Big Data Project2--Yelp Review Text Analysis

My project idea in project 2 is to use the Yelp review data. And use the number of "useful" the comment got to set the label "useful" or "not" to them. And Use word2Vec to represent the reviews content with numeric vectors. The vectors then taken as the feature vectors train the classifier to judge what kind of reviews are useful while what reviews are useless. And I think this will help a lot to user analysis and e-commerce analysis and improve the user experience by gathering the useful information from the reviews.

1. Word2Vec

Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.
And then the represent the words into a K-dimension vector. And Word2Vec uses Distributed representation to present the word vector.

Spark Word2Vec has following parameters to set：

inputCol ,the col name of the input dataframe to be processed.
outputCol, the output col name of the vector got from the model.
vectorSize, the dimension of the output vector. Default is 100.
windowSize, context window size. Default value is 5.
numPartitions, the number of partitions to calculate.
maxIter, the maximum iteration times of this algorithm.
minCount, only when some word appears more than this, will they be included in the word list.
stepSize,the learning rate of the optimization and default value is 0.025.

2. MLP

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.
Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

Spark (MultilayerPerceptronClassifer) has following parameters to set:

featuresCol:the input col name of data frame taken as feature vectors
labelCol:the col name of label in data frame
layers:This parameter is an integer array.The first int should be equal to the size of the feature vector while the final int should be equal to the classification. E.g if it's two classification, then use 2. The integers in the middle represents the learning layers。 E.g int[] layers = Array[Int](100,6,5,2)。
maxIter, the maximum of the iterations time. Default value is 100.
predictionCol:the output col name of data frame

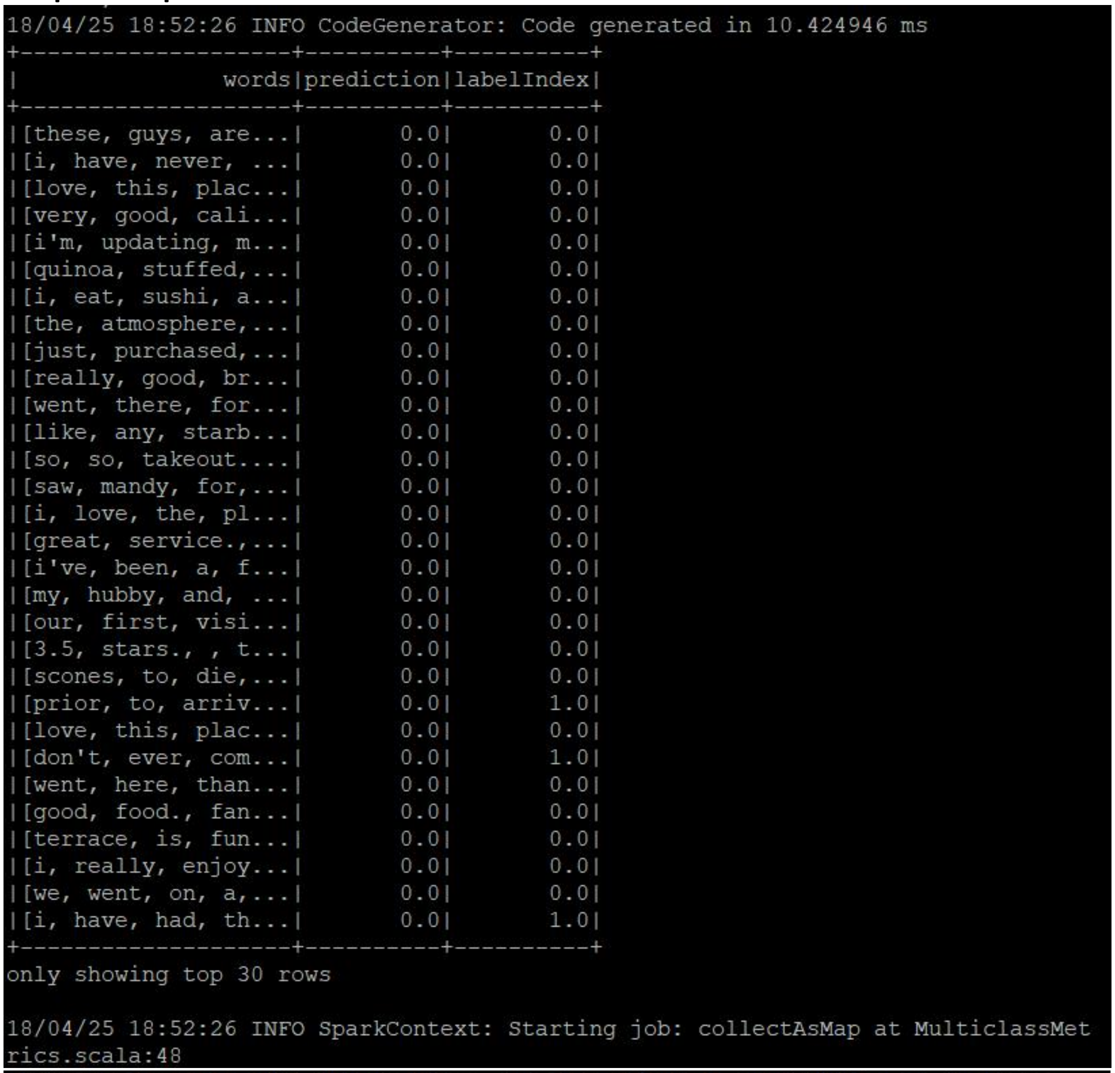
Code:

```
public class Word2VecPredict {
      public static class ParseLineWhole implements FlatMapFunction<Tuple2<String, String>, String[]>
{
            public Iterator<String[]> call(Tuple2<String, String> file) throws Exception {
                  CSVReader reader = new CSVReader(new StringReader(file._2()));
                  Iterator<String[]> data = reader.readAll().iterator();
                  reader.close();
                  return data;
            }
      }
```

```
	public static void main(String[] args){
		SparkConf conf=new SparkConf().setAppName("word2vec").setMaster("local[*]");
		JavaSparkContext sc=new JavaSparkContext(conf);
		//JavaPairRDD<String,String>
csvData=sc.wholeTextFiles("/home/2018/spring/nyu/6513/qy508/HW2/data/tt.csv");
		JavaPairRDD<String,String> csvData=sc.wholeTextFiles("C:/Users/qy508/Desktop/tt.csv");
		JavaRDD<String[]> keyedRDD = csvData.flatMap(new ParseLineWhole());
		JavaRDD<Row> rowRDD=keyedRDD.map(s-> {
			if(Integer.valueOf(s[6].toString())>=3) {
				return RowFactory.create(s[0], s[5],"good");
			}else{
				return RowFactory.create(s[0], s[5],"bad");
			}
		}).repartition(10);
		System.out.print("This first step:"+rowRDD.first());
		StructType schema = new StructType(new StructField[]{
				new StructField("id", DataTypes.StringType,false, Metadata.empty()),
				new StructField("sentence", DataTypes.StringType, false, Metadata.empty()),
				new StructField("label", DataTypes.StringType, false, Metadata.empty())
		});
		SparkSession spark = SparkSession
				.builder()
				.appName("Java Spark SQL basic example")
				.master("local[*]")
				.config("spark.some.config.option", "some-value")
				.getOrCreate();
		Dataset<Row> sentenceData = spark.createDataFrame(rowRDD, schema);
		Tokenizer tokenizer = new Tokenizer().setInputCol("sentence").setOutputCol("words");
		Dataset<Row> wordsData = tokenizer.transform(sentenceData);
		//word2vec
		Word2Vec word2Vec = new Word2Vec()
				.setInputCol("words")
				.setOutputCol("result")
				.setWindowSize(8)
				.setVectorSize(100)
				.setNumPartitions(10)
				.setMinCount(100);
		Word2VecModel model1 = word2Vec.fit(wordsData);
		Dataset<Row> result = model1.transform(wordsData);
		//change label
		StringIndexer stringIndexer=new StringIndexer()
				.setInputCol("label")
				.setOutputCol("labelIndex");
		StringIndexerModel model2=stringIndexer.fit(result);
		result=model2.transform(result);
		System.out.println("This is second step"+result.first());
		//Divide the dataset into 8/2
		Dataset<Row>[] splits = result.randomSplit(new double[]{0.8, 0.2});
		Dataset<Row> train = splits[0];
		Dataset<Row> test = splits[1];
		System.out.println("This is third step:"+train.first());
		//Set up classifier
		int[] layers = new int[] {100, 5, 2};
		MultilayerPerceptronClassifier trainer = new MultilayerPerceptronClassifier()
				.setLayers(layers)
				.setBlockSize(128)
				.setSeed(1234L)
```

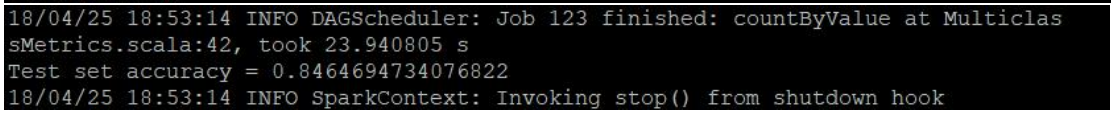
```
                .setMaxIter(100)
                .setFeaturesCol("result")
                .setLabelCol("labelIndex")
                .setPredictionCol("prediction");
        // train the model
        MultilayerPerceptronClassificationModel classify = trainer.fit(train);
        Dataset<Row> classresult = classify.transform(test);
        // compute accuracy on the test set
        Dataset<Row> predictionAndLabels = classresult.select("prediction","labelIndex");
        System.out.println(predictionAndLabels.first());
        MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator()
                .setLabelCol("labelIndex")
                .setPredictionCol("prediction")
                .setMetricName("accuracy");
        classresult.select("words","prediction","labelIndex").show(30);
        System.out.println("Test set accuracy = " + evaluator.evaluate(predictionAndLabels));
    }
}
```

Output Sample:



```
18/04/25 18:52:26 INFO CodeGenerator: Code generated in 10.424946 ms
+--------------------+----------+----------+
|               words|prediction|labelIndex|
+--------------------+----------+----------+
|[these, guys, are...|       0.0|       0.0|
|[i, have, never, ...|       0.0|       0.0|
|[love, this, plac...|       0.0|       0.0|
|[very, good, cali...|       0.0|       0.0|
|[i'm, updating, m...|       0.0|       0.0|
|[quinoa, stuffed,...|       0.0|       0.0|
|[i, eat, sushi, a...|       0.0|       0.0|
|[the, atmosphere,...|       0.0|       0.0|
|[just, purchased,...|       0.0|       0.0|
|[really, good, br...|       0.0|       0.0|
|[went, there, for...|       0.0|       0.0|
|[like, any, starb...|       0.0|       0.0|
|[so, so, takeout....|       0.0|       0.0|
|[saw, mandy, for,...|       0.0|       0.0|
|[i, love, the, pl...|       0.0|       0.0|
|[great, service.,...|       0.0|       0.0|
|[i've, been, a, f...|       0.0|       0.0|
|[my, hubby, and, ...|       0.0|       0.0|
|[our, first, visi...|       0.0|       0.0|
|[3.5, stars., , t...|       0.0|       0.0|
|[scones, to, die,...|       0.0|       0.0|
|[prior, to, arriv...|       0.0|       1.0|
|[love, this, plac...|       0.0|       0.0|
|[don't, ever, com...|       0.0|       1.0|
|[went, here, than...|       0.0|       0.0|
|[good, food., fan...|       0.0|       0.0|
|[terrace, is, fun...|       0.0|       0.0|
|[i, really, enjoy...|       0.0|       0.0|
|[we, went, on, a,...|       0.0|       0.0|
|[i, have, had, th...|       0.0|       1.0|
+--------------------+----------+----------+
only showing top 30 rows

18/04/25 18:52:26 INFO SparkContext: Starting job: collectAsMap at MulticlassMet
rics.scala:48
```



```
18/04/25 18:53:14 INFO DAGScheduler: Job 123 finished: countByValue at Multiclas
sMetrics.scala:42, took 23.940805 s
Test set accuracy = 0.8464694734076822
18/04/25 18:53:14 INFO SparkContext: Invoking stop() from shutdown hook
```

Output analysis:

1. About the prediction accuracy, it's not very high. First, I think it's about the vector size. If I choose to use 100, then the prediction accuracy will be about 85%. But meanwhile it will dramatically increase the running time. So I have to make a good balance between the accuracy and running time. And with my experiment, the good range should be 50-100.
2. And there are also other parameters to change. For example, decrease the layer of the MLP. And it will drop the running time without affecting the classify accuracy much. Because it's a huge training set. Secondly, it's the mincount parameter. We have set a big number to ensure to filter some useless words or tokens got. And this will decrease some useless training words and make it much more powerful.
3. About partitions, at first I think partitions should be as large as possible. However, it's not according to my test. If partition number is too small, it will waste the advantage of the devices. If too large, it will actually increase the communicating cost, which will actually harm the efficiency of computing. So in my work, this should set 10-15 according to my test.

Acknowledge

The author wish to thank New York University, Dr. Raman Kannan, for providing necessary academical assistance to Qi Yao to finish this project successfully. The author also like to thank the continuous support and help from TA, Om Narayan. The author would also like to thank IBM sincerely for sharing IBM Cloud Facility as academical initials! Finally, the author also would like to thank Yelp company for sharing the data to public.