classification.
- Gradient-Boosted Tree Classifier: Gradient-boosted trees are another ensemble method that iteratively builds decision trees, with each subsequent tree focusing on the samples that were misclassified by previous trees. Spark MLlib's gradient-boosted tree classifier supports binary and multiclass classification.
- Naive Bayes Classifier: Naive Bayes is a probabilistic classifier that assumes independence among features.

Spark MLlib provides both multinomial and Bernoulli variants of the Naive Bayes classifier.

- Support Vector Machine (SVM) Classifier: SVM is a popular classifier that separates classes by finding the optimal hyperplane in a high-dimensional space. Spark MLlib's SVM classifier supports binary classification.

These classifiers in Spark MLlib provide powerful tools for performing classification tasks on large-scale datasets. They are designed to work efficiently in a distributed computing environment, making it possible to train and deploy models on clusters of machines.

IV. PRELIMINARIES

A. Problem definition

Sentiment analysis is a natural language processing (NLP) task that involves determining and categorizing the sentiment expressed in a given piece of text, such as a sentence, review, or social media post. The objective of sentiment analysis is to automatically identify the underlying sentiment or opinion expressed by the author, which can be positive, negative, or neutral. The problem of sentiment analysis has several challenges. In this paper, we have the challenges specific to sentiment analysis using the Amazon Books Review dataset include:

- Large and diverse dataset: The Amazon Books Review dataset is extensive and contains a wide range of reviews from various genres, authors, and reader perspectives. Handling this large dataset efficiently and effectively requires scalable and robust algorithms and techniques.
- Noise and ambiguity: Reviews may contain noise, such as typographical errors, slang, or colloquial language, making sentiment interpretation more challenging. Mixed sentiments in the text require the model to handle uncertain and conflict signals.
- Domain-specific sentiment: Sentiment expressions and word usage can be specific to the book domain. The model needs to be trained and fine-tuned on this dataset to capture the unique language patterns and sentiment associations related to books accurately.
- Addressing class imbalance: The dataset have imbalance distribution of positive, negative and neutral reviews. The imbalance data may effect to model performance and bias predictions. Apply some technique is crucial for improve the accuracy of sentiment classification.
- Model generalization: A model trained on the Amazon books reviews dataset should be able to generalize well to new, unseen book reviews beyond the dataset. In the other words, the model need to capture the patterns and features that are not specific to the dataset.

B. Dataset: Amazon Book Reviews

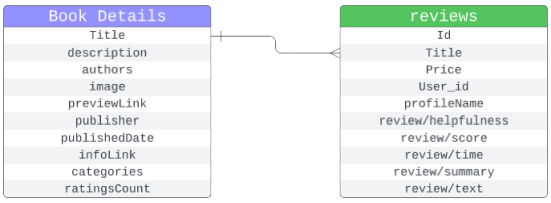

Fig. 3: Dataset's Schema

The dataset consists of two files, and their schema is illustrated in the figure above. The first file, "Books_rating.csv", comprises feedback from 3 million users on 212,404 distinct books. This dataset is a component of the Amazon reviews dataset, which incorporates product reviews and metadata from Amazon, encompassing 142.8 million reviews, spanning from May 1996 to July 2014. The file contains the following attributes:

TABLE I: Books rating descriptions

Features	Description
id	The Id of Book
Title	Book Title
Price	The price of Book
User_id	Id of the user who rates the book
profileName	Name of the user who rates the book
review/helpfulness	helpfulness rating of the review, e.g. 2/3
review/score	rating from 0 to 5 for the book
review/time	time of given the review
review/summary	the summary of a text review
review/text	the full text of a review

The second file Books Details file contains detailed information about 212404 unique books. The file is built by using google books API to get detailed information about books which were rated in the first file. This file contains the following features.

TABLE II: Books data descriptions

Features	Description
Title	book Title
Descript	decription of book
authors	name of book authors
image	url for book cover
previewLink	link to access this book on google Books
publisher	name of the publisher
publishedDate	the date of publish
infoLink	link to get more information about the book on google books
categories	genres of books
ratingsCount	averaging rating for book

Tasks recommended for this dataset may include:

- Recommendation system
- Sentiment analysis
- Text classification
- Text clustering
- Gans generate book cover
- Data analysis
- Data visualization

1) Data distribution: Counting sentences within "review/text" feature.

The majority of customer reviews is between 1 and 20 sentences in length, around 96% record of the dataset.

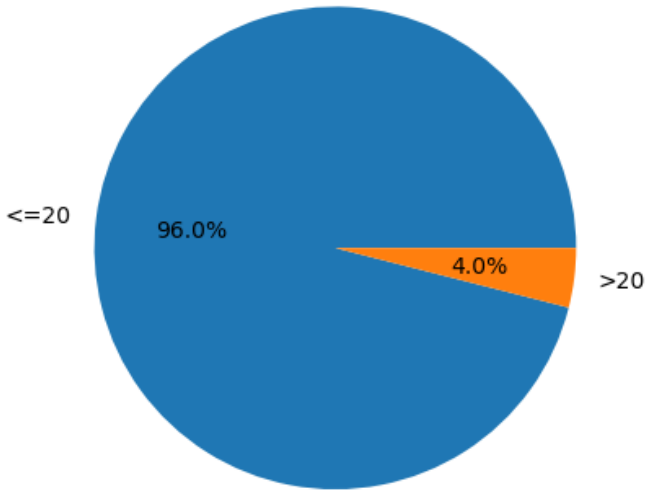

Fig. 4: Sentences in length

Most customer rating scores are positive (4-5/5), which takes 79% of the dataset.

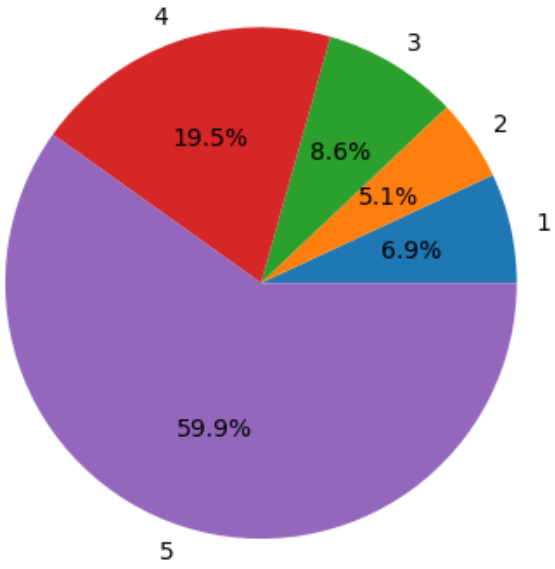

Fig. 5: Label distribution

The labels of the dataset are not uniformly distributed, we need to deal with unbalanced data.

V. METHODOLOGY

In this section, we present methods to train the sentiment analysis models.

A. Overview

Before using classification models, review or sentiment needs to be vectorized into numeric arrays for computation. After tokenizing and removing stopwords, words will be indexed by integers before being vectorized using statistical methods or vectorized computation. The methods of representing word vectors can be mentioned as: TF-IDF, Word2Vec, embedding by models using deep learning architecture (RNN, BertEmbeddings),... The n-gram capture techniques can be used for optimizing feature extraction process on the entire dataset. The n-word sets that appear with high frequency will be retained and become important features in the training process. In this experiment, we use two methods for vectorization: TF-IDF and Word2Vec. For n-grams, we use 1-gram and 2-gram counter to extract the set of single words or word pairs with the most frequency.

At the end of the vectorization phase, the feature vectors continue to be used to train the classifiers. We prefer to use classifiers that consume less resources. Models used include:

Naive Bayes Classifier (NB), Logistic Regression (LR), Support Vector Machine (SVM). These models are suitable for criteria such as consuming less computational resources and training time, and easy to fine-tune hyper-parameters. We increase the accuracy of the models by optimally focusing on the preprocessing and feature extraction phases.

B. Preprocessing

1) Tokenization: First, sentiment or review will be passed through a word separator that is responsible for isolating words, turning sentences into a list containing only words and no spaces. Besides, the words will also be standardized with the following criteria:

- uppercase letters are converted to lowercase.
- special characters are removed, including punctuation.

2) Stop word removal: Stop words are generally meaning a set of commonly used words in any language, not just English, which have very little meaning and are not registered by search engines or other applications that process natural language. It can include personal pronouns, articles, auxiliary verbs, etc. However, they all have one feature that appears a lot in a sentence and does not bring useful information. Therefore, this can be considered a form of noise and should be removed. The removal of these words will contribute to a relative reduction in the number of variables that need to be calculated in the next phases. Currently, the development of natural language processing is leading to the complete completion of stop word corpus. This technique makes a great contribution to saving computational resources during training.

TABLE III: An example of tokenization and stop word removal.

original sentence	"This book is very amazing ^.^!"
tokenized	"this", "book", "is", "very", "amazing"
stop word removal	"book", "amazing"

C. TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency, which is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.

TF-IDF works by multiplying two components:

- Term Frequency (TF), which measures how frequently a word occurs in a document. The more times a word appears, the higher its TF value. However, TF may be adjusted by different methods, such as dividing by the total number of words in the document, or using logarithmic scaling.
- Inverse Document Frequency (IDF), which measures how rare or common a word is across the entire collection or corpus of documents. The more documents that contain a word, the lower its IDF value. IDF is usually calculated by taking the logarithm of the ratio of the total number of documents to the number of documents that contain the word.

The documents are also represented as vectors but instead of a vector of '0's and '1's, now the document contains scores for each of the words. These score are calculated by multiplying TF and IDF for specific words. So, the score of any word in any document can be represented as per the following equation:

$$TFIDF(words, doc) = TF(word, doc) \times IDF(word)$$

Therefore in this method, two matrices have to be calculated, one containing the inverse document frequency of a word in the whole corpus of documents and another containing the term frequency of each word in each document. The formula to calculate both of them are as follows:

$$TF(word, doc) = \frac{Frequency of word \in the doc}{No. of words \in the doc}$$

$$IDF(word) = log_e(1 + \frac{No. of docs}{No. of docs with word}$$

Table IV shows a sample of output after feature extraction by TF-IDF.

TABLE IV: TF-IDF model

Doc/ Wods	the	book	of	pair	was	a	wont	mind
D1	0.09	0.09	0	0	017	0.17	0	0
D2	0	0.09	0.09	0	0	0	0	0
D3	0	0	0	0.09	0	0	0	0
D4	0	0	0	0.09	0	0	0.17	0.17

D. Word2vec

The Word2Vec implementation in PySpark uses the skip-gram model and the hierarchical softmax method to train the word vectors. The skip-gram model tries to predict the context words given a target word, while the hierarchical softmax method speeds up the computation of the softmax function.

The skip-gram model is usually more suitable for large corpora and rare words, as it can capture more semantic

and syntactic information than the continuous bag-of-words (CBOW) model, which tries to predict the target word given the context words.

The skip-gram objective function sums the log probabilities of the surrounding n words to the left and right of the target word w_t to produce the following objective:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leq j\leq n,\neq 0} log_p(w_{j+1}|w_t)$$

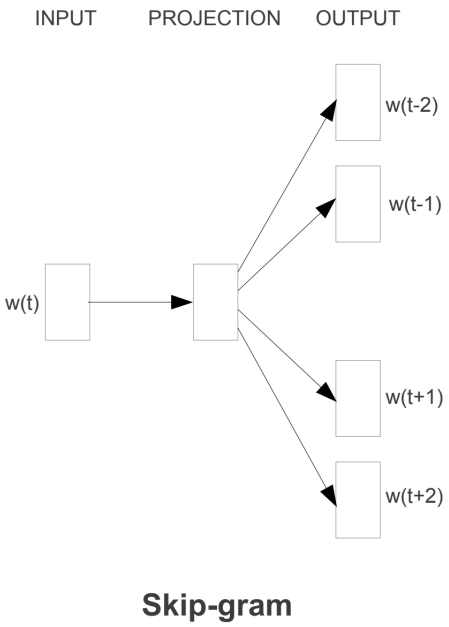

Fig. 6: Skip-gram model

Figure 7 shows an example represents words to vector space.

E. N-gram

An n-gram model models sequences, notably natural languages, using the statistical properties of n-grams. For example, in the sentence "I love dogs", there are three unigrams (1-gram): "I", "love", and "dogs"; two bigrams (2-gram): "I love" and "love dogs"; and one trigram (3-gram): "I love dogs".

The dense presence of sets of words in the corpus shows its influence on the content of the reviews. Two reviews are considered to have the same opinion if there are many similar sets of words between them. This is considered an important feature in the classifier training process.

Once we separated the sets of words, we utilize statistics to count and keep the sets with the highest frequency according to a fixed threshold. Afterwards, we use IDF to extract the features that represent the review with the corresponding

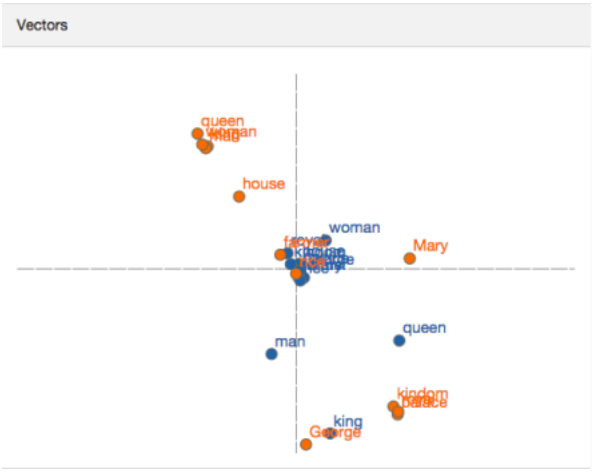

Fig. 7: Vector representation of words in vector space

frequency, then concat the feature vectors of each n-gram model as input to the classification models.

Figure 8 and 9 show an example of feature extraction with bi-grams.

```
+------------------------------------------------------------
|2_grams
+------------------------------------------------------------
|[come tolkien, tolkien many, many us, us fans, fans reread,
|[love brand, brand new, new nicely, nicely illustrated, ill
|[loved book, book better, better hoped, hoped ,  glad, glad
|[much written, written tolkiens, tolkiens grand, grand saga
|[someone third, third world, world ,  huge, huge investment
+------------------------------------------------------------
only showing top 5 rows
```

Fig. 8: Bi-gram capture

```
+------------------------------------------------------------
|2_tfidf
+------------------------------------------------------------
|(4096,[8,529],[3.542909083159237,5.891599774952046])
|(4096,[58,499,641,1322,1684,3432],[4.525389582097757,5.
|(4096,[49,73,585,2348],[4.4446282254532985,4.6191167893
|(4096,[83,995,3763],[10.00203293779571,6.230308648604977
|(4096,[2,10,60,170,180,404,495,530,640,811,903,906,1023
+------------------------------------------------------------
only showing top 5 rows
```

Fig. 9: Bi-gram with IDF

VI. EXPERIMENTS

In this section, we conduct experiments and analyze the acceptance.

A. Implementation details

1) Setting up: We perform the experiment in the system environment as presented in the table V.

TABLE V: Main settings

	Version
Python	3.6.8
JDK	11.0.19
Hadoop	3.3.1
Spark	2.0
pyspark and spark-nlp	3.0.0

2) Relabeling the dataset: In this task, we focus on predicting the likelihood of the reviews being satisfied or not. Therefore, converting the problem into binary classification is more beneficial in the later training and application process. Specifically, with a user's score between 1 and 5, we label comments with scores of 1, 2, 3 as negative and 4, 5 as positive.

3) Hyper-parameters tuning: We perform hyper-parameter tuning in both feature extraction and training. For the classifier models, the hyper-parameters are properly tuned to avoid overfitting. The hyper-parameters are shown in the table VI.

TABLE VI: Hyper-parameters tuning

HashingTF	numFeatures=1000
Word2Vec	vectorSize = 100 minCount = 0 maxIter = 5
NGram (single gram)	ngram = 1 vocabSize = 8096 minDocFreq = 50
NGram (bi-gram)	ngram = 2 vocabSize = 4096 minDocFreq = 5
Naive Bayes	smoothing = 1.0 modelType = 'multinomial'
Logistic Regression	regParam = 0.01 elasticNetParam = 0.01 maxIter = 1000
Linear SVM Classifier	regParam = 0.01 maxIter = 1000

4) Using pre-trained models: We use two pre-trained models on 2 datasets of 1.6 million tweets [12] and 50,000 movie reviews [11]. The above models are public on spark-nlp models hub.

Table VII show the accuracy of pre-trained model on the original dataset.

B. Result and Discussion

In table VII, the best model for F1-score results is 0.88. It is easy to see that the feature extractor using NGram is making better use of the data. Word pairs appearing with high frequency are retained as the most important feature for classification models. Besides, the use of vectorization method using Word2vec has slightly improved the results compared to the statistical method TF-IDF because it can represent rare words.

TABLE VII: Pre-trained model accuracy on the original dataset

Model	**Accuracy**
sentimentdl_use_twitter	0.80
sentimentdl_use_imdb	0.85

TABLE VIII: Evaluation result

Feature Extractor\ Classifier	Naive Bayes		Logistic Regression		Linear SVC	
Metric	f1	acc	f1	acc	f1	acc
HashingTF-IDF	0.79	0.78	0.77	0.82	0.78	0.80
Word2Vec			0.85	0.85	0.84	0.84
NGram-CountVectorizer-IDF	0.83	0.82	0.87	0.88	0.88	0.88

For pre-trained models, we perform an evaluation on the current dataset and give positive results. In table IX, the pre-trained models have not really reached the exact threshold as our models. This can be explained by the following two points:

- vocabulary difference
- imbalance in negative and positive labels

VII. CONCLUSIONS AND FUTURE WORK

For online retail companies, sentiment analysis is crucial in comprehending the customers' feedback. Analyzing reviews is vital in developing recommendation algorithms that cater to the customers' needs. Amazon, which is one of the largest e-commerce platforms, generates massive amounts of feedback data. In our study, we analyze customer book reviews from Amazon.com, and present a comparative analysis of the accuracy of different classifiers. In this paper, we have utilized serveral preprocessing methods and different classifiers, including NB, LR and SVM. According to the results, our proposed models have outperformed the pre-trained models on the other datasets, which may be due to vocabulary difference and unbalanced data.

TABLE IX: Pretrained model evaluation result on Amazon book reviews dataset

Model	**F1**	**Acc**
sentimentdl_use_twitter	0.75	0.73
sentimentdl_use_imdb	0.82	0.83

To expand the scope of this study, we could integrate diverse feature selection methods such as mutual information (MI), chi-squared test, and information gain to achieve superior representation. We can experiment with hybrid classifiers, such as SVM combined with other techniques, to enhance accuracy further. Additionally, we could build a better recommendation algorithm by taking into account the emotions evoked by customers' reviews.

ACKNOWLEDGMENT

Authors would like to thanks Mr. Nguyen Ngoc Hoa and Mr. Vu Thang Long for their constant encouragement and supervision towards the realization of this work.

REFERENCES

[1] Srujan, K. S., Nikhil, S. S., Raghav Rao, H., Karthik, K., Harish, B. S., & Keerthi Kumar, H. M. (2018). Classification of amazon book reviews based on sentiment analysis. In Information Systems Design and Intelligent Applications: Proceedings of Fourth International Conference INDIA 2017 (pp. 401-411). Springer Singapore.

[2] Jagdale, R. S., Shirsat, V. S., & Deshmukh, S. N. (2019). Sentiment analysis on product reviews using machine learning techniques. In Cognitive Informatics and Soft Computing: Proceeding of CISC 2017 (pp. 639-647). Springer Singapore.

[3] Wang, K., Shen, W., Yang, Y., Quan, X., & Wang, R. (2020). Relational graph attention network for aspect-based sentiment analysis. arXiv preprint arXiv:2004.12362.

[4] Chen, X., Sun, C., Wang, J., Li, S., Si, L., Zhang, M., & Zhou, G. (2020, July). Aspect sentiment classification with document-level sentiment preference modeling. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 3667-3677).

[5] Aljuhani, S. A., & Alghamdi, N. S. (2019). A comparison of sentiment analysis methods on Amazon reviews of Mobile Phones. International Journal of Advanced Computer Science and Applications, 10(6).

[6] Elmurngi, E. I., & Gherbi, A. (2018). Unfair reviews detection on amazon reviews using sentiment analysis with supervised learning techniques. J. Comput. Sci., 14(5), 714-726.

[7] Ahmed, H. M., Javed Awan, M., Khan, N. S., Yasin, A., & Faisal Shehzad, H. M. (2021). Sentiment analysis of online food reviews using big data analytics. Hafiz Muhammad Ahmed, Mazhar Javed Awan, Nabeel Sabir Khan, Awais Yasin, Hafiz Muhammad Faisal Shehzad (2021) Sentiment Analysis of Online Food Reviews using Big Data Analytics. Elementary Education Online, 20(2), 827-836.

[8] Ahmed, H. M., Javed Awan, M., Khan, N. S., Yasin, A., & Faisal Shehzad, H. M. (2021). Sentiment analysis of online food reviews using big data analytics. Hafiz Muhammad Ahmed, Mazhar Javed Awan, Nabeel Sabir Khan, Awais Yasin, Hafiz Muhammad Faisal Shehzad (2021) Sentiment Analysis of Online Food Reviews using Big Data Analytics. Elementary Education Online, 20(2), 827-836.

[9] Jha, B. K., Sivasankari, G. G., & Venugopal, K. R. (2021). Sentiment analysis for E-commerce products using natural language processing. Annals of the Romanian Society for Cell Biology, 166-175.

[10] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia (2020). Learning Spark: Lightning-Fast Big Data Analysis, 2nd Edition, O'Reilly.

[11] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning Word Vectors for Sentiment Analysis. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 142–150. Retrieved from http://www.aclweb.org/anthology/P11-1015

[12] Go, A., Bhayani, R. and Huang, L., 2009. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(2009), p.12. Retrieved from https://www.kaggle.com/datasets/kazanova/sentiment140