HELLENIC REPUBLIC
National and Kapodistrian University of Athens

DEPARTMENT OF INFORMATICS & TELECOMMUNICATIONS

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS - FACULTY OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

M111: Big Data Management
Spring '19

Project Final Report
Title: Restaurants classification using Apache Spark

Kyriakos Christodoulou - cs2180019
Demetris Flouris - cs2180023

List of Contents

Project Description and Purpose

In order to make world a better place for food lovers we decided to make an application that will take as input reviews of restaurants and will decide if each review talks about a good restaurant or a bad restaurant.

In more detail, the approach we will follow is to build a Python program that will use pySpark, Apache's Spark API for Python, to be able to deal with much larger datasets compared to ours and in real time. Using the same algorithms, Apache Spark obviously made things faster and simpler. First of all, it let us load and preprocess the training dataset very fast. Next we use 10-fold Cross Validation on the train set to find the classifier with the best accuracy.

The purpose of our project is to use the best classifier to estimate if the reviews of the test set are about good or bad restaurants. Data will be found locally on the hard disk and results will also be saved locally on a csv file on hard disk as well. Every machine learning used or data feature extractor was included in the pyspark API, so it was extremely easy for us as new users of Apache Spark to transform our knowledge on other subjects into its world.

The challenging part of this project was the need to achieve the best accuracy possible for each algorithm used. This was due to the fact that we handle the textual reviews word by word, ignoring the semantic of the sentences. It was also interesting, the fact that

using Apache Spark's *cache* functionality, to keep a dataframe in the Main Memory instead of reproducing it each time, led to an approximate 85% decrease of the time needed for the program to be executed.

Dataset Description

Train Set

Our train set consists of a tab separated value file which contains two columns and 82,065 records. The first column is the numerical rating the user has given to the restaurant and it ranges between 1-5. We will consider ratings 1,2,3 as bad and ratings 4 and 5 as good. The second column is the textual review given which may describe the experience of the user at the restaurant. We will use this dataset in order to determine which classifier has the best accuracy in classification with 10-fold cross validation.

Example

Score	Text
4	Thank you thank you thank you !! I want to thank the people that made this place...
5	A Humane Society store at the Biltmore? Interesting. I had seen an adorable ....
1	Don't buy Nike sneakers if you want to return or exchange them.. I've bought ...
3	I have to say I love most things about Sprouts and especially this particular store ...
5	The tire pressure light came on a day or so ago, so I had my husband fill the tire ...

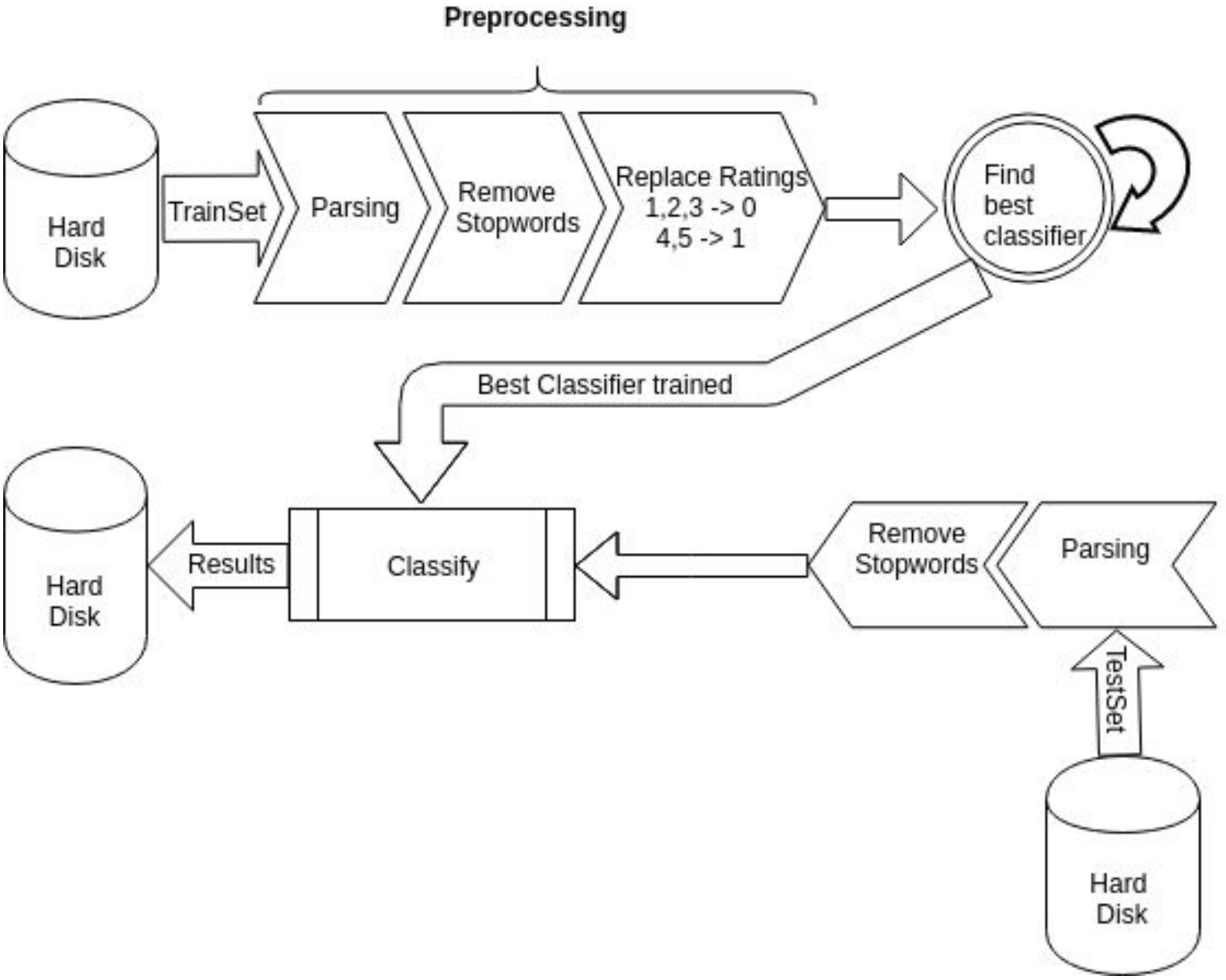
Test Set

Our test set is also a tab separated value which contains only one column, the textual review, and has 34,194 records. We will use this dataset to predict whether the review is bad or good.

Example

Text
My son just loves this place. Weird that he'd ask to come here everytime we go grocery...
We gave it a 9, so we will make that 5-, 4,5 stars. To start with it's just beautiful and ...
After three lunch visits I've come to the conclusion that this restaurant fares just ok in my...
What started out as a simple attempt to find a perfect birthday present turned into a life ...
If they had a Culver's on every street corner, the cardiologists would all have twice as ...

Architecture and Components Description



Components:

1. Preprocessing data

Preprocessing data is consisted of loading data from hard disk, removing stopwords and replacing ratings with 0 and 1. Parsing will first read data from the file passed as argument and get a dataframe containing the review and the rating. Then we will remove stopwords from the reviews so that will not affect classification's results. To do so, we transform the textual review to a list of words. If the file passed in the parse_data function is the training file then we will also change the ratings of 1,2,3 to 0 to represent bad reviews and 4,5 to 1 to represent good reviews. The output will be a dataframe consisting of two columns for the training dataset (review in a list of words and rating) and one dataframe of one column for the testing dataset (review in a list of words) and one other dataframe with the review as a whole string. This will be used at the end so that we can return the results along with their predicted rating.

```
def lower_clean_str(x):
 punc='!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
 lowercased_str = x.lower()
 for ch in punc:
   lowercased_str = lowercased_str.replace(ch, '')
 return lowercased_str

def rate_transform(x):
   if ((x=="1") or (x=="2") or (x=="3")):
       x="0"
   elif ((x=="4") or (x=="5")):
```

```
        x="1"
    return x

def parse_data(path):
    spark = SparkSession.builder.appName("BigDataProject").getOrCreate()
    if (path == "../train.csv"):
        lcs = udf(lower_clean_str)
        rt = udf(rate_transform)
        df = spark.read.csv(path, header=False, sep="\t");
        df = df.withColumn("_c1", lcs("_c1"))
        expres = [split(col("_c1"), " ").alias("_c1")]
        df = df.withColumn("_c1",*expres)
        remover = StopWordsRemover(inputCol="_c1", outputCol="filtered")
        swlist = remover.getStopWords()
        swlist= swlist + list(set(stopwords.words('english')))+ ['']
        remover.setStopWords(swlist)
        final = remover.transform(df.select("_c1"))
        df = df.withColumn('row_index', func.monotonically_increasing_id())
        final = final.withColumn('row_index', func.monotonically_increasing_id())
        final = final.join(df["row_index", "_c0"],on=["row_index"]).drop("row_index")
              .drop("_c1")
        final = final.withColumn("_c0", rt("_c0"))
        return final
    elif (path == "../test.csv"):
        lcs = udf(lower_clean_str)
        df = spark.read.csv(path, header=False, sep="\t");
        initial=df
        df = df.withColumn("_c0", lcs("_c0"))
        expres = [split(col("_c0"), " ").alias("_c0")]
        df = df.select(*expres)
        remover = StopWordsRemover(inputCol="_c0", outputCol="filtered")
        swlist = remover.getStopWords()
        swlist= swlist + list(set(stopwords.words('english')))+ ['']
        remover.setStopWords(swlist)
        remover.transform(df).select("filtered")
        final = remover.transform(df.select("_c0"))
        return final,initial
    else:
        print "Wrong File or Path"
        return -1
```

2. Find best classifier

The data acquired from the previous component will be used in repetition from three different functions. In order to find the best classifier from a set of choices. We decided to try Logistic

Regression, Naive Bayes and Random Forest as classification algorithms and using 10 Fold Cross Validation we train them and calculate their accuracy. We take the dataset from the previous component and transform the rating given to a label. Then we convert the textual review from a list of words to a vector of token counts. This is the first step of the pipeline while the second step is the algorithm itself. These functions return their accuracy, the label model (used to transform the labels back to the initial ratings) and the trained model. These values are passed in a list, sorted based on the accuracy and we return the one with the highest accuracy. We will get in more detail in Section *Classifiers Comparison and Final Choice*.

```
def find_best(data):
    classifiers = []
    classifiers.append((lr_train_cv(data),"Logistic Regression"))
    classifiers.append((rf_train_cv(data),"Random Forest"))
    classifiers.append((nb_train_cv(data),"Naive Bayes"))
    classifiers.sort(key=lambda tup: tup[0][0])
    print str(classifiers[-1][0][2])+ " is the best with accuracy: "
            +str(classifiers[-1][0][0])
    return classifiers[-1][0]
```

3. Classification of test dataset

We use the result of the previous component, which is the best classifier to be used trained with the train set. Parse_data function is used again with the test dataset file so we can parse it and remove any stopwords. The output of this

component is a dataframe of two columns. One with the review as a list without the stopwords and one with the predicted label.

```
predictions=best[-1].transform(df_test)
```

4. Demonstration of results

We have to convert the predicted label to the actual prediction (0,1) and add these predictions to the dataframe containing only the textual review. This dataframe will be saved in a tab separated value file containing these two columns.

```
converter = IndexToString(inputCol="prediction",
       outputCol="originalCategory",labels=best[1])
converted=converter.transform(predictions)
df = df.withColumn('row_index', func.monotonically_increasing_id())
converted = converted.withColumn('row_index', func.monotonically_increasing_id())
df = df.join( converted["row_index", "originalCategory"],
        on=["row_index"]).drop("row_index")
df.repartition(1).write.csv('../predictions',sep="\t")
```

Classifiers comparison and final choice

At the beginning we convert the first column, which is the original rating, into a numeric label. Then we cache the dataframe, so that it will not needed to been regenerated each time is used. To proceed, we had to transform the list of words for each textual review into a vector of token counts, using a Count Vectorizer, with a specific maximum number of tokens in vocabulary and ignore terms that appear less than 5% in our training dataset. This is the first stage of the pipeline followed by the classification algorithm (Logistic Regression, Random Forest and Naive Bayes). Then we build a grid parameter in order to pass parameters for cross validator stages and chosen 10 as a number of folds. In addition we use a Binary Classification Evaluator which gives a binary prediction and save the outcome into a new column, named ‘predictions’. Finally, we fit our cross validator model with the data provided from the previous component and return its accuracy, the model to transform the labels back and the cross validator model itself to use it for the test set prediction.

```
def lr_train_cv(data):
    label_stringIdx = StringIndexer(inputCol="_c0", outputCol="label")
    lsmodel=label_stringIdx.fit(data)
    data=lsmodel.transform(data)
    data.cache()
    countVectors = CountVectorizer(inputCol="filtered", outputCol="cfeatures",
           vocabSize=10000, minDF=5)
    evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction")
    lr = LogisticRegression(regParam=0.3,elasticNetParam=0,
             featuresCol=countVectors.getOutputCol(), labelCol="label")
    pipeline = Pipeline(stages=[countVectors,lr])
    grid = ParamGridBuilder().addGrid(lr.maxIter, [20]).build()
```

```
    crossval = CrossValidator(estimator=pipeline,
                              estimatorParamMaps=grid,
                              evaluator=evaluator,
                              numFolds=10)
    cvmodel=crossval.fit(data)
    return (evaluator.evaluate(cvmodel.transform(data)),lsmodel.labels,cvmodel)

def rf_train_cv(data):
    countVectors = CountVectorizer(inputCol = "filtered", outputCol = "rfFeatures",
              vocabSize = 200, minDF =7)
    label_stringIdx = StringIndexer(inputCol = "_c0", outputCol = "label")
    lsmodel=label_stringIdx.fit(data)
    data=lsmodel.transform(data)
    data.cache()
    evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction")
    rf = RandomForestClassifier(labelCol = "label", featuresCol = "rfFeatures")
    pipeline = Pipeline(stages = [ countVectors, rf])
    grid = ParamGridBuilder().addGrid(rf.numTrees, [10]).build()
    crossval = CrossValidator(estimator=pipeline,
                              estimatorParamMaps=grid,
                              evaluator=evaluator,
                              numFolds=10)
    cvmodel = crossval.fit(data)
    return (evaluator.evaluate(cvmodel.transform(data)), lsmodel.labels, cvmodel)


def nb_train_cv(data):
    label_stringIdx = StringIndexer(inputCol="_c0", outputCol="label")
    lsmodel=label_stringIdx.fit(data)
    data=lsmodel.transform(data)
    data.cache()
    countVectors = CountVectorizer(inputCol="filtered", outputCol="features",
           vocabSize=10000, minDF=5)
    evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction")
    nb = NaiveBayes()
    pipeline = Pipeline(stages=[countVectors,nb])
    grid = ParamGridBuilder().addGrid(nb.smoothing, [1]).build()
    crossval = CrossValidator(estimator=pipeline,
                              estimatorParamMaps=grid,
                              evaluator=evaluator,
                              numFolds=10)
    cvmodel = crossval.fit(data)
    return (evaluator.evaluate(cvmodel.transform(data)), lsmodel.labels, cvmodel)
```

We also tried to use Train Validation Split instead of Cross Validation, which would split randomly the training dataset into two parts to train and test, but it would only do this only once, so the accuracy returned was a bit lower.

Results demonstration

(a)	1	This place is soooooo good!!!!! They have a great selection of hummus. The village hummus (spicy one) is my favorite. Along with the grilled chicken salad. They have a good and sometimes cheap selection of flavored mojitos. I prefer the regular ones but they are all pretty delicious. There aren't too many places on Mill to just go and eat and not get caught with bar food or super greasy food so this is a great change of pace and the food is always good
(b)	1	The Coffee Shop has become a Sunday tradition to the ladies in my family. It reminds us of all the things we love; the beautiful weather of Arizona, the hip indoor rustic d\u00e9cor that reminds us of San Diego or Santa Monica, and the beautiful farm outside. We love sitting out on the beautiful patio surrounded by lush greens and flowers, at times it feels like I'm sitting back at my family's home in Michigan. I typically order the quiche which is fantastic and comes with either fresh cut fruit or a spring salad which is a perfect combination to the fluffy quiche. The coffee is good; I have yet to try a fancy coffee drinks since I typically prefer iced black coffee. If you are ever in the area or want to try something different, I would highly recommend a stop to The Coffee Shop. If you are ever there on a Sunday and see a group of ladies (we all look alike) laughing, chances are that's us...stop by and say hi!
(c)	0	Passable for a korean bbq place. Had the usual ribs, bibimbap, bulgogi was good as usual. However the dumplings were not as great as it should be. The kimchi was ok and so were the sides.This is as good as gets in Arizona I suppose so I cannot be too spoiled coming from Portland with great BBQ places. I still ate and finished as much as I good, so of course it is not a 4 star. It was disappointing that they did not have the seafood pancake...as much as the waitress tried apologizing...she seems to be apologizing all the time. I feel bad as I am sure she wants the business to succeed. How to improve? Easy...make sure the service that everything is right the first time...all sides and rice given in a timely manner and inform guests of what is out on the menu. They should procure better better quality meat as I can see the meat was not as red as it should have being.
(d)	0	While I give the management props for honoring our gift certificate, even though our total rang up below the minimum price, the food was teh ick. I got the fish tacos. I understand the urge to make them unique, and different from the thousands of varieties of fish tacos available in the Valley. But this attempt was a huge miss. The tilapia tasted pretty muddy, even for tilapia. It was served with a really sweet, sticky, almost BBQ-style sauce. I was alternately pleased that it was so cloying that it almost covered the icky river flavor of the tilapia, and perplexed at this odd pairing. The avocados also had a funny, chemical flavor to them. Double-yuck.

(e)	0	I went to my local Petco to purchase a fish. I was in the area waiting to be helped for almost 30 minutes until i finally went outside and got the attention of all three employees who were outside gawking at a puppy a customer came into the store seriously? They were outside when i came in, i walked in knowing what i was going to buy. Then they had to have a conversation to decide who would help me because no one wanted to say goodbye to the puppy. After i finally was handed my fish, i had to again go outside to find out who was ringing me up because the employee i had help me had a line after me, ridiculous!
(f)	0	I hang out in this complex a lot (at Gold Bar) and finally broke down and ate here recently. Here's my thoughts on the whole place.Extreme Pita isn't bad. It's just not great. I mean, theres a decent menu, prices are okay, service is mediocre. But, something about it feels like a subway with pitas. I got the gyro pita, and it was fine, but it just isnt anything that I would crave/miss food wise.Like Micheal C said, it has an overlit, generic fast food decor feeling to it. I think he said it best in his review of this place, and pretty much covered how I feel about Extreme Pita. Go Micheal C!
(g)	1	WOW! The service here was outstanding! Everyone in the entire place seemed to smile, work as a team, and generally do everything and anything possible to make sure me and my family had a great time. And the food was great too! -1 star for a the music/interior. It was almost like rat pack cool type of vibe, but then there was a kinda creepy lady on a piano playing old people music. I didn't mind it, but my family remarked multiple times they thought it was creepy.

From every train session we got about 80% accuracy of rating prediction using reviews and we notice that happens because if the comment is neutral, the selected algorithm’s prediction depends on the words that the reviewer used, rather than the semantic meaning of the sentences. For example if a customer used the words “excellent services” then is possible that the prediction will be possibly 1 (Positive). If a customer reviews “good food but the music is horrible” the prediction will be possibly 0 (Negative).

Conclusions

Our target is to classify textual reviews of restaurants as bad or good. We developed a software that takes as input two datasets, train and test, and preprocess them, by parsing them and removing their stopwords. Then the training data is used to train three different machine learning algorithms and we pick the one with the best accuracy so that we will use it in order to predict the test set.

The approximate accuracy each time we run it is 81% for the Naive Bayes and Logistic Regression algorithms and 60% for the Random Forest. We have the impression that the 85% is due to the following facts:

1. We handle and classify the textual reviews word by word instead of using the semantic meaning of the sentences. This would demand a very different approach, maybe using Neural Networks
2. Although we removed stopwords and symbol characters from the datasets, some words were may be spelled wrong, or in slang. This made the classification even more difficult.

Important things we learned

Using Apache's Spark for the first time was really interesting and it is definitely something we are going to use again in the future. We

handled data very easily and the speed of processing them on only one local computer with 4 GB of RAM was extreme. However, we can easily say that the next time we will use Spark, we will take advantage not only of *cache* but also *persist,* which uses both Main Memory and Hard Disk in order to keep the RDD's needed without reproduce them every time. It can be a very slow experience for someone who:

- re-uses RDD in iterative machine learning applications
- re-uses RDD in standalone Spark applications
- has expensive RDD computations

Future Steps

In order to succeed a better accuracy, we plan to totally change the way we handle the reviews. We will have to take advantage of the sentences' semantic meaning and use these to classify the sentences. Also, a component that would "translate" the reviews from any slang or typing mistakes into english would give an extra advantage to the software.

Link to our Github

https://github.com/kyriakoschr/bigDataProject