BIG DATA AND INTELLIGENT ANALYTICS
SENTIMENT ANALYSIS – IMDB MOVIE REVIEWS - REPORT

REPORT PREPARED BY

MUBEEN

RASHMI KARUNANITHI

SANDEEP KUMAR RAMADOSS MAHENDRAN

Table of Contents

1. INTRODUCTION

In this project we will study the basics of sentiment analysis with Natural Language Processing techniques using both spark and gensim. We will try to build a predictive model for a real-world dataset and provide results and recommendations. We begin by looking at the basics of

1.1 SENTIMENT ANALYSIS

Sentiment Analysis aims to determine the attitude of the speaker or a writer with respect to some topic using NLP. A basic task in this analysis is classifying the polarity of a given text at the document, sentence or feature/aspect level - whether the expressed opinion in a document, sentence or an entity feature/aspect is positive or negative.
The rise of social media such a blogs and social networks have fueled interest in sentiment analysis. With the proliferation of reviews, ratings, recommendations etc, there is a great opportunity for businesses to identify new opportunities and manage their reputations.

NLP comprises of making the machine understand how humans speak, write and communicate. Humans communicate or convey messages in structured pattern, unstructured pattern, contain regional slangs and idioms. NLP tries to bridge the gap in communication. As our project deals with sentimental analysis of movie reviews we will be using NLP to build different models like TF –IDF and Word2Vec which will predict the polarity of the review.

2. EXECUTIVE SUMMARY

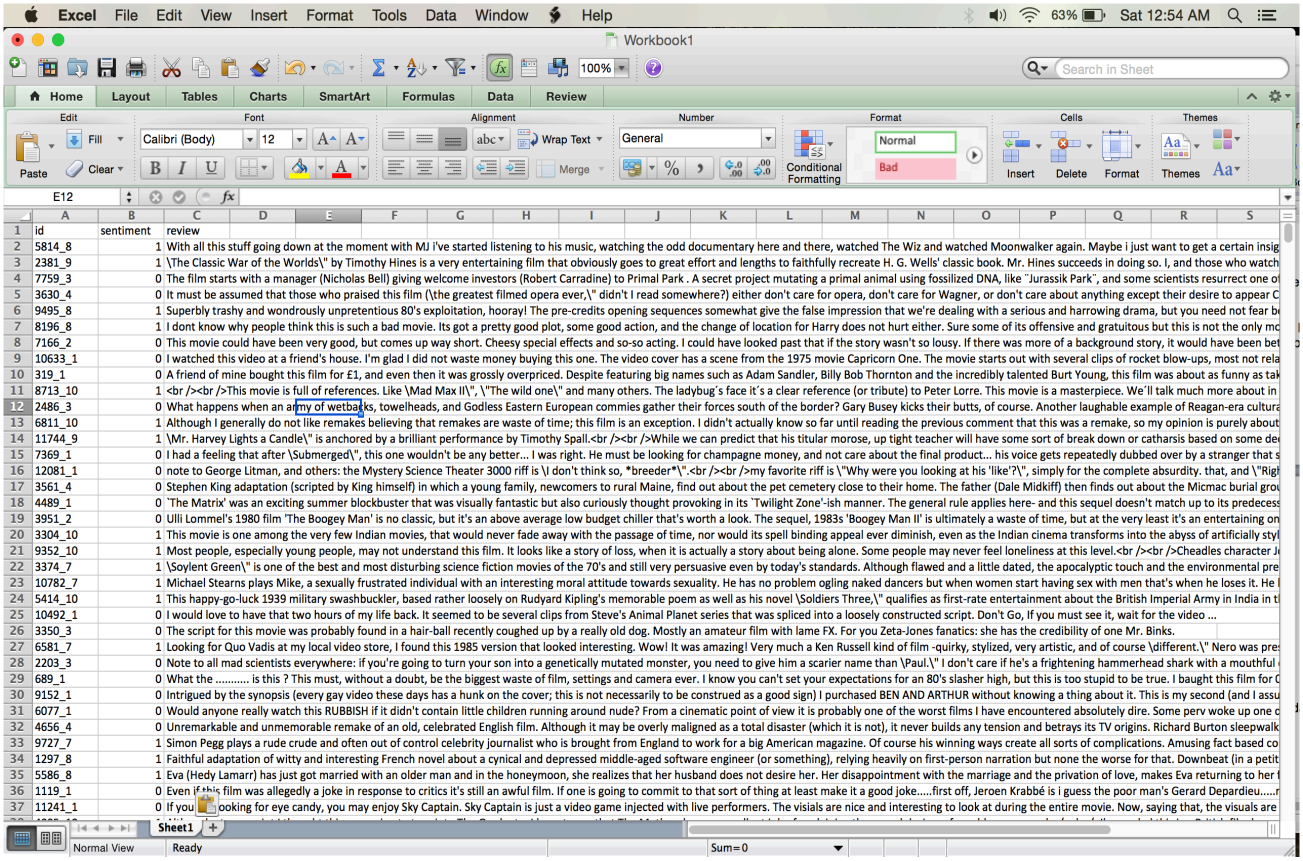
2.1 PROBLEM STATEMENT

The given dataset provides the 25000 reviews of movies on IMDB. The sentiment is associated with each review as 0(thumbs down) or 1(thumbs up).

Our approach aims to apply simple TF-IDF as well as Word2Vec models to try and build a system, which can predict the polarity of the review as thumbs down (0) or thumbs up (1).

Dataset Screenshot:

Labeled TrainData

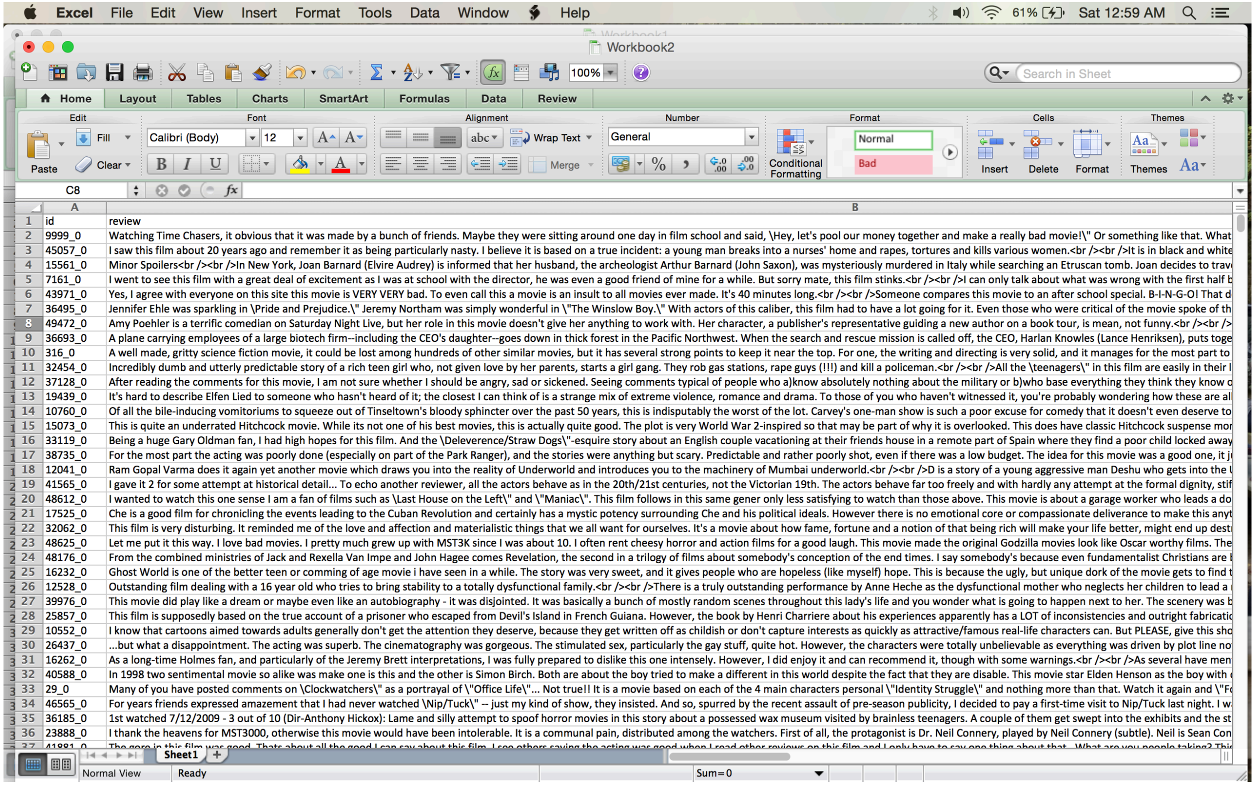


	A	B	C
1	id	sentiment	review
2	5814_8	1	With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again. Maybe i just want to get a certain insig
3	2381_9	1	\The Classic War of the Worlds\" by Timothy Hines is a very entertaining film that obviously goes to great effort and lengths to faithfully recreate H. G. Wells' classic book. Mr. Hines succeeds in doing so. I, and those who watch
4	7759_3	0	The film starts with a manager (Nicholas Bell) giving welcome investors (Robert Carradine) to Primal Park . A secret project mutating a primal animal using fossilized DNA, like ¨Jurassik Park¨, and some scientists resurrect one of
5	3630_4	0	It must be assumed that those who praised this film (\the greatest filmed opera ever,\" didn't I read somewhere?) either don't care for opera, don't care for Wagner, or don't care about anything except their desire to appear C
6	9495_8	1	Superbly trashy and wondrously unpretentious 80's exploitation, hooray! The pre-credits opening sequences somewhat give the false impression that we're dealing with a serious and harrowing drama, but you need not fear be
7	8196_8	1	I dont know why people think this is such a bad movie. Its got a pretty good plot, some good action, and the change of location for Harry does not hurt either. Sure some of its offensive and gratuitous but this is not the only mo
8	7166_2	0	This movie could have been very good, but comes up way short. Cheesy special effects and so-so acting. I could have looked past that if the story wasn't so lousy. If there was more of a background story, it would have been bet
9	10633_1	0	I watched this video at a friend's house. I'm glad I did not waste money buying this one. The video cover has a scene from the 1975 movie Capricorn One. The movie starts out with several clips of rocket blow-ups, most not rela
10	319_1	0	A friend of mine bought this film for £1, and even then it was grossly overpriced. Despite featuring big names such as Adam Sandler, Billy Bob Thornton and the incredibly talented Burt Young, this film was about as funny as tak
11	8713_10	1	 This movie is full of references. Like \Mad Max II\", \"The wild one\" and many others. The ladybug´s face it´s a clear reference (or tribute) to Peter Lorre. This movie is a masterpiece. We´ll talk much more about in
12	2486_3	0	What happens when an army of wetbacks, towelheads, and Godless Eastern European commies gather their forces south of the border? Gary Busey kicks their butts, of course. Another laughable example of Reagan-era cultura
13	6811_10	1	Although I generally do not like remakes believing that remakes are waste of time; this film is an exception. I didn't actually know so far until reading the previous comment that this was a remake, so my opinion is purely about
14	11744_9	1	\Mr. Harvey Lights a Candle\" is anchored by a brilliant performance by Timothy Spall. While we can predict that his titular morose, up tight teacher will have some sort of break down or catharsis based on some dee
15	7369_1	0	I had a feeling that after \Submerged\", this one wouldn't be any better... I was right. He must be looking for champagne money, and not care about the final product... his voice gets repeatedly dubbed over by a stranger that s
16	12081_1	0	note to George Litman, and others: the Mystery Science Theater 3000 riff is \I don't think so, *breeder*\". my favorite riff is \"Why were you looking at his 'like'?\", simply for the complete absurdity. that, and \"Righ
17	3561_4	0	Stephen King adaptation (scripted by King himself) in which a young family, newcomers to rural Maine, find out about the pet cemetery close to their home. The father (Dale Midkiff) then finds out about the Micmac burial grou
18	4489_1	0	`The Matrix' was an exciting summer blockbuster that was visually fantastic but also curiously thought provoking in its `Twilight Zone'-ish manner. The general rule applies here- and this sequel doesn't match up to its predecess
19	3951_2	0	Ulli Lommel's 1980 film 'The Boogey Man' is no classic, but it's an above average low budget chiller that's worth a look. The sequel, 1983s 'Boogey Man II' is ultimately a waste of time, but at the very least it's an entertaining on
20	3304_10	1	This movie is one among the very few Indian movies, that would never fade away with the passage of time, nor would its spell binding appeal ever diminish, even as the Indian cinema transforms into the abyss of artificially styl
21	9352_10	1	Most people, especially young people, may not understand this film. It looks like a story of loss, when it is actually a story about being alone. Some people may never feel loneliness at this level. Cheadles character Jo
22	3374_7	1	\Soylent Green\" is one of the best and most disturbing science fiction movies of the 70's and still very persuasive even by today's standards. Although flawed and a little dated, the apocalyptic touch and the environmental pre
23	10782_7	1	Michael Stearns plays Mike, a sexually frustrated individual with an interesting moral attitude towards sexuality. He has no problem ogling naked dancers but when women start having sex with men that's when he loses it. He l
24	5414_10	1	This happy-go-luck 1939 military swashbuckler, based rather loosely on Rudyard Kipling's memorable poem as well as his novel \Soldiers Three,\" qualifies as first-rate entertainment about the British Imperial Army in India in th
25	10492_1	0	I would love to have that two hours of my life back. It seemed to be several clips from Steve's Animal Planet series that was spliced into a loosely constructed script. Don't Go, If you must see it, wait for the video ...
26	3350_3	0	The script for this movie was probably found in a hair-ball recently coughed up by a really old dog. Mostly an amateur film with lame FX. For you Zeta-Jones fanatics: she has the credibility of one Mr. Binks.
27	6581_7	1	Looking for Quo Vadis at my local video store, I found this 1985 version that looked interesting. Wow! It was amazing! Very much a Ken Russell kind of film -quirky, stylized, very artistic, and of course \different.\" Nero was pres
28	2203_3	0	Note to all mad scientists everywhere: if you're going to turn your son into a genetically mutated monster, you need to give him a scarier name than \Paul.\" I don't care if he's a frightening hammerhead shark with a mouthful o
29	689_1	0	What the .... is this ? This must, without a doubt, be the biggest waste of film, settings and camera ever. I know you can't set your expectations for an 80's slasher high, but this is too stupid to be true. I baught this film for 0
30	9152_1	0	Intrigued by the synopsis (every gay video these days has a hunk on the cover; this is not necessarily to be construed as a good sign) I purchased BEN AND ARTHUR without knowing a thing about it. This is my second (and I assu
31	6077_1	0	Would anyone really watch this RUBBISH if it didn't contain little children running around nude? From a cinematic point of view it is probably one of the worst films I have encountered absolutely dire. Some perv woke up one d
32	4656_4	0	Unremarkable and unmemorable remake of an old, celebrated English film. Although it may be overly maligned as a total disaster (which it is not), it never builds any tension and betrays its TV origins. Richard Burton sleepwalk
33	9727_7	1	Simon Pegg plays a rude crude and often out of control celebrity journalist who is brought from England to work for a big American magazine. Of course his winning ways create all sorts of complications. Amusing fact based co
34	1297_8	1	Faithful adaptation of witty and interesting French novel about a cynical and depressed middle-aged software engineer (or something), relying heavily on first-person narration but none the worse for that. Downbeat (in a petit
35	5586_8	1	Eva (Hedy Lamarr) has just got married with an older man and in the honeymoon, she realizes that her husband does not desire her. Her disappointment with the marriage and the privation of love, makes Eva returning to her f
36	1119_1	0	Even if this film was allegedly a joke in response to critics it's still an awful film. If one is going to commit to that sort of thing at least make it a good joke.....first off, Jeroen Krabbé is i guess the poor man's Gerard Depardieu.....r
37	11241_1	0	If you [illegible]ooking for eye candy, you may enjoy Sky Captain. Sky Captain is just a video game injected with live performers. The visials are nice and interesting to look at during the entire movie. Now, saying that, the visuals are

COLUMN DETAILS:

Column Name	Data Type	Description
id	String	Unique Identifiers
Sentiment	Number	Sentiment of rview
Review	String	Review Statements

UnLabeled TrainData



2.2 APPROACH

2.2.1 Random Split

We used random split to split the data into train data and test data so that the split data could be used for training the model and for testing the model.

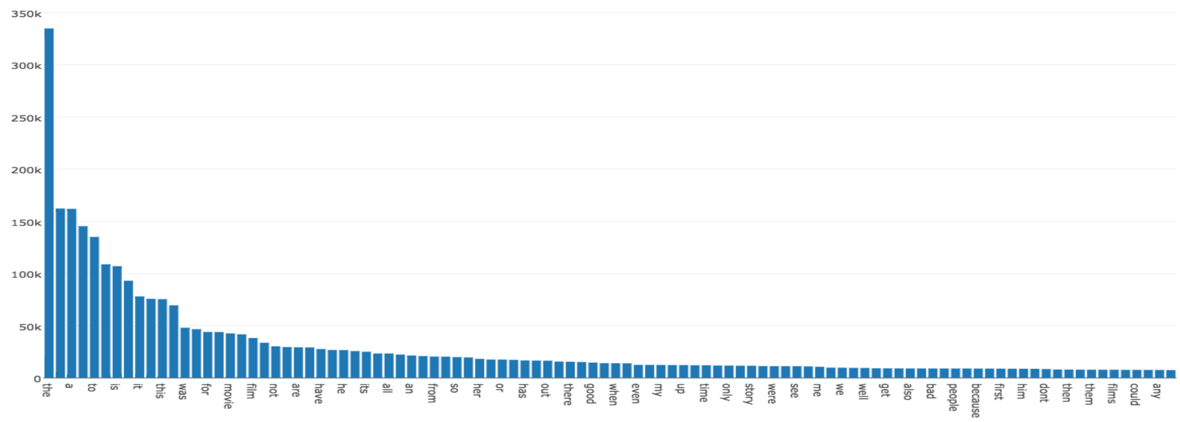
2.2.2 Exploratory Data Analysis

First the dataset is looked for irregularities. In our dataset the reviews will have special characters and html tags that needs to be removed as a part of data preprocessing step. We remove those html tags using The Beautiful Soup library. To remove the punctuation, Numbers we used NLTK.

Following are the plots of few data analysis

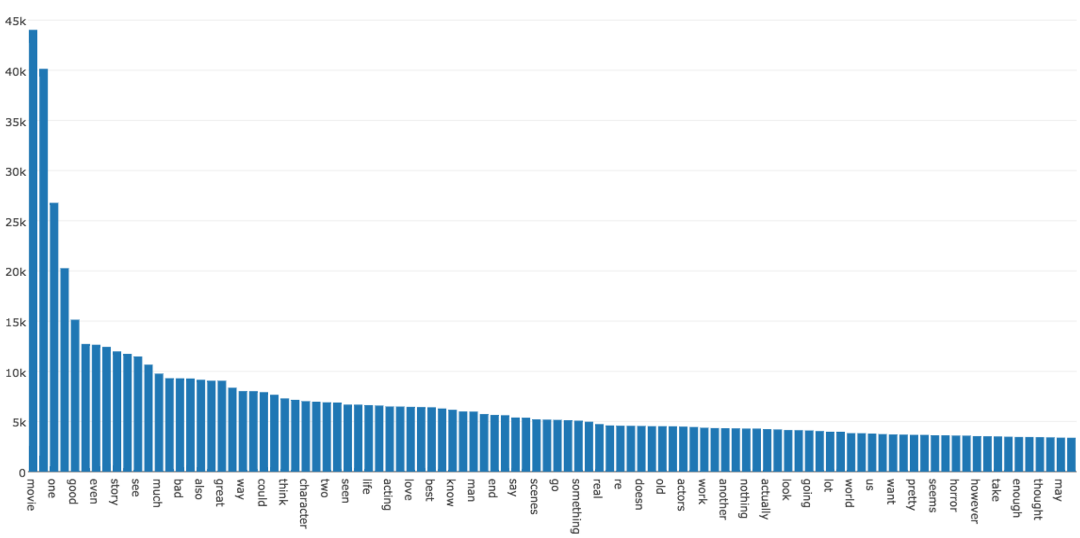
2.2.2.1 Case 1

The following plot tells about the word count in the dataset that is a very naive word count:



2.2.2.2 Case 2

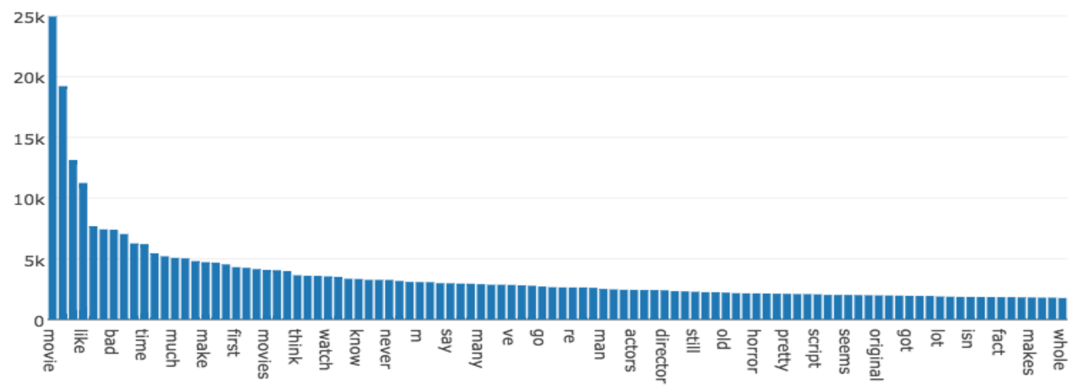
In the previous plot the words like: the, of, is, I etc show up, which are not really useful. So we are using NLTK (Natural Language ToolKit) which will remove words like the, of, is and keep words that don't repeat often. After using NLTK the data analysis of the reviews is as follows:



The plot has the following words - movie, like, even, really, much, people, great, make, think, watch

2.2.2.3 Case 3

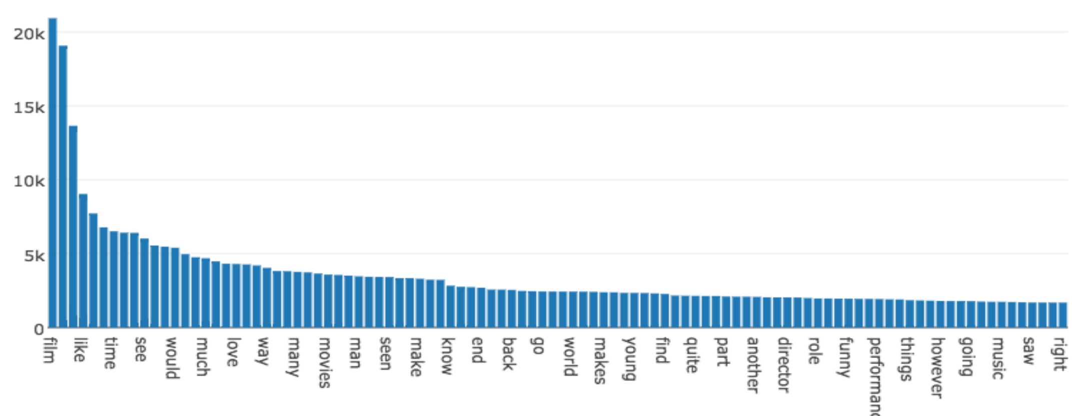
Below is the plot of negative reviews which tells about the negative word count



And the plot has words like bad, never, old, horror etc.

2.2.2.4 Case 4

Below is the plot of positive reviews which tells about the positive word count



And the plot has words like love, funny, right etc.

2.3 DATA STORAGE:

We store the test data and train data in Amazon S3. In future if a new data comes, it will also be stored in S3. Amazon Simple Storage Service (Amazon S3), provides developers and IT teams with secure, durable, highly-scalable object storage. Amazon S3 is easy to use, with a simple web services interface to store and retrieve any amount of data from anywhere on the web. Amazon S3 can be used alone or together with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), and Amazon Glacier, as well as third party storage repositories and gateways.

3. FEATURE ENGINEERING

Feature engineering is the process of creating 'features' that could be used in machine learning algorithms. For the purpose of sentiment analysis, the text that we use to train and predict has to be converted to usable and efficient numerical vectors to be used in the classification algorithms. Unlike other process, for text analysis, the assigning of vectors cannot be done manually and natural language processing (NLP) has to be used. For this specific problem, we used two different techniques for feature engineering, with varied results.

3.1 Term frequency–inverse document frequency (TF-IDF)

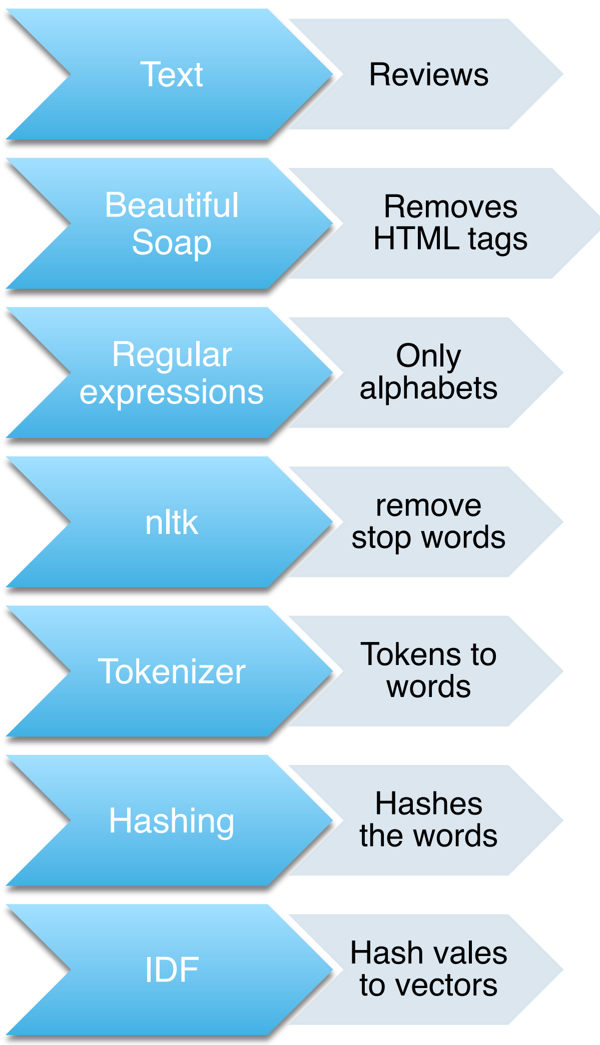
Term frequency–inverse document frequency (TF-IDF), is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. TF-IDF is available as part of pyspark.ml.features package and helps in converting the text to vectors depending on the occurrence and repetition of the word. Before using TF-IDF, The data has to be cleaned. The first step in cleaning the text is the removal of HTML tags from the text. For this purpose, we use a package called Beautiful Soup available in python. The next step would be removed the punctuations and non-alphabetical characters, these characters will affect the efficiency of the created vectors. This is done using regular expressions by replacing anything that is not alphabetical with a space.

Every sentence in a spoken language has words, which repeat frequently. In the English language it would be the preposition, articles, pronouns etc. These words are called stopwords in natural language processing. It is important we remove these words before the features could be created. We use the package called nltk for this purpose. We download the stop words and remove those words from our text by looping through it.

Once we have reached this state, the next step would be to tokenize the text into words. This is done using the tokenizer function available in pyspark.ml.features. Once the tokenization is done, we use a hashing function to convert the words into numerical values for the creation of TF-IDF model. The converted vectors are then fed to the IDF class for fitting. Once the model is fitted, it can be used for transforming our hashed vales to vectors. The IDF model on transformation gives a SparseVector of features. These features can be used in the ml algorithm. The IDF model also gives us options to choose the number of features we would want to use.

We tried the IDF model for 50 features and 300 features. On evaluation of the classification model, the model with 300 features gave an area under ROC value of about 80% whereas the model with 50 features performed under an area under ROC of 67%.

The steps from text to vectors using TF-IDF are given below.

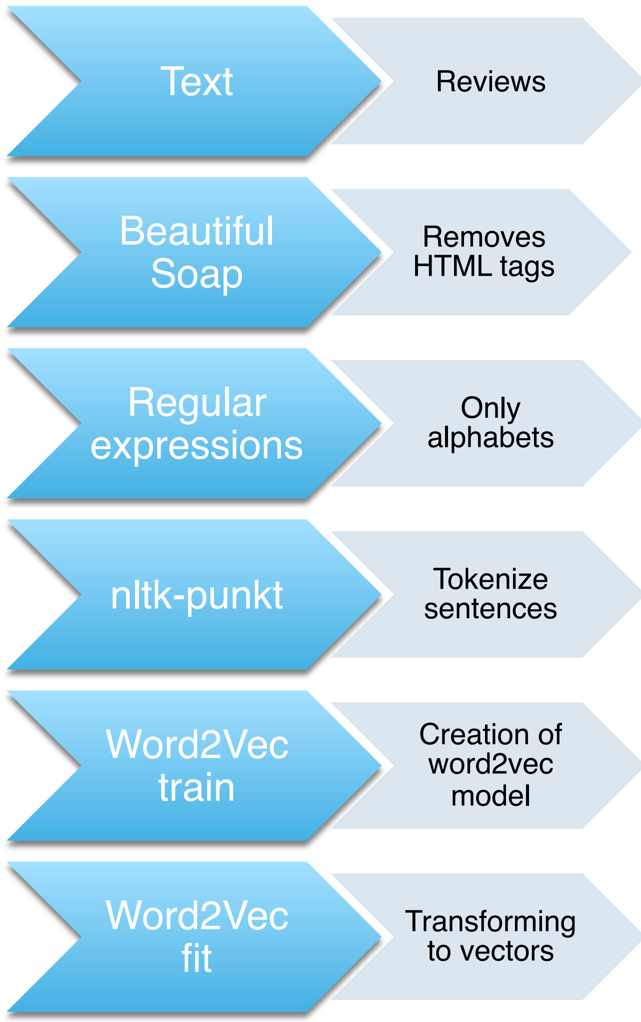


3.2 Word2Vec

The Word2Vec also converts the words to vectors, but taking into consideration the semantics and grammar of the language used. Word2Vec is also available in the pyspark.ml.feature package. The Word2Vec works by creating a model with the language training data to learn the semantics of the language used in processing. This requires a lot of text for training and it is to be noted that this training data need not be labeled for the sentiment it represents. So, we used the unlabeled reviews that were available for this process. The unlabeled data also goes through the process of cleaning. Beautiful Soup and regular expressions are used to remove html tags and non-alphabetic characters in the text.

For the word2vec model to be fitted it is important that this data has to be in the form of sentences instead of words. English language has various ways in which its sentences would be ended. To create sentences from paragraphs, we make use of yet another feature from the nltk package called punkt. This punkt is fed to a tokenizer and the reviews are tokenized into sentences. This is then fed to the Word2Vec model for fitting. Once our model is fitted, we can use it to train our actual training data for the machine leaning algorithm. This data is also cleaned for html tags and non-alphabets, but in both the word2vec training data and the ml training data is not treated for stop words, as the stop words are integral part of language semantics. We were able to attain an area under ROC of about 89% with the vectors created using the word2Vec model (300 features)

The process diagram is as follows



4. MODELS FOR TF - IDF

4.1 RANDOM FOREST CLASSIFICATION

Random forests train a set of decision trees separately, so the training can be done in parallel. The algorithm injects randomness into the training process so that each decision tree is a bit different. Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data.

The randomness injected into the training process includes:

- Subsampling the original dataset on each iteration to get a different training set (a.k.a. bootstrapping).
- Considering different random subsets of features to split on at each tree node.

Apart from these randomizations, decision tree training is done in the same way as for individual decision trees.

CODE

```
stringIndexer = StringIndexer(inputCol="label", outputCol="indexed")
si_model = stringIndexer.fit(selectData)
td = si_model.transform(selectData)
rfc = RandomForestClassifier(maxDepth=2, labelCol="indexed")

pipeline = Pipeline(stages=[tokenizer, hashingTF,idf,stringIndexer , rfc])

paramGrid = ParamGridBuilder().addGrid(hashingTF.numFeatures, [300,
400]).addGrid(rfc.maxDepth, [2, 5, 10]).build()

cv =
CrossValidator().setNumFolds(3).setEstimator(pipeline).setEstimatorParamMaps(param
Grid).setEvaluator(BinaryClassificationEvaluator())
cvModel = cv.fit(pipelineTrainingData)
testTransform = cvModel.transform(pipelineTestData)
predictions = testTransform.select('review', 'label', 'prediction')
predictionsAndLabels = predictions.map(lambda x : (x[1], x[2]))
trainErr = predictionsAndLabels.filter(lambda r : r[0] != r[1]).count() /
float(testData.count())
print("TrainErr: "+str(trainErr))
BinaryClassificationEvaluator().evaluate(testTransform)
```

RESULT

Accuracy = 55.600362

4.2 NAIVE BAYES

Naive Bayes is a simple multiclass classification algorithm with the assumption of independence between every pair of features. Naive Bayes can be trained very efficiently. Within a single pass to the training data, it computes the conditional probability distribution of each feature given label, and then it applies Bayes' theorem to compute the conditional probability distribution of label given an observation and use it for prediction.

These models are typically used for document classification. Within that context, each observation is a document and each feature represents a term whose value is the frequency of the term (in multinomial naive Bayes) or a zero or one indicating whether the term was found in the document (in Bernoulli naive Bayes). Feature values must be nonnegative. The model type is selected with an optional parameter “multinomial” or "bernoulli" with "multinomial" as the default. Additive smoothing can be used by setting the parameter λ (default to 1.0). For document classification, the input feature vectors are usually sparse, and sparse vectors should be supplied as input to take advantage of sparsity. Since the training data is only used once, it is not necessary to cache it.

CODE

```
lp = selectData.map(lambda x : LabeledPoint(x.label,x.features))
(trainingData, testData) = lp.randomSplit([0.6, 0.4])
nb = NaiveBayes.train(trainingData,1.0)
pipeline = Pipeline(stages=[tokenizer, hashingTF,idf, nb])
model = pipeline.fit(trainingData)
selected = model.transform(testData).select('review', 'label', 'prediction')

# Build a parameter grid.
paramGrid = ParamGridBuilder().addGrid(hashingTF.numFeatures, [300,
400]).addGrid(nb.regParam, [0.01, 0.1, 1.0]).build()

#Set up cross-validation.
cv =
CrossValidator().setNumFolds(3).setEstimator(pipeline).setEstimatorParamMaps(param
Grid).setEvaluator(BinaryClassificationEvaluator())

#Fit a model with cross-validation.
cvModel = cv.fit(trainingData)
testTransform = cvModel.transform(testData)
predictions = testTransform.select("review", "prediction", "label")
predictionsAndLabels = predictions.map(lambda x : (x[1], x[2]))
trainErr = predictionsAndLabels.filter(lambda r : r[0] != r[1]).count() /
float(testData.count())
print("TrainErr: "+str(trainErr))
BinaryClassificationEvaluator().evaluate(testTransform)
```

RESULT

accuracy = 72
recall = 71.5
precision: 72.05

4.3 LOGISTIC REGRESSION WITH ML

Logistic Regression is a regression model where the nominal variable is categorical. Use Binary Logistic regression when you have one nominal variable with two values like male/female or good/bad and one measurement variable, which is an independent variable. Use Multiple Logistic regression when you have one nominal variable and more than one measurement variable. The goal of logistic regression is to predict the probability of getting the desired nominal value given the measurement value and also to predict that getting a particular nominal value is dependent on the measurement value.

CODE

```
(trainingData, testData) = selectData.randomSplit([0.6, 0.4])
lr = LogisticRegression(maxIter=5, regParam=0.01)
model = lr.fit(trainingData)
result =  model.transform(testData)
evaluator = BinaryClassificationEvaluator()
evaluator.evaluate(result, {evaluator.metricName: "areaUnderPR"})
evaluator.evaluate(result, {evaluator.metricName: "areaUnderROC"})
```

Here we do a random split of the data that is available to us. We train the model with 60 percent of data and test the model with 40 percent of the data. We set the maximum no of iterations as 5 and the regularization parameter to 0.01. We fit the model according to the training data. Then the test data is transformed based on the model. We predict whether the sentiment of the review is 1 (good) or 0 (Bad). AreaUnderROC values plots false positive rate against true positive rate. So higher the auROC value better the model.

RESULT

areaUnderPR = 0.8065552181670259
Area under ROC = 0.8275891274493837
training error = 0.244788029925

CROSS VALIDATOR

An important task in ML is *model selection*, or using data to find the best model or parameters for a given task. This is also called *tuning*.Pipelines facilitate model selection by making it easy to tune an entire Pipeline at once, rather than tuning each element in the Pipeline separately.

Currently, spark.ml supports model selection using the CrossValidator class, which takes an Estimator, a set of ParamMaps, and an Evaluator. CrossValidator begins by splitting the dataset into a set of *folds* which are used as separate training and test datasets; e.g., with k=3 folds,CrossValidator will generate 3 (training, test) dataset pairs, each of which uses 2/3 of the data for training and 1/3 for testing. CrossValidator iterates through the set of ParamMaps. For each ParamMap, it trains the given Estimator and evaluates it using the given Evaluator. The ParamMap that produces the best evaluation metric (averaged over the k folds) is selected as the best model. CrossValidator finally fits the Estimator using the best ParamMap and the entire dataset.

CROSS VALIDATOR CODE

```
# Build a parameter grid.
paramGrid=ParamGridBuilder().addGrid(hashingTF.numFeatures,[300, 400])
.addGrid(lr.regParam, [0.01, 0.1, 1.0]).build()

#Set up cross-validation.
cv=CrossValidator().setNumFolds(3).setEstimator(pipeline).setEstimatorParamMaps(pa
ramGrid).setEvaluator(BinaryClassificationEvaluator())


#Fit a model with cross-validation.
cvModel = cv.fit(trainingData)
testTransform = cvModel.transform(testData)
predictions = testTransform.select("review", "prediction", "label")
predictionsAndLabels = predictions.map(lambda x : (x[1], x[2]))
trainErr = predictionsAndLabels.filter(lambda r : r[0] != r[1]).count() /
float(testData.count())
print("TrainErr: "+str(trainErr))
BinaryClassificationEvaluator().evaluate(testTransform)
```

For our model we have decided to go with 300 features to run the cross validator class.

RESULT

areaUnderPR = 0.8065552181670259
Area under ROC = 0.8275891274493837
training error = 0.244788029925

5. MODELS FOR WORD2VEC

5.1 LOGISTIC REGRESSION WITH LBFGS

Logistic Regression is a regression model where the nominal variable is categorical. Use Binary Logistic regression when you have one nominal variable with two values like male/female or good/bad and one measurement variable, which is an independent variable. Use Multiple Logistic regression when you have one nominal variable and more than one measurement variable. The goal of logistic regression is to predict the probability of getting the desired nominal value given the measurement value and also to predict that getting a particular nominal value is dependent on the measurement value.

CODE

```
#Creating RDD of LabeledPoints
lpSelectData = selectData.map(lambda x : (x.id, LabeledPoint(x.label,x.features)))
#Spliting the data for training and test
(trainingData, testData) = lpSelectData.randomSplit([0.9, 0.1])
# training the Logistic regression with LBFGS model
lrm = LogisticRegressionWithLBFGS.train(trainingData.map(lambda x: x[1]),
iterations=10)
#fetching the labels and predictions for test data
labelsAndPreds = testData.map(lambda p: (p[0],p[1].label, lrm.predict(p[1].features)))
#calculating the accuracy and printing it.
accuracy = labelsAndPreds.filter(lambda (i, v, p): v == p).count() / float(testData.count())
print("Accuracy = " + str(accuracy))
```

RESULT:

Accuracy = 0.843182696022

5.2 K MEANS CLUSTERING

k-means is one of the most commonly used clustering algorithms that clusters the data points into a predefined number of clusters. The MLlib implementation includes a parallelized variant of the k-means++ method called kmeans. The implementation in MLlib has the following parameters:

- k is the number of desired clusters.
- Max iterations are the maximum number of iterations to run.
- Initialization Mode specifies either random initialization or initialization via k-meansII.
- Runs are the number of times to run the k-means algorithm (k-means is not guaranteed to find a globally optimal solution, and when run multiple times on a given dataset, the algorithm returns the best clustering result).
- Initialization Steps determines the number of steps in the k-meansII algorithm.
- Epsilon determines the distance threshold within which we consider k-means to have converged.

CODE

```
selectRDD = selectData.map(lambda s: s.features)
(trainingData, testData) = selectRDD.randomSplit([0.6, 0.4])
clusters = KMeans.train(trainingData, 2, maxIterations=10,
```

```
    runs=10, initializationMode="random")

def error(point):
  center = clusters.centers[clusters.predict(point)]
  return sqrt(sum([x**2 for x in (point - center)]))

WSSSE = trainingData.map(lambda point: error(point)).reduce(lambda x, y: x + y)
print("Within Set Sum of Squared Error = " + str(WSSSE))
```

RESULT

Within Set Sum of Squared Error = 3347.68

5.3 RANDOM FOREST CLASSIFICATION

Random forests train a set of decision trees separately, so the training can be done in parallel. The algorithm injects randomness into the training process so that each decision tree is a bit different. Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data.

The randomness injected into the training process includes:

- Subsampling the original dataset on each iteration to get a different training set (a.k.a. bootstrapping).
- Considering different random subsets of features to split on at each tree node.

Apart from these randomizations, decision tree training is done in the same way as for individual decision trees.

CODE

```
#Creating RDD of LabeledPoints
lpSelectData = selectData.map(lambda x : (x.id, LabeledPoint(x.label,x.features)))
#Instantiating string indexer for random forest
stringIndexer = StringIndexer(inputCol="label", outputCol="indexed")
#fitting the data in stringindexer
si_model = stringIndexer.fit(selectData)
#transforming the data
transformData = si_model.transform(selectData)
#Spliting the data for training and test
(trainingData, testData) = transformData.randomSplit([0.6, 0.4])
#instantiating Random forest model
randomForest = RandomForestClassifier(numTrees=2, maxDepth=2,
labelCol="indexed", seed=42)
#training the model
randomForestModel = randomForest.fit(trainingData)
```

```
#trsnforming test data
result = randomForestModel.transform(testData)
#calculating the accuracy and printing it.
accuracy = result.filter(result.label == result.prediction).count() / float(testData.count())
print("Accuracy = " + str(accuracy))
```

RESULT

Accuracy = 61.215326

6. COMPARISON

6.1 TF - IDF

We did run the above said three models for TF – IDF and the results are listed below

Algorithm	**Result**
Random Forest Classification	Accuracy = 55.600362
Naive Bayes	accuracy = 72 recall = 71.5 precision: 72.05
TF_IDF LR	areaUnderPR = 79.20658874070523 Area under ROC = 80.61931789957208 training error = 26.133306765
ML TF_IDF LR cross validator	areaUnderPR = 80.65552181670259 Area under ROC = 82.75891274493837 training error = 24.4788029925

Logistic Regression with ML was better of all three because of the ML implementation with cross validator.

FEATURE SELECTION:

Based on the results we decided to run the Logistic Regression With LBFGS of mllib because the results were better in Logistic Regression with ML implementation.

CODE:

```
#Creating RDD of LabeledPoints
lpSelectData = selectData.map(lambda x : (x.id, LabeledPoint(x.label,x.features)))
#Spliting the data for training and test
(trainingData, testData) = lpSelectData.randomSplit([0.9, 0.1])
# training the Logistic regression with LBFGS model
```

```
lrm = LogisticRegressionWithLBFGS.train(trainingData.map(lambda x: x[1]), iterations=10)
#fetching the labels and predictions for test data
labelsAndPreds = testData.map(lambda p: (p[0],p[1].label, lrm.predict(p[1].features)))
#calculating the accuracy and printing it.
accuracy = labelsAndPreds.filter(lambda (i, v, p): v == p).count() / float(testData.count())
print("Accuracy = " + str(accuracy))
```

Result

Accuracy = 74.8927

6.2 WORD2VEC

We did run the above said three models for word2vec and the results are listed below

Algorithm	Result
Random Forest Classification	Accuracy = 61.215326
K means Clustering	Within Set Sum of Squared Error = 3347.68
Logistic Regression with LBFGS	Accuracy = 84.3182696022

Logistic Regression with LBFGS gave better result than the other two models.

7. FUTURE REVIEW PREDICTION

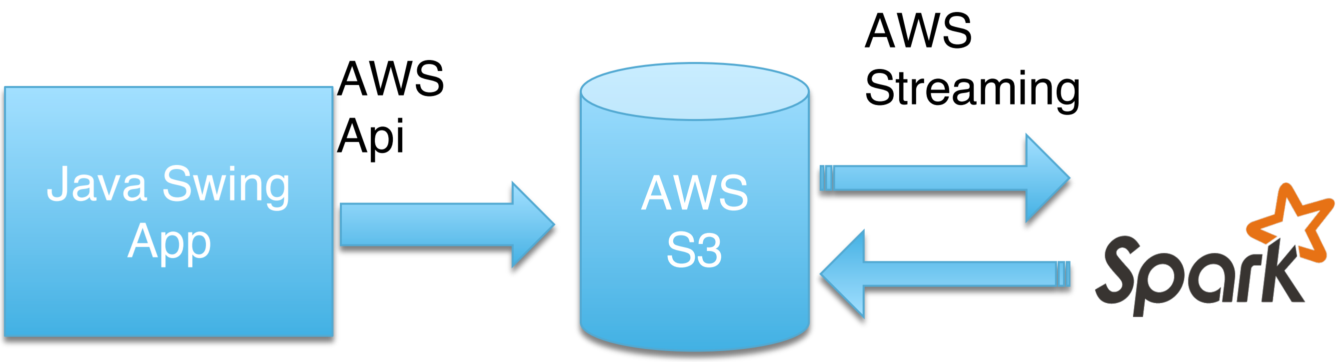
If we want to predict the sentiment of a new review we have a User Interface where the user has to enter his/her review and click on Submit button.

WORKFLOW

The UI is developed is using JAVA Swing. From the Java application the review is read and stored in Amazon S3 using Amazon S3 API. From the storage using AWS streaming the review is given for prediction to the model.

AWS STREAMING

Amazon CloudFront, the easy-to-use content delivery service, now supports the ability to stream audio and video files. Traditionally, world-class streaming has been out of reach of for many customers – running streaming servers was technically complex, and customers had to negotiate long- term contracts with minimum commitments in order to have access to the global streaming infrastructure needed to give high performance.



After prediction the result is written back to S3 using AWS Streaming.

8. CONCLUSION

Logistic Regression with ML gave better result in TF – IDF. So we decided to implement Logistic Regression with LBFGS in ml. Word2Vec is a deep learning model that is better than TF – IDF. Since we got better result for Logistic regression with LBFGS in TF – IDF we decided to implement Logistic Regression with LBFGS in Word2Vec, which gave better result as expected than every other model.

9. REFERENCES

http://spark.apache.org/docs/latest/

https://databricks.com/blog/2015/07/29/new-features-in-machine-learning-pipelines-in-spark-1-4.html

http://alexminnaar.com/word2vec-tutorial-part-i-the-skip-gram-model.html

https://code.google.com/p/word2vec/