A PROJECT REPORT ON

Multi-Class Classificatand Sentiment Analysis using Distributed Text Processing

Submitted in the partial fulfillment for the award of degree in

Masters of Science in Data Science

To

Dhirubhai Ambani Institute Of

Information And Communication Technology

Submitted by

Srivatsa Sethu (202218021)
Asish Kumar Sahoo (202218022)
Zeel Gudhka (202218025)
Radhika Singhal (202218062)

Under the guidance of

Prof. Sandip Modha

Problem Statement

Our challenge was to perform Multi Class Text Classification and sentiment analysis through Distributed Text Processing.

Definition

1. **Distributed Text Processing**

   Distributed text processing refers to the processing of large amounts of text data using distributed computing technologies such as Apache Spark, Hadoop, and others. Distributed text processing can be used for a variety of tasks such as text classification, sentiment analysis, topic modeling, entity recognition, and more. The basic idea behind distributed text processing is to distribute the text data across multiple nodes in a cluster and perform processing tasks in parallel on each node to speed up the processing time.

2. **Sentiment Analysis**

   Sentiment analysis is the process of determining the overall sentiment or emotion expressed in a piece of text, such as a review, social media post, or customer feedback. It involves analyzing the words and phrases used in the text to classify whether the sentiment is positive, negative, or neutral. By analyzing and categorizing the sentiment of large amounts of text data, sentiment analysis provides valuable insights for businesses to understand customer opinions, make informed decisions, and improve products or services.

3. **Multi-class Classification**

   Multi-class classification is a machine learning task where the goal is to classify data instances into one of three or more classes or categories. In contrast to binary classification, which involves distinguishing between two classes, multi-class classification involves distinguishing between multiple classes. The task of multi-class classification can be approached using various algorithms, such as logistic regression, decision trees, support vector machines, random forests, or neural networks. These algorithms learn patterns and relationships from labeled training data to make predictions on unseen data. Multi-class classification has various applications, such as text categorization, image recognition, speech recognition, medical diagnosis, sentiment analysis, and customer segmentation.

Use Case

Distributed text processing in PySpark refers to the ability of PySpark to process large volumes of text data in a distributed and parallel manner across multiple nodes of a cluster. PySpark is a

popular framework for distributed data processing, which allows users to leverage the power of distributed computing to process large volumes of data.

The scope of distributed text processing in PySpark is quite broad and includes tasks such as data cleaning, natural language processing, sentiment analysis, text classification, and more. With PySpark, users can leverage powerful text processing libraries such as NLTK, spaCy, and gensim to perform these tasks in a distributed and scalable way.

PySpark also provides a wide range of text preprocessing functions that allow users to transform raw text data into a format that can be easily analyzed. These functions include tokenization, stop-word removal, stemming, and lemmatization. Another important aspect of distributed text processing in PySpark is its ability to integrate with other big data technologies such as Apache Hadoop, Apache Kafka. This allows users to seamlessly process large volumes of text data from various sources and store the processed data in a distributed and fault-tolerant manner.

Overall, the scope of distributed text processing in PySpark is quite extensive and can be used in a wide range of applications such as natural language processing, recommendations, information retrieval, social media analysis, and more.

Scope of the Project

The scope of the Distributed Text Processing project is to process and analyze large volumes of unstructured text data using distributed computing frameworks such as Apache Spark. The project involves implementing various natural language processing (NLP) techniques such as text cleaning, tokenization, stemming, lemmatization, and sentiment analysis to extract meaningful insights from the text data. The project also involves implementing machine learning algorithms for multi-label classification to build predictive models using the processed text data.

Today there is a problem of Big Data everywhere in the real world. We are generating millions of data every second through various applications such as Amazon, Twitter, Facebook, Spotify to help users gain personalized experience. Is this data useful? Yes, it is. This data can be processed and analyzed in order to draw useful insights based on our requirements. For example, in our problem statement, we have done distributed text processing and sentiment analysis using PySpark. Doing sentiment analysis helps us to know about the satisfaction of the users on buying the product. It is impossible to monitor such huge data of reviews and find out if the product is doing good or not. But when using Spark we can draw sentiment analysis of huge data and discover whether the review of the product is positive, negative or neutral.

The project can be applied to a wide range of domains such as social media analysis, customer feedback analysis, news article analysis, and more.

Technology

Spark

Apache spark is one of the most important software when it comes to handling and processing big data. Apache spark is an open source unified analytical image which is used for real time large scale data processing and allows us to process big data in fast and efficient manner across the clusters of computers. Spark provides a high-level API for programming with data, making it easy to develop and implement applications for batch processing, real-time streaming, machine learning, and graph processing.

The important feature of spark is that it is highly scalable which means that it can handle large volume of data in a distributed manner and is fault tolerant, memory management, and caching for data processing is efficient and reliable Spark provides us a number of advantages which helps in during the big data processing like speed, ease of use, scalability, fault tolerance, compatibility, memory management, caching and machine learning, generality, immutable, advance analytics.

Pyspark

PySpark is a Spark library written in Python to run Python applications using Apache Spark capabilities, using PySpark we can run applications parallelly on the distributed clusterIn other words, PySpark is a Python API for Apache Spark. Apache Spark is an analytical processing engine for large scale powerful distributed data processing and machine learning applications.Some of the important features of data processing is in memory computation, distributed processing, fault tolerant, immutable, lazy evaluation.

Applications running on pySpark perform much faster than other platforms, data can be processed from Hadoop HDFS, AWS S3 using PySpark, provides real time data using streaming and Kafka.Spark supports memory clusters like Hadoop Yarn, Apache Mesos etc.

Databricks

Databricks is a unified analytics platform which is designed to help organizations process and helps us to analyze large volumes of data in a fast, efficient, and collaborative manner. It was founded by the creators of Apache Spark, which is the core of the Databricks platform.Databricks provides a cloud-based environment for data engineers, data scientists to collaborate and work together on data-related projects. It includes a suite of tools and services that allow users to store, manage, process, and analyze data at scale.

The Databricks workspace provides a unified interface and tools for most data tasks including data processing workflows, working in SQL, generating dashboards, performing visualizations, data ingestion, managing security, data discovery, compute management and machine learning modeling.

Flow and Results of the Project

Dataset description

Amazon toys and games reviews dataset - a 1GB dataset of json format, which has about 18,28,971 rows and 12 columns. We dropped 9 unwanted columns and converted it into a csv file. Also, removed null values and reduced the dataset to 11,75,821 rows and 3 columns. Since we had 5 unique values - 1, 2, 3, 4, 5, hence it's a Multi-Class Classification problem.

Text Processing and Multi-Class Classification

Implemented following Pipeline:
Pipeline - lowercase and tokenize the data, removing stop words and doing Tf-idf feature transformation or using word embeddings such as Word2Vec, Doc2Vec, etc

overall	reviewText	summary	sentiment	words
5	[stained, glass, ...]	[nice, book]	positive	[stained, glass, ...]
5	[11, loved, know,...]	[great, picture]	positive	[11, loved, know,...]
5	[picture, great, ...]	[picture, great, ...]	positive	[picture, great, ...]
5	[absolutely, love...]	[beautiful]	positive	[absolutely, love...]
5	[love]	[five, star]	positive	[love]
5	[husband, loved, ...]	[five, star]	positive	[husband, loved, ...]
5	[love]	[five, star]	positive	[love]
4	[cool]	[four, star]	positive	[cool]
5	[exactly, describ...]	[nice, picture, g...]	positive	[exactly, describ...]
5	[sometimes, need,...]	[great, 30, minut...]	positive	[sometimes, need,...]
5	[little, book, gr...]	[great]	positive	[little, book, gr...]
2	[indeed, small, b...]	[ok, best]	negative	[indeed, small, b...]
4	[bought, several,...]	[fun]	positive	[bought, several,...]
1	[total, waste, mo...]	[save, money]	negative	[total, waste, mo...]
5	[nephew, age, 5, ...]	[age, 5, loved, b...]	positive	[nephew, age, 5, ...]
5	[cute]	[cute]	positive	[cute]
3	[pretty, much, av...]	[pretty, much, av...]	neutral	[pretty, much, av...]
5	[age, appropiate,...]	[five, star]	positive	[age, appropiate,...]
4	[cute, little, bo...]	[tiny]	positive	[cute, little, bo...]
3	[tiny, book, page...]	[worked, trip, page]	neutral	[tiny, book, page...]

only showing top 20 rows

Here are 10 most frequent words:

```
+------+------+
|  word| count|
+------+------+
|  love|591735|
| great|472400|
|   one|396312|
|   toy|350121|
|   old|344098|
|  like|328257|
|  year|310035|
|  game|281816|
|little|281676|
|   kid|280446|
+------+------+
```

Training and Evaluation

We tried different models for Training but due to multi-class, some of them failed to fit and the best performing model was Logistic Regression.
The best accuracy we got is using the Logistic Regression Model by using Doc2Vec transformation.

Accuracy Table

Transformation Technique	**Model**	**Accuracy**
Tf - idf	Logistic Regression	61.37%
Tf - idf	Naive Bayes	9%
Word2Vec	Logistic Regression	70.34%
Doc2Vec	Logistic Regression	74.41%
Hashing Term Frequency	RNN	71.91%

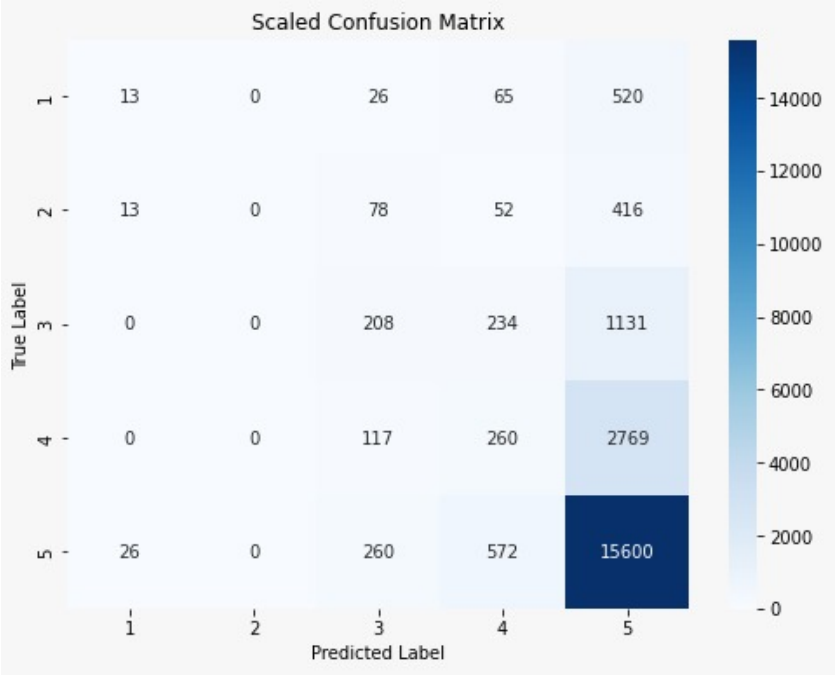
Confusion Matrix

1: Tf-idf with Logistic Regression

```
|overall|prediction|count|
+-------+----------+-----+
|      1|       1.0| 3077|
|      1|       2.0|  210|
|      1|       3.0| 1080|
|      1|       4.0|  997|
|      1|       5.0| 7257|
|      2|       1.0|  810|
|      2|       2.0|  443|
|      2|       3.0| 1651|
|      2|       4.0| 1795|
|      2|       5.0| 6829|
|      3|       1.0|  367|
|      3|       2.0|  190|
|      3|       3.0| 3442|
|      3|       4.0| 5669|
|      3|       5.0|16194|
|      4|       1.0|  170|
|      4|       2.0|  127|
|      4|       3.0| 1528|
|      4|       4.0|12461|
|      4|       5.0|40099|
+-------+----------+-----+
only showing top 20 rows

Logistic Regression Accuracy = 0.613706
```

2. RNN



2: Word2Vec with Logistic Regression

prediction	overall	count
1.0	1.0	3
1.0	2.0	1
1.0	4.0	1
1.0	5.0	3
3.0	1.0	5
3.0	2.0	18
3.0	3.0	14
3.0	4.0	5
3.0	5.0	11
4.0	2.0	1
4.0	3.0	8
4.0	4.0	5
4.0	5.0	5
5.0	1.0	87
5.0	2.0	100
5.0	3.0	209

3: Doc2Vec with Logistic Regression

prediction	overall	count
1.0	1.0	29
1.0	2.0	11
1.0	3.0	7
1.0	4.0	4
1.0	5.0	5
2.0	1.0	6
2.0	2.0	6
2.0	3.0	5
2.0	4.0	1
2.0	5.0	1
3.0	1.0	10
3.0	2.0	15
3.0	3.0	61
3.0	4.0	20
3.0	5.0	10
4.0	1.0	2

Sentiment Analysis

Predicted Sentiments of reviewText with the help of *textblob* inbuilt library.

```
    if sentiment > 0:
        return "Positive"
    elif sentiment < 0:
        return "Negative"
    else:
```

```
        return "Neutral"
```

We had one column in our dataset - “overall’ - which has values 1,2,3,4,5. So, we transformed 1,2 as Negative, 3 as Neutral and 4,5 as Positive.
Finally, we printed the **Confusion Matrix** using the *MulticlassClassificationEvaluator* .

Confusion matrix is :

```
+----+------+
|code| count|
+----+------+
|   1|806548|
|   2| 61886|
|   3| 97058|
|   4| 48236|
|   5| 48650|
|   6| 25084|
|   7| 68921|
|   8| 11100|
|   9|  8338|
+----+------+
```

Where, the code is:

```
((col("Predicted sentiment") == "Positive") & (col("Actual sentiment") == "Positive"), 1)
((col("Predicted sentiment") == "Positive") & (col("Actual sentiment") == "Negative"), 2)
((col("Predicted sentiment") == "Positive") & (col("Actual sentiment") == "Neutral"), 3)
((col("Predicted sentiment") == "Negative") & (col("Actual sentiment") == "Positive"), 4)
((col("Predicted sentiment") == "Negative") & (col("Actual sentiment") == "Negative"), 5)
((col("Predicted sentiment") == "Negative") & (col("Actual sentiment") == "Neutral"), 6)
((col("Predicted sentiment") == "Neutral") & (col("Actual sentiment") == "Positive"), 7)
((col("Predicted sentiment") == "Neutral") & (col("Actual sentiment") == "Negative"), 8)
((col("Predicted sentiment") == "Neutral") & (col("Actual sentiment") == "Neutral"), 9)
```

Conclusion

Word2Vec with Logistic Regression is a good fit for the given problem. RNN also performed nicely. But overall Accuracy could have been better.