ILLINOIS DATA SCIENCE INITIATIVE

TECHNICAL REPORT

# Matching and Classifying Question Similarity Using Rudimentary Techniques

*Author:*
Osmar Coronel
Joshua Dunigan
Professor Robert J. Brunner

June 2, 2017

# Matching and Classifying Question Similarity using Rudimentary Techniques

**Joshua Dunigan**[1], **Osmar Coronel**[1], and **Professor Robert J. Brunner**[1,2,3]

[1] *University of Illinois at Urbana-Champaign*
[2] *National Center For Supercomputing Applications (NCSA)*
[3] *Laboratory for Computation, Data, and Machine Learning*

*Compiled June 2, 2017*

**The focus of this paper is to determine the similarity of questions using techniques such as TFIDF, Word2vec, as well as multiple clustering and classification algorithms.**

https://github.com/lcdm-uiuc

## 1. INTRODUCTION

This technical report demonstrates basic sentence similarity matching techniques using Apache Spark. The datasets that were used were the StackOverflow question dataset along with Quora's Question Pairs Dataset.

This paper also details how to setup closest question pairs on Apache Spark. The similarity of two questions is defined as two questions having the same exact meaning and both questions only needing one answer. The included scripts for parsing and analyzing the data sets used within this report are included in the directory under the name Question_Pairs. They can be found on Github using the link addlocationlater.

## 2. ASSUMPTIONS

It is assumed that users have all the necessary components installed to run a Pyspark cluster in either version 1.5.2 or 2.0.0. The Word2vec model requires that numpy, scipy, scikit-learn, and gensim libraries are installed on the cluster.

## 3. ABOUT STACKOVERFLOW AND QUORA DATASET

StackOverflow and Quora are two websites whose purpose are to answer questions. StackOverflow focuses on technical styled questions, while Quora focuses on more commonplace problems. The StackOverflow dataset was composed of 48GB of data in XML format. The Quora dataset was much smaller (300MB), but was formatted to show whether questions were similar. The Quora dataset is a .csv file, with example data shown in figure 2.

```
<row Id="11" PostTypeId="1" AcceptedAnswerId
    ↪ ="1248" CreationDate="2008-07-31T23
    ↪ :55:37.967" Score="1106" ViewCount
    ↪ ="115048" Body="&lt;p&gt;Given a
    ↪ specific &lt;code&gt;..." OwnerUserId
    ↪ ="1" LastEditorUserId="1136709"
    ↪ LastEditorDisplayName="user2370523"
    ↪ LastEditDate="2015-12-29T02:08:37.450"
    ↪ LastActivityDate="2016-07-13T23
    ↪ :23:58.537" Title="How can relative
    ↪ time be calculated in C#?" Tags="&lt;c
    ↪ #&gt;&lt;..." AnswerCount="33"/>
```

**Fig. 1.** Sample StackOverflow Data

| id | qid1 | qid2 | question1 | question2 | is_duplicate |
|---|---|---|---|---|---|
| 0 | 1 | 2 | What is the ... | What is the ... | 0 |
| 1 | 3 | 4 | What is the ... | What would ... | 0 |
| 2 | 5 | 6 | How can I ... | How can Inter ... | 0 |

**Fig. 2.** Sample Quora Data

## 4. ABOUT APACHE SPARK

Apache Spark is one of the largest open source data tools currently being used. Spark is based off of Hadoop, improving upon some of the downsides of Hadoop and adding new features to fit the new world of data science, such as preferring to write to RAM over disk and parallelizing data into many resilient distributed data sets (RDD) instead of a number of programs. These distributed data sets are immutable, meaning that they cannot be changed since they are shared among the computers. Many large companies and universities contribute to and use Spark for their data processing needs.

## 5. METHODS OF VECTORIZING TEXT

### A. Word2vec

Word2vec, developed by Mikolov et al at Google, is a 2-layer neural network that takes an input of individual sentences and constructs a multi-dimensional space, anywhere from 1 to hundreds of dimensions, where each vector represents a unique word. Similar or closely related words occupy similar areas of space. For example, the Quora dataset when run on the Word2vec model related the words "Trump", "Clinton", "Obama", "Sanders" and "President" closely together in the n-dimensional space. The implementation of Word2vec used is from Gensim.

### B. TF-IDF

Term frequency-inverse document frequency (TF-IDF) is a numeric value given to a word based on how uncommon a word is in a document. Term frequency is calculated using the following equation:

$$tf(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}} \tag{1}$$

where $t$ is the term, $d$ is the document, and $f_{t,d}$ is the raw frequency of the term in the document.

The following equation calculates inverse document frequency. Inverse document frequency takes in consideration of how many times the word appears across all documents.

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2}$$

Where $N$ is the number of documents in the corpus, $t$ is a term, $D$ is a set of all the documents and $d$ is a document in $D$. Multiplying tf and idf gives you the tf-idf score for a given word. Tf-idf is an easy to implement and popular method for text classification.

## 6. MEASURES OF VECTOR SIMILARITY

### A. Cosine Distance

The cosine distance is a measurement of how similar two vectors are. The cosine distance of two vectors that are similar will have a score closer to 1, and vectors that are different will have a score closer to 0. The cosine distance is calculated as one minus the cosine similarity, as show below:

$$\text{distance = 1 - similarity} \tag{3}$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}} \tag{4}$$

### B. Squared Euclidean Distance

The Squared Euclidean Distance is derived from the Pythagorean Theorem. The following equation explains how the distance between two vectors, p and q, is calculated.

$$\begin{gathered} d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \iff \\ d^2(p,q) = \sum_{i=1}^{n} (p_i - q_i)^2 \end{gathered} \tag{5}$$

Where $p_i$ and $q_i$ are the i$^{\text{th}}$ dimension of the vectors $p$ and $q$

### C. Jaccard Index

The Jaccard index is defined as area of intersection of two sets over the total area of their union. The Jaccard index is one way to find the amount of similarity between two sets. The equation below shows how it is calculated for normal sets (A and B).

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{6}$$

## 7. MACHINE LEARNING ALGORITHMS FOR BINARY CLASSIFICATION

### A. Random Forest

The Random Forest algorithm is a collection of decision trees. The random in Random Forest comes from the method of randomly choosing a subspace before fitting each tree and also because the decision of each node in a tree is determined by a random procedure. These methods help reduce overfitting when the algorithm is being over trained. In a classification problem, the algorithm outputs the mode of the classes for predicting an output value, where regression outputs the mean of the classes.

### B. Logistic Regression

Logistic regression is a regression model which separates the data into two distinct sections based on various features of the data by drawing a line/plane between the data-points. Based on which side of the line (or plane, depending on the dimension of the data) the data-point lies, the linear regression model assigns either a 0 or 1 to the data-point. In this paper's case, a 1 is interpreted as both questions being the same and a 0 means that the questions are different. Logistic regression can be thought of as trying to find the best choice of $\beta_0$ and $\beta_1$ for the following equation.

$$y = \begin{cases} 1 & \beta_0 + \beta_1 x + \varepsilon > 0 \\ 0 & otherwise \end{cases} \tag{7}$$

The logistic regression function is given by

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{8}$$

where F(x) outputs the predicted class for the variable.

### C. Naive Bayes

Naive Bayes is a collection of binary classification algorithms all based on the underlying principle that each feature value is independent in deciding the binary class of some data. A multinomial Bayes classifier is used in this paper. This model generates a histogram based on the features of the vectors, where the probability of a particular histogram is calculated using the equation:

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i} \tag{9}$$

where $x$ is the vector, $C_k$ is the conditional outcome being tested, and $p_i$ is the probability that $i$ occurs.

### D. Feedforward Neural Network

A feedforward neural network is a multi-layer perceptron network that only moves in one direction, meaning the network contains no cycles. A multi-layer perceptron is a neural network in which each neuron has direct connections to a neuron in the following layer. Data flows from the first layer which is the input layer, through the intermediate layers, and finally through the output layer. When passed through the intermediate and output layers, functions are applied to the input data. The nodes in the intermediate layers of the network use a logistic function on some data

$$f(x) = \frac{1}{1+e^{-x}} \tag{10}$$

and the nodes in the output layer use a softmax function, which is a generalization of the logistic function.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \quad 1 \le j \le K \tag{11}$$

Where $z$ is a K dimensional vector.

Pyspark's implementation of the model uses backward propagation of errors in the learning model. Backpropagation is used by neural networks to minimize the error value for the output. Pyspark uses a logistic loss function and L-BFGS as the optimization algorithm.

## 8. CLUSTERING ALGORITHMS

### A. K-Means

K-Means is a form of clustering that takes a set of vectors and attempts to cluster each vector by minimizing the overall variance between the center of the clusters and the points in the cluster. In other words, the algorithm assigns each vector the closest center to it, and then through iterative refinement, finds the local optimum centers for each cluster, therefore minimizing the variance of each cluster. The equation below shows the ideal value the K-Means model looks for when matching vectors.

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i \tag{12}$$

Where $k$ is the number of clusters, $S_i$ is the set of all $x$ in the cluster $i$, and $\mu_i$ is the center of $S_i$.

### B. LSH Forest

LSH Forest is a method of clustering that finds the nearest neighbors using a locality sensitive hash (LSH). Locality sensitive hashes are hash functions designed to increase the number of collisions. With more collisions, similar words can be better matched because they are more likely going to have a similar or the identical hashes. The LSH Forest uses this hash to match nearby neighbors using the neighbors' cosine distance.

## 9. PARSING, CLUSTERING AND MATCHING THE STACKOVERFLOW DATA

### A. Cleaning the Data

The 48GB of StackOverflow data was written in an XML format. Inside each row, the PostTypeId tag differentiated whether the line of code was a question or an answer: 1 represented a question and 2 represented an answer. The purposes of this research was to only match questions, so the answers posts were disregarded. It is noted that more precision could be obtained by adding the exact answers to match the question, but it was decided that since each question can have drastically different answers that the answers should be omitted. The answer for each question is defined as the most general solution the question is seeking. Once the answers were discarded from the data, each XML line was split using its quotation marks. Then a CSV writer was initialized by using tab delimiters to create a neat table to parse using Apache Spark. Tab delimiters were picked because the use of tabs was thought to make it easier to load in as a text file. To create this table, the first row of the table were the tags corresponding to the correct values. Then an algorithm looped through the stripped text, took each odd value and translated each line of XML into its respective CSV version. The resulting CSV file was 24 GB.

```
with open('/mnt/volume/stackOverflowFormatted/
   ↪ data.csv','wb') as csvfile:
   writer = csv.writer(csvfile,delimiter='\t')
   writer.writerow([#Array of Tags])

   #Looping through the XML stream
   for line in sys.stdin:
     if len(line)>50:

        #Split each XML line by quotes
        split_line = line.split("\"")

        #Make sure that its a question
        if split_line[3] =="1":
           csv_row = []

           #Going through the 36 elements in
           #the split
           for i in xrange(1,36,2):
              if i < len(split_line):
                 csv_row.append(split_line[i])
              else:
                 csv_row.append("")
              writer.writerow(csv_row)
```

**Fig. 3.** XML Parser

| Id | Score | Body |
|---|---|---|
| 4 | 441 | &lt;p&gt;I want to use a track-bar... |
| 1 | 198 | &lt;p&gt;I have an absolutely posi... |
| 9 | 1374 | &lt;p&gt;Given a &lt;code&gt;Date... |

**Fig. 4.** Newly Parsed Data

### B. Setting Up the "Cleaner" File

The Spark Databricks package https://spark-packages.org/package/databricks/spark-csv was used to parse the CSV data, which returns a DataFrame. Unfortunately, this DataFrame is not cleaned, so it has to be transformed back to an RDD and cleaned. In the RDD, white noise (such as &lt;, p&gt;, &#xA;, /, &gt;, &quot;, &amp;, nbfsp;, lt;) is removed from the 6th column of the RDD using regular expressions. The cleaning process was restricted to the 6th column because it was the column in which the body of the question was located. With this new RDD, any sentence that had 10 or less characters was filtered as this served as a check to make sure that any analysis done on the questions would work in some way. The final step in setting up a Word2vec model is to split the question into individual words.

```
#databricks CSV Spark parsing
df = sqlContext.read.format("com.databricks.
    ↪ spark.csv").option("header", "true").
    ↪ option("inferSchema", "true").option("
    ↪ delimiter", '\t').load("hdfs:///shared/
    ↪ stackOverflow/data_formatted.csv")

data = df.rdd

#Cleaning the data
dataClean = data.map(lambda column: re.sub(r"
    ↪ (&lt;)|(p&gt;)|(&#xA;)|(/)|(&gt;)|(&
    ↪ quot;)|(&amp;)|(nbsp;)|(lt;)","␣",
    ↪ column[6])).filter(lambda question: len
    ↪ (question)>10)

#Cleaned data for the Word2Vec model
cleanData = dataClean.map(lambda question:
    ↪ question.split())
```

**Fig. 5.** Parsing and cleaning the CSV data.

### C. Using the Word2Vec Model

The Word2vec model helps create meaningful numerical vectors from words. Since the data is made up of sentences, each specific word had to be mapped to a distinct vector and the meaning of the sentence can be represented as the vector sum of its words.

Passing the sentences into Spark's Word2vec Model caused many memory leaks. Thus, the best alternative is the Gensim's Word2vec model. However, it does come with a drawback: since Gensim was made for singular computers, its algorithm could only map words in their own partition. Withal, while this does cause a few slight inefficiencies in the code, it will still yield a useful result.After creating summed vectors from the sentences, the question pairs were ready to be compared using the squared magnitude of each vector and the cosine similarity between the vector and the vector (1,1,1).

### D. Hashing and Clustering each Question Pair

To pair each question, first K-Means was used to cluster each question and then the questions were paired with each of their 3 closets neighbors inside the clusters. However, K-Means was returning suboptimal results due to the large amounts of data, so we shifted our focus onto LSHForrest. LSHForrest fit its model onto our data and then predicted the closest 3 neighbors and distances to those neighbors from the data itself. We paired a

```
[[996.68446862365818, 140.78347389306873,
    ↪ 71.254942409694195],
 [376.47404986619949, −103.73854973458219,
    ↪ −185.17367091216147],
 [450.57150414213538, −61.323632740415633,
    ↪ −236.1701169180451],
 [549.85922444425523, −262.07716701144818,
    ↪ −243.10790812002961]]

[(1018277.1833348331, 0.6923849755803575),
(186783.68532277748, 0.11711142151904544),
(262551.59240247076, 0.17268660500405739),
(430131.06316568109, 0.039374022443619087)]
```

**Fig. 6.** Example vectors

question with the closest other question in the LSHForest model and labeled the pair 1 to indicate similarity, all other question pairs which had a further distance were labeled 0.

```
LSHForest(min_hash_match=4, n_candidates=50,
    ↪ n_estimators=10, n_neighbors=5,
     radius=1.0, radius_cutoff_ratio=0.9,
        ↪ random_state=42)
```

**Fig. 7.** Running the LSHForest algorithm.

| Quid1 | Quid2 | Similarity |
|---|---|---|
| I have an idea... | I work in VBA,... | 1 |
| How do you shade... | I'm looking for... | 0 |
| This is related to... | Consider the following... | 0 |

**Fig. 8.** Newly parsed data.

## 10. TRANSFORMING, READING AND ANALYZING SIMILARITY BETWEEN QUESTIONS

### A. Cleaning the text

First, two questions and the label from each data point were extracted. Then, each sentence was cleaned and all the words were changed to lowercase and any punctuation in the sentence was removed to lower random noise. Finally, to deal with special characters - which would cause problems while vectorizing the text - instead of removing them all together, the characters were replaced with the string representation of their ASCII value. That way, if two sentences shared some character, the accuracy of the models is not degraded.

### B. TF-IDF and Word2Vec Models

Once the text was cleaned, two models were created to vectorize the sentences. For creating the TF-IDF, a dictionary was created where words mapped to their TF-IDF score; the Word2vec model, made by Gensim, was used to generate vectors. This model takes in a list of lists of strings, where the lists of string are all the cleaned words in the sentence.

```
def clean_sentence(sentence):
    cleaned_sentence = ""
    cleaned_sentence = re.sub(r'[^\w\s]','',
        ↪ cleaned_sentence)
    for letter in sentence:
        if ord(letter) > 127:
            cleaned_sentence += ord(letter)
        else:
            cleaned_sentence += letter.lower()
    return cleaned_sentence
```

**Fig. 9.** The python code to clean the sentences.

```
# w2v is the word2vec model name
# weights is the tfidf weights for each word
def weight_vector(sentence):
    sentence_vector = []
    for word in sentence:
        if word is not '':
            w2v_vector = w2v.wv[word]
            tfidf_score = weights.get(word)
            sentence_vector.append(w2v_vector*
                ↪ tfidf_score)
    return np.sum(sentence_vector)
```

**Fig. 10.** Code for calculating the sum of a sentence.

### C. Turning Text Into A Vector

After creating the models, each word is turned into a vector. There are many combinations that can be done with the Word2vec and TF-IDF score, such as: Word2vec times the TF-IDF score, adding them together, and also using one of each without combining them. In order to combine the individual vectors of each word in a sentence into one vector, different methods such as taking the mean, sum, and normalized sum are used.

### D. Feature extraction

After the sentences are vectorized, similarity measurements were applied on the vectors. The two methods used were the Cosine Distance and the Squared Euclidean Distance. Another method that takes a different kind of input is the Jaccard Index. In order to find the Jaccard Index, instead of doing the final step of combining all the vectors of a word into one vector, the list of strings were put into a set.

```
# a and b are arrays of vectorized words
def jaccard_index(a, b):
    a = set(a)
    b = set(b)
    c = a.intersection(b)
    return float(len(c)) / (len(a) + len(b) -
        ↪ len(c))
```

**Fig. 11.** Calculating the Jaccard Index.

### E. Classification Algorithms

All of the Pyspark classification algorithms take in a labeled point, where the first index is 0 or 1, and the second index of the labeled point is the array containing the features extracted. Naive Bayes and Logistic Regression did not require fine tuning of the parameters for training. However, for Random Forest - after trial and error - it was found that setting max depth to 10 and the number of trees to 20 yielded the most optimal results. For the feed forward neural network, the best parameters were the default ones, where the intermediate layers of sizes 5 and 4.

## 11. RESULTS

The keys serve as a shared value for both the results table. Each key was a certain method for turning the sentence into some vector. Then each one of the methods of vectorizing words were input into each classification algorithm, where the classification algorithm was measured using its precision, which are the percentages shown below.

| Features | Similarity Measure | Key |
|---|---|---|
| TFIDF sum | difference | A |
| TFIDF mean | difference | B |
| w2v sum | cosine | C |
| w2v sum | sqeuclidean | D |
| w2v mean | cosine | E |
| w2v mean | sqeuclidean | F |
| jaccard | nothing | G |
| TFIDF*w2v sum | cosine | H |
| TFIDF*w2v sum | sqeuclidean | I |
| TFIDF*w2v mean | cosine | J |
| TFIDF*w2v mean | sqeuclidean | K |
| TFIDF*w2v sum, Jaccard | cosine | L |
| TFIDF*w2v sum, Jaccard | sqeuclidean | M |
| TFIDF*w2v mean, Jaccard | cosine | N |
| TFIDF*w2v mean, Jaccard | sqeuclidean | O |

| Key | Naive Bayes | Logistic Regression | Neural Net | Random Forest |
|---|---|---|---|---|
| A | 63.22 % | 63.14 % | 63.29 % | 63.24 % |
| B | 63.02 % | 63.12 % | 63.36 % | 63.21 % |
| C | 63.38 % | 36.72 % | 67.59 % | 67.36 % |
| D | 63.26 % | 63.08 % | 63.00 % | 65.25 % |
| E | 63.26 % | 36.71 % | 67.78 % | 67.47 % |
| F | 63.10 % | 61.37 % | 65.18 % | 65.18 % |
| G | 61.41 % | 63.25 % | 65.33 % | 65.88 % |
| H | 63.10 % | 63.35 % | 66.66 % | 66.62 % |
| I | 63.38 % | 63.35 % | 63.22 % | 63.78 % |
| J | 63.32 % | 63.24 % | 67.03 % | 66.64 % |
| K | 63.35 % | 63.18 % | 63.22 % | 64.58 % |
| L | 63.47 % | 63.29 % | 68.56 % | 69.20 % |
| M | 63.31 % | 63.21 % | 67.74 % | 68.09 % |
| N | 63.16 % | 63.17 % | 68.61 % | 69.00 % |
| O | 63.09 % | 63.36 % | 67.48 % | 67.71 % |

| Key | L | M | N | O |
|---|---|---|---|---|
| Random Forest | 36.90 % | 36.75 % | 36.99 % | 37.03 % |

### A. Matching Similar Questions

K-Means was the first solution that was thought of, but the algorithm did not work due to the fundamental differences between clustering and pairing. Since clustering tries to minimize the variance, no matter how many clusters there are there is no guarantee that each cluster will have 2 questions. LSH Forest was the best method to find the distances and indexes of all the questions that matched. However, it was lacking in the sense that it did not work for large amounts of data. The final results came from the LSH Forest, because it yielded a more reliable way of matching questions.

### B. Evaluating Similar Questions

With the first three algorithms of Naive Bayes, Logistic Regression, and SVM, using different methods of similarity and vectorizing the sentences, the difference in results was negligible. However, a combination of Word2vec and TFIDF did very well in the three. The neural network and random forest performed the best. Word2vec outperformed the TFIDF and Jaccard score by themselves in these two. The results also show that the cosine distance is a more accurate measure in each one as well over the squared euclidean distance. Similarly, sum seemed to outperform mean when combining all of the individual word vectors into one vector. Adding the Jaccard Index of two sentences also improved the accuracy of the algorithm. The highest accuracy was using TFIDF * Word2vec vectors summed up and the cosine distance, along with Jaccard index which had an accuracy of 69.20 %. This method improves upon the current baseline method vector based similarity that has an accuracy of 65.4 % (Mihalcea et al).

## 12. ERRORS

### A. Errors in Matching

Most of the errors that were obtained when matching the questions came from the lacking tools in Spark 1.5.2. When MLLib's Word2Vec was used many memory leaks occurred. Since the program was written in python not much could be done to fix the leaks. Other errors came when an LSH Forest was created using a large amount of data. The Forest only creates a set sized tree and if two hashes match, it picks a random hash based on the set seed value.

### B. Errors in Classifying Questions

After parsing the data and extracting features, some of the data would be be NaN or lower than 0, which would throw errors with the classification models, so those values had to be filtered out. Calculating the cosine distance on certain values would throw an error, so a try-catch statement was used to handle it, and those values would be given the highest cosine distance signifying that the two vectors were farthest apart.

## 13. CONCLUSION

The model was able to correctly evaluate whether two questions are the same for a significant percentage of the dataset. The clustering methods produced less than optimal results. However, classifying similar questions with these simple methods turned out to be fairly accurate, showing that Word2vec is a good indicator of the meaning of sentences. There are more advanced techniques of classifying similar questions that yield higher accuracies, but this method yielded around a 70 % accuracy using relatively simple methods.

## 14. SOURCES

1. Spark Overview. (2015). Retrieved April 26, 2017, from https://spark.apache.org/docs/1.5.2/index.html
2. Spark Overview. (2015). Retrieved April 26, 2017, from https://spark.apache.org/docs/1.5.2/index.html
3. sklearn.neighbors.LSHForest (2013). Retrieved April 26, 2017, fromhttp://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LSHForest.html
4. Bogdanova, D., Santos, C. D., Barbosa, L., & Zadrozny, B. (2015).Detecting Semantically Equivalent Questions in Online User Forums. Proceedings of the Nineteenth Conference on Computational Natural Language Learning. doi:10.18653/v1/k15-1013
5. Classification and regression. (2017). Retrieved April 26, 2017, from https://spark.apache.org/docs/2.1.0/ml-classification-regression.html
6. Classification and regression. (2017). Retrieved April 26, 2017, from https://spark.apache.org/docs/2.1.0/ml-classification-regression.html
7. Distance computations (scipy.spatial.distance)¶. (2016). Retrieved April 26, 2017, from https://docs.scipy.org/doc/scipy-/reference/spatial.distance.html
8. (n.d.). Retrieved April 26, 2017, from http://www.mit.edu/~andoni/LSH/
9. Tsvetkov, Y., Faruqui, M., & Dyer, C. (2016). Correlation-based Intrinsic Evaluation of Word Vector Representations. Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP. doi:10.18653/v1/w16-2520
10. Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining (n.d.). Retrieved April 26, 2017, from http://www.tfidf.com/
11. MLlib - Clustering. (2015). Retrieved April 26, 2017, from https://spark.apache.org/docs/1.5.2/mllib-clustering.html
12. (n.d.). Retrieved April 26, 2017, from https://stackoverflow.com/
13. First Quora Dataset Release: Question Pairs (2017). Retrieved April 26, 2017, from https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
14. Mihalcea, R., Corley, C., and Strapparava, C. (2006). Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity. Proceedings of the National Conference on Artificial Intelligence (AAAI 2006), Boston, Massachusetts, pp. 775-780.