CSP 554 – Big Data Technologies
Final Project Report

Sentiment Analysis on the Yelp Dataset using PySpark

Professor: Joseph Rosen

Student's name: Tien Tran

Spring 2019

Introduction

Nowadays, online shopping has become more and more popular. The main advantages of online shopping are that it is much more convenient, and arguably less time consuming. When using online shopping, customers do not have to care about opening hours of the stores or malls, or the weather outside. Besides, they do not spend lots of time trying different stuffs before deciding what to buy. With online shopping, customers can make orders online, wait for them to come, try them and can return them if they are not satisfied. Very convenient and fast indeed! From my own experience, I use online shopping all the time and spend less and less time going to brick and mortar stores. One of the things that help me tremendously in shopping online (other than free shipping, returns and sale-offs throughout the year) is reviews from other customers. Those reviews can be images of the real products, or several sentences showing how good/ bad the products are. They really help online shoppers like me to be more confident when purchasing something.

Nevertheless, customer's review is not only useful in the retail industry. It has become very valuable in other industries such as entertainment (movies/ music reviews) or foods and beverages (restaurant reviews). As more people use reviews (both writing and reading), there are more platforms for that: IMDb, Rotten Tomatoes, TripAdvisor, Yelp, just to name a few.

Bo Pang and Lillian Lee in their paper "Opinion mining and sentiment analysis" (2008) [6] cites some interesting data from two surveys of more than 2000 American adults [7, 8]:

- 81% of Internet users (or 60% of Americans) have done online research on a product at least once;
- 20% (15% of all Americans) do so on a typical day;
- among readers of online reviews of restaurants, hotels, and various services (e.g., travel agencies or doctors), between 73% and 87% report that reviews had a significant influence on their purchase;
- consumers report being willing to pay from 20% to 99% more for a 5-star-rated item than a 4-star-rated item (the variance stems from what type of item or service is considered);
- 32% have provided a rating on a product, service, or person via an online ratings system, and 30% (including 18% of online senior citizens) have posted an online comment or review regarding a product or service.

As a customer, I find those review systems very helpful. But as a data scientist, I am very curious to see how retailers and service providers use the data they get from their customers. Do they try to improve the products/services? Do they hide negative/bad reviews? Do they give incentives to customers with good reviews? And most importantly, do they use that data to make decisions regarding their businesses?

To answer my question, I decide to do some researches on how to extract information from such reviews. The dataset I use is a subset of Yelp's businesses, reviews, and user data. This data set is available on Kaggle: https://www.kaggle.com/yelp-dataset/yelp-dataset/home. It is not a huge dataset, but it has more than 5 million user reviews and hopefully can provide me enough data to come up with some useful conclusions.

The type of data from user reviews is considered as unstructured data, in comparison with structured data like database tables and financial records [1]. Enabling computers to understand and process human languages (Natural Language Processing) is a very broad topic that has been studied extensively over many decades.

Text analytics (text mining) refers to techniques that extract information from textual data [3]. A brief review of text analytics is presented below [3]:

- Information extraction (IE): techniques to extracts structured data from unstructured text, which can be further divided into two sub-tasks: Entity Recognition (ER) and Relation Extraction (RE). In ER, names in text are found and classified in to different categories. Meanwhile, RE extracts relationships between entities.

- Text summarization: techniques to automatically generate a brief summary of single or multiple documents.
- Question answering (QA): techniques that give answers to questions in natural language. These techniques can be further categorized into 3 subsets: the information retrieval-based approach, the knowledge-based approach, and the hybrid approach.
- Sentiment analysis (opinion mining): techniques that analyze text which contains people's opinions toward entities such as products, organization, individuals, and events.

For this project, I'll focus on the Sentiment Analysis of the user reviews only. Another definition of this can be found in Bing Liu's book "Opinion Mining and Sentiment Analysis" (2012). In the book, Liu stated: "Sentiment analysis, also called opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes." [2]. I'll also adopt the Sentiment Classification Using Supervised Learning approach, in which sentiment is classified as positive or negative (binary classification). A review of 4 or 5 stars is considered a positive review, while a review of 1 or 2 stars is a negative one. Reviews with a score of 3 will be ignored to reduce the complexity of the problem.

Sentiment classification is a text classification problem; therefore, various supervised learning methods can be applied, from naïve Bayes to support vector machines. Besides, using a bag of words as features in classification performed quite well with naïve Bayes and SVM [3]. Personally, I'm most familiar with Logistic Regression and Decision Tree-based methods, so I will try to implement those techniques and see if there is any difference in the model accuracy.

Finally, most of the work in this project will be done using the Spark Machine Learning library MLLib. MLLib is Spark's distributed ML library which uses data parallelism technique to store and work with data. The library contains implementations of standard ML algorithms such as classification, regression, clustering, decomposition techniques and others [4]. Since Apache Spark is a commonly used framework for big data processing, many people have chance acquire knowledge and experience using Spark RDD, or Spark SQL dataframes. Such knowledge and experience help a lot to take away the complexity of preparing data to feed into the ML algorithms [5]. Therefore, MLLib is quite an intuitive choice when dealing with ML framework with big data.

In comparison with MLLib, Tensorflow can also do regression, classification, neural networks, etc. and runs on both on CPUs and GPUs. But it is very complex. Tensorflow requires users to understand Numpy arrays very clearly. Some frequently used tasks like converting arrays to one-hot encoded vectors require users to do on their own [5]. Because Tensorflow has the ability to do neural networks and run on GPUs, it is suitable for deep learning projects. However, the type of scale is mostly vertical. In order to achieve better horizontal scaling (which is the goal of big data technologies), Spark MLLib should be applied.

Project Objectives & Description

Objectives

In this project, my main objectives are:

1. Become familiar with the PySpark API, using the Spark SQL (DataFrame) for data munging and analysis, and implementing simple machine learning models with the Spark MLLib.
2. Learn techniques that can be applied to text classification.

The Yelp dataset on Kaggle is suitable for various data science/machine learning research: from exploratory data analysis, data visualization to machine learning modeling techniques. In the Kernel section of the dataset, however, there are not many Kernel focusing on Sentiment analysis and classification. Besides, they are implemented using different frameworks (Machine Learning with R or Deep Learning with Keras). Since I'm

inspired by the Sentiment analysis by Suzanaiacob [9], I'll try to achieve, or if possible, beat her model's accuracy of around 86%.

Setting up PySpark on standalone mode

With those objectives in mind, and with the knowledge of Spark Core and Spark Data Frame that I acquired through lessons and homework, I was very confident that the implementation should be straight-forward. Nevertheless, setting up PySpark on my machine is the hardest task that I had to overcome in this project. I wanted to use Jupyter notebook so that I can debug my codes as quickly as possible. But lots of problems with environment variables, Python version compatibility made me really struggle. In the end, though, I managed to got things working. So shout out to Michael Galarnyk who wrote a wonderful guide about installing PySpark on Windows [10].

Data Acquisition

The Yelp dataset is available for download on Kaggle. Thus, for this part, I just need to download the dataset and then unzip it. In this project, I use only the reviews and the business files, which will be shown in the next parts.

Data Preparation & Analysis

One thing I realized when dealing with this dataset is that Spark Data Frame is more efficient when working with JSON files, in comparison with pandas. Using pandas to read JSON file usually takes 10-25 times memory of the original file []. Therefore, my machine ended up having MemoryError. On the other hand, Spark has no problem loading multiple JSON files.

A quick look at the review schema and first 3 rows:

```
In [6]: review.printSchema()

root
 |-- business_id: string (nullable = true)
 |-- cool: long (nullable = true)
 |-- date: string (nullable = true)
 |-- funny: long (nullable = true)
 |-- review_id: string (nullable = true)
 |-- stars: double (nullable = true)
 |-- text: string (nullable = true)
 |-- useful: long (nullable = true)
 |-- user_id: string (nullable = true)
```

Figure 1. Schema of 'review' Data Frame

```
In [8]: review.show(3)

+--------------------+----+-------------------+-----+--------------------+-----+--------------------+------+------------------
-+
|         business_id|cool|               date|funny|           review_id|stars|                text|useful|             user_i
d|
+--------------------+----+-------------------+-----+--------------------+-----+--------------------+------+------------------
-+
|ujmEBvifdJM6h6RLv...|   0|2013-05-07 04:34:36|    1|Q1sbwvVQXV2734tPg...|  1.0|Total bill for th...|     6|hG7b0MtEbXx5Qzbz
E...|
|NZnhc2sEQy3RmzKTZ...|   0|2017-01-14 21:30:33|    0|GJXCdrto3ASJOqKeV...|  5.0|I *adore* Travis ...|     0|yXQM5uF2jS6es16S
J...|
|WTqjgwHlXbSFevF32...|   0|2016-11-09 20:09:03|    0|2TzJjDVDEuAW6MR5V...|  5.0|I have to say tha...|     3|n6-Gk65cPZL6Uz8q
R...|
+--------------------+----+-------------------+-----+--------------------+-----+--------------------+------+------------------
-+
only showing top 3 rows
```

Figure 2. First 3 rows of 'review' Data Frame

Next, let's look at the top 20 categories of services, extracted from the Yelp's business file:

```
In [11]: business.groupBy("categories").count().orderBy(col("count").desc()).show()
```

```
+--------------------+-----+
|          categories|count|
+--------------------+-----+
|  Restaurants, Pizza| 1042|
|Nail Salons, Beau...| 1031|
|  Pizza, Restaurants|  993|
|Beauty & Spas, Na...|  947|
|  Food, Coffee & Tea|  888|
|Mexican, Restaurants|  885|
|  Coffee & Tea, Food|  865|
|Restaurants, Mexican|  853|
|Chinese, Restaurants|  840|
|Hair Salons, Beau...|  831|
|Beauty & Spas, Ha...|  819|
|Restaurants, Chinese|  789|
|Automotive, Auto ...|  585|
|Auto Repair, Auto...|  534|
|       Food, Grocery|  492|
|       Grocery, Food|  491|
|                null|  482|
|Restaurants, Italian|  474|
|Italian, Restaurants|  446|
|Banks & Credit Un...|  439|
+--------------------+-----+
only showing top 20 rows
```

Figure 3. Top 20 service categories

It seems that many frequently reviewed services in the dataset is related to restaurants and beauty & spa. That's good news for me! I can use this dataset to find out some places to eat in the future. But that's for another project. Now I want to look at the rating distribution across the dataset.

```
In [12]: review.groupBy("stars").count().orderBy(col("stars")).show()
```

```
+-----+-------+
|stars|  count|
+-----+-------+
|  1.0|1002159|
|  2.0| 542394|
|  3.0| 739280|
|  4.0|1468985|
|  5.0|2933082|
+-----+-------+
```

Figure 4. Rating distribution of services

Most of the ratings are 5 stars, followed by 4 and 1 stars. 3 and 2 stars are least likely to appear. From this rating distribution, it can be observed that customers are likely to leave rating when they are satisfied or extremely unhappy with the services. If the services are not so good but not so bad either, they tend to not care so much.

Those are some simple data analysis with the review and business files from the Yelp dataset, just to give a broad overview of how the data looks like. Deep data analysis is not the focus of this project. One thing to notice is that by using a somewhat declarative, SQL-like commands, such data analysis can be intuitive and much easier to understand when compared to data frames in R or pandas.

Data processing for Machine Learning

In this part, data is processed for Machine Learning. For this project, only "stars" (ratings) and "text" (reviews) columns are considered. Ratings will be used as the label, and text will be transformed into appropriate format for machine learning models. Since only those 2 columns are concerned, any entries with missing values (either review without a rating, or rating without a review) can be safely discarded. To reduce the complexity of the problem, reviews with 3-star rating are also eliminated.

```
In [15]: to_select = ['stars', 'text']
         df = review.select(to_select)
         df.show(5)
```

```
+-----+--------------------+
|stars|                text|
+-----+--------------------+
|  1.0|Total bill for th...|
|  5.0|I *adore* Travis ...|
|  5.0|I have to say tha...|
|  5.0|Went in for a lun...|
|  1.0|Today was my seco...|
+-----+--------------------+
only showing top 5 rows
```

```
In [16]: df = df.dropna(how='any')
```

```
In [17]: df = df.filter(df['stars'] != 3)
```

Figure 5. Dropping unnecessary rows/ columns from the data frame

In the next part, we convert 4-star and 5-star ratings into 'Good' Sentiment, and 1-star and 2-star ratings into 'Bad' ones. This was done by using a UDF (User defined function) imported from pyspark.sql.functions.

Pipeline

The following part is heavily inspired by Susan Li and her guide [11]. The pipeline used in this project includes three steps:

1. Tokenization: is the process of taking text (such as a sentence) and breaking it into individual terms (usually words) [12].
2. Remove Stop Words. Stop word are words which should be excluded from the input, typically because the words appear frequently and don't carry as much meaning [12].
3. Count Vectorization: convert a collection of text documents to vectors of token counts [12].

After fitting the pipeline to the data, we have the following data frame:

```
In [41]: from pyspark.ml import Pipeline
         from pyspark.ml.feature import StringIndexer

         pipeline = Pipeline(stages=[tokenizer, swRemover, countVectors])

         # Fit the pipeline to data
         pipelineFit = pipeline.fit(df2)
         data = pipelineFit.transform(df2)
         data.show(5)
```

```
+-----+--------------------+---------+--------------------+--------------------+--------------------+
|stars|                text|Sentiment|               words|            filtered|            features|
+-----+--------------------+---------+--------------------+--------------------+--------------------+
|  1.0|Total bill for th...|      Bad|[total, bill, for...|[total, bill, hor...|(10000,[16,113,14...|
|  5.0|I *adore* Travis ...|     Good|[i, *adore*, trav...|[*adore*, travis,...|(10000,[1,3,4,5,6...|
|  5.0|I have to say tha...|     Good|[i, have, to, say...|[say, office, rea...|(10000,[1,3,10,26...|
|  5.0|Went in for a lun...|     Good|[went, in, for, a...|[went, lunch., st...|(10000,[1,9,22,34...|
|  1.0|Today was my seco...|      Bad|[today, was, my, ...|[today, second, t...|(10000,[3,7,8,12,...|
+-----+--------------------+---------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Figure 6. Data after going through the pipeline transformations.

In the next step, the above data frame is transformed once more to get labeled data. 'Good' sentiment is labeled as 0, while 'Bad' sentiment is labeled as 1.

Machine Learning Models implementation

We can check the distribution of the class in the whole dataset.

```
In [29]: data.count()
Out[29]: 5946620

In [31]: data.filter(data['Sentiment']=='Good').count()
Out[31]: 4402067

In [32]: 4402067/5946620
Out[32]: 0.7402637128318271
```

Figure 7. “Good sentiment” class distribution

This data set is skewed towards good reviews [9]. About 74% of reviews are good ones. Therefore, the baseline accuracy is about 74%.

One important note is that the whole dataset contains more than 5.9 million reviews, which cannot be handled by my machine. Therefore, my approach is to use the first 20,000 rows of the dataset, in which 25% are used as a hold-out test set.

The implementation of machine learning models is straight-forward. Spark MLLib provides detailed documentation [13] and even an example about binary classification [14] to illustrate machine learning pipelines. In this project I implemented several simple models: Logistic Regression, Naïve Bayes, Decision Trees, all of which using the default settings.

For the Logistic Regression and Decision Tree, I also tried hyperparameter tuning using ParamGridBuilder (parameter grid search) with CrossValidator (k-fold cross validation). Results of all models will be discussed in the next part.

Result summary & model evaluation

For model evaluation, I use the Binary Classification Evaluator from the pyspark.ml.evaluation. Results for all models are presented in the following table:

ML model	Accuracy on test set
Logistic Regression	95.55
Naïve Bayes	39.98
Decision Tree	50.07
Logistic Regression with GridSearchCV	97.10
Decision Tree with GridSearchCV	43.77
Baseline accuracy	~ 74%

Comments:

- Logistic Regression and tuned Logistic Regression perform super well on the test set. Even though this technique is very simple and easy to implement, it still brings very good results (even better than my expected results of around 86%).
- All the text in this dataset is sentiment related. Therefore, those ML models can obtain high accuracy. Reviews data is ideal for sentiment analysis [9].
- Naïve Bayes and Decision Tree classifiers perform poorly, much lower than the baseline accuracy. My hypothesis is that those techniques are heavily affected by the curse of dimensionality. With the amount of data that I used (15,000 for training) and the number of features is 10,000; it is very hard for these classifiers to learn from the data. The default settings for decision tree is also not ideal for this kind of high-dimensional data (max-depth = 3).

- To my surprise, the tuned Decision Tree does not perform better. Instead its accuracy decreases sharply. My hypothesis is that when using parameter grid search, bigger trees tend to fit the data better. Therefore, overfitting is more likely to happen.

Possible next steps

If possible, I want to set up a Spark cluster to run modeling on the whole dataset. That could help me to test my hypotheses about model performance. I also want to implement more ML models such as Random Forest, Gradient Boosted Tree, or Support Vector Machine to have a comprehensive comparison between those techniques. Besides, I want to use different approaches to do feature extraction, for example, TF-IDF or Word2Vec. Deeper, more thorough data analysis would also be a good thing to do given the amount of data at hand.

One last thing I can think of is deploying a simple classification model to tell whether a review is positive or negative.

Conclusion

Through this project, I've learned a lot about Spark, a very powerful and popular Big Data framework. I've applied Spark SQL and Spark MLLib to a practical data science problem. Therefore, not only did I become more familiar with Spark SQL and Spark MLLib, I also learned about dealing with textual data and doing sentiment classification.

Most importantly, I learn the value of perseverance. For the last couple of days, I've met so many obstacles in completing this project, and for many times I wanted to stop trying. Luckily, I did not give up and finally achieved some results. I hope those results and lessons will become valuable for my future career.

References

1. https://towardsdatascience.com/an-easy-introduction-to-natural-language-processing-b1e2801291c1
2. Bing Liu, Sentiment Analysis and Opinion Mining, 2012.
3. Amir Gandomi, Murtaza Haider, Beyond the hype: Big data concepts, methods, and analytics, *International Journal of Information Management*, Volume 35, Issue 2, page 137 – 144, 2015.
4. https://www.analyticsindiamag.com/how-mllib-has-become-an-indispensable-part-of-apache-sparks-machine-learning-endeavor/
5. https://www.bmc.com/blogs/machine-learning-ai-frameworks/
6. Bo Pang, Lillian Lee, Opinion mining and sentiment analysis, 2008.
7. comScore/the Kelsey group. Online consumer-generated reviews have significant impact on offline purchase behavior. Press Release, November 2007. http://www.comscore.com/press/release.asp?press=1928.
8. John A. Horrigan. Online shopping. Pew Internet & American Life Project Report, 2008.
9. https://www.kaggle.com/suzanaiacob/sentiment-analysis-of-the-yelp-reviews-data
10. https://medium.com/@GalarnykMichael/install-spark-on-windows-pyspark-4498a5d8d66c
11. https://towardsdatascience.com/multi-class-text-classification-with-pyspark-7d78d022ed35
12. https://spark.apache.org/docs/latest/ml-features.html
13. https://spark.apache.org/docs/latest/ml-classification-regression.html
14. https://docs.databricks.com/spark/latest/mllib/binary-classification-mllib-pipelines.html

Appendix

1. Dataset: https://www.kaggle.com/yelp-dataset/yelp-dataset
2. Code: https://github.com/crtien/Sentiment-Analysis-With-PySpark