**《Fundamentals of Programming》**

**Curriculum Design Report**

Subject ：Word classification based on KNN idea

Specialty ：Highly accurate text classification

Class ：19lq CST

Student’s Name ：MUNTASIR MD FAHIM王一然

School Number ：1911521213

Teacher ：Mr. Chao Li

2020/ 12 / 21 ～ 2020/ 12 / 28 /

**Produced by School of Computer**

Menu

1. Requirements Analysis ………………………………………………………………… 3

2. General Design …………………………………………………………………………… 3

3. Detailed Design ………………………………………………………………………… 4

4. Tests And Analyses of Program Running Results ………………………………… 7

5. Complimentary Close …………………………………………………………………… 8

6. References ………………………………………………………………………………… 9

**The Main Body**

1.Requirements Analysis

This paper demonstrates that highly accurate text classification may be obtained through word embedding representations of documents in combination with a K-Nearest Neighbor algorithm. This study found that document embeddings with a Distributed Bag of Words model in combination with KNN not only scores higher than other TF-IDF vector KNN approaches, but rivaled the dominant SVM approach in terms of accuracy.

Keyterms: Keyword Extraction, Text Classification, Word Embedding, Word2Vec, Term Frequency – Inverse Document Frequency, Doc2Vec, K-Nearest Neighbor

**Description of files:**

(1) BBC\_doc2Vec.py : The main KNN algorithm used for computations. To run properly, the user must pass three flags or four flags. The first is a number representing the k value to evaluate. The second is either -k or -m. -k will plot KNN and -m will show the k-means of the model. The third, -y will retrain the document, -n will not. if the user would like to see the accuracy evaluation, the fourth flag is -a.

Ex: python BBC\_doc2Vec.py 35 -k -n

Ex: python BBC\_doc2Vec.py 35 -k -n -a

Ex: python BBC\_doc2Vec.py 35 -m -y

(2) tfidf.py will extract keywords based on the model. Just run from command line along with the number of keywords you would to get.

Ex: python tfidf.py 10

(3) stemmed\_text.py: this simply stems the abc documents into one file, “bbc\_stemmed.txt”.

Doc2Vec + KNN Text Classification

2.General Design

Word embeddings capture many semantic relationships and meanings from text. Word vectors quantify a word based on its relationship. Proximity, and usage relative to other words.

Unfortunately, word vectors do not capture the specific context of the word at the instance of usage. For example, without proper context, chicken could refer to the literal animal, a coward, or dinner. But through the use of document embeddings, or paragraph embeddings, the entire document may be represented as a word embedding defined by all the words within the documents. This would mean a word embedding with a specific and narrow context and setting as opposed to a general word embedding capturing all possible relationships.

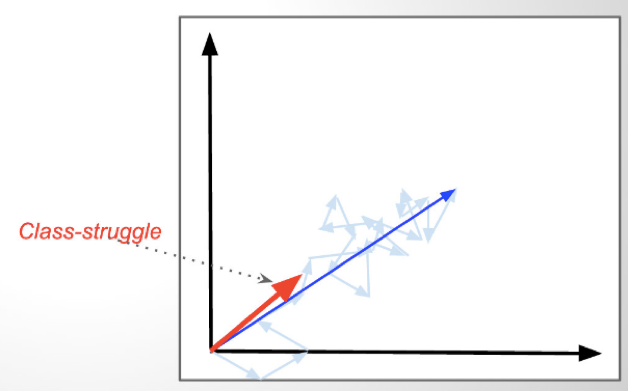


Figure 1: The above graph shows several word embedded vectors found in &quot;The Tale of Two Cities.&quot; Each individual word has its own individual vector direction. When averaged together, we get one singular vector representative of the entire document. In this case, that average vector resembles &quot;Class Struggle.&quot; 3

**Dataset: BBC**

The dataset used was the BBC news dataset. This dataset consists of 2225 short news articles for five categories from 2004-2005: 'Business,' 'Sport,' 'Politics,' 'Entertainment,' and 'Tech.' The dataset was chosen for its simplicity of only five categories and because the the categories 'naturally' feel different (i.e. sport and technology typically wouldn't be clustered together by a human annotator). Additionally, this dataset has been relatively underused. It should be noted that the articles are not evenly distributed (Ex: sport and business articles form roughly 46% of the dataset. This is relevant more so for when introducing KNN because if a certain article type is represented more, it may skew the algorithm to label articles as sport or business.

**Preprocessing Phase**

First the user will receive all data from the BBC news articles. The articles are labeled in separate folders. The user goes through and labels all folders as falling under one of the five pre-assigned categories: Business, Sports, Politics, Entertainment, and Tech. The data is then preprocessed by removing all stop words, utilizing Named Entity Recognition, stemming words, and sending all letters to lowercase. Lastly, the words are POS tagged.

1. Detailed Design

**Evaluating The Challenges involved in Word Embeddings**

As mentioned before, word embedding's can be incredibly effective, yet they still perform significantly worst than the typical SVM and LDA models which can achieve in some cases more than 90% correctness. The reasons and issues further explained are: the algorithm may pick up on connections not considered with human annotators and the topic of a documents can be ambiguous or have multiple categories.

\*\*Facebook is business, technology, and entertainment: \*\***Most News Documents aren't just one category.**

The issue with pinpointing the correct category is that a category could have a justifiably equal connection to multiple categories. Take for example this title from one document: "Ink helps drive democracy in Asia." The remainder of that article talked about the relationship of ink and the coming election in a certain area of Asia. The majority of readers would inherently think the category of the article is politics, but the article is actually talking about the technology of ink and thus is labeled as "tech." Esoteric category labeling was found abundantly within the news corpus. To visualize the space of these documents, Gensim's Doc2Vec library was used along with PCA and K-Means. The results show that it may be possible to use a KNN approach of labeling new incoming documents which place on the outer rims of the cluster. But for those documents which fall closer to the center, the model would struggle to solve.

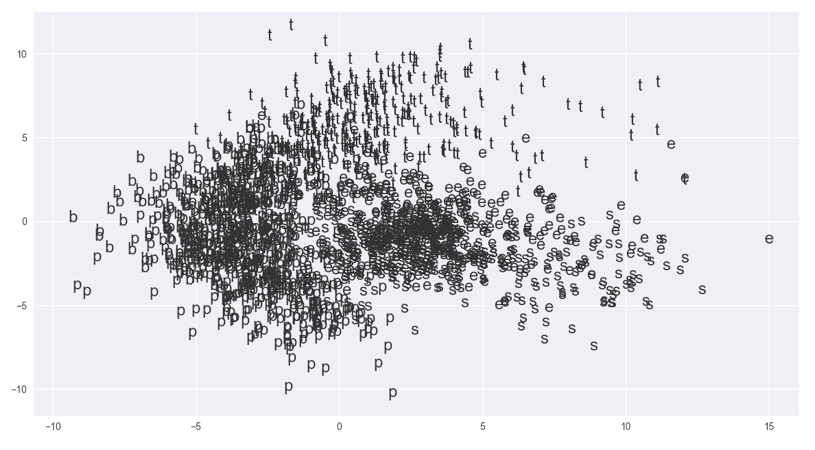


Figure 2: PCA Representation with single character labels. Notice the central overlap.



Figure 3: K-means of PCA Representation. Centroids are darkened circular centers. Notice how the center overlaps.

**The Model: Document Embedding + KNN**

Through vectorizing the entire document into a word embedding space, we capture the context and theme of the entire document as opposed to individual word vectors. The unlabeled document embedded vector is then mapped to a graph along with other pre-categorized documents. The unlabeled document takes on the label of the K nearest labeled documents. This is the k-nearest neighbor model with document vectors as inputs.

The results show a viable KNN classification approach that beats the traditional KNN utilizing TF-IDF vector algorithm on the same BBC dataset.

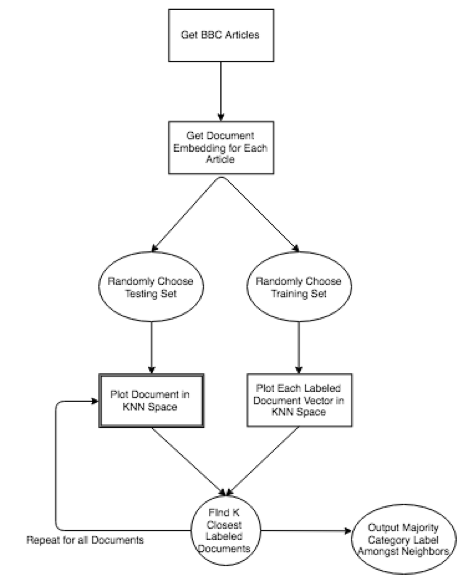


Figure 4: A higher level overview of the algorithm

**KNN Overview**

KNN functions by first plotting several pre-labeled inputs onto an N-dimensional graph. An unlabeled input will be plotted within the graph. We check for its closest K neighbors. The label of the unmarked input will be whatever is found to be the majority category amongst the closest K neighbors.

First the documents are vectorized using Gensim's Doc2Vec with a Bag-Of-Word model. After modeling of the documents is finished, 90 percent of the articles are randomly chosen to be used as the training set and the remaining 10 percent are the test set. The test set is then labeled according to the KNN algorithm. The document model uses a vector size of 100 dimensions, a window size of 10, and is trained over 100 iterations with a minimum word frequency of 1 to capture all words.

**Computing Average Top Accuracy**

The program took 1000 randomly chosen training and test sets and computed the average of the accuracy given a k. The best results were found to be around K=35 and K=105 with an accuracy score, Precision, Recall, and F1-Score averaging about 95%

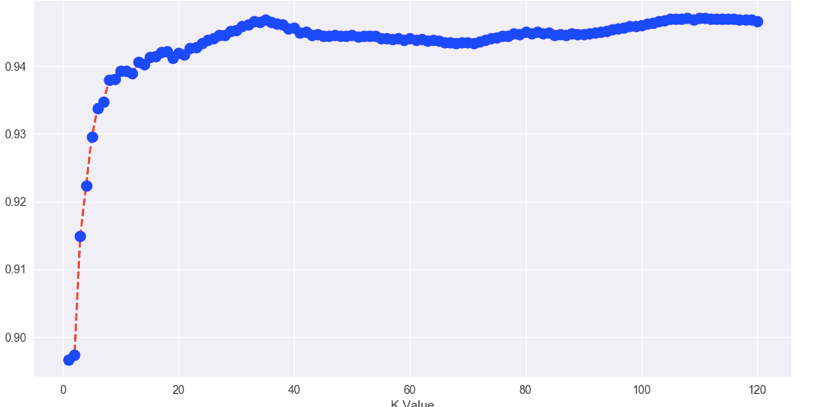


Figure 4: Graph of average accuracy score per given k. Average accuracy was computed over a 1000 iterations of randomly selected testing and train sets. Accuracy is the worst when searching for the first closest neighbors at K=1 and peaks at K =35 and 105.

4.Tests And Analyses of Program Running Results

The KNN scored on average more than 94% accuracy amongst the top 120 K scores. The best K values were found to be 35 and 105, both achieving more than 95% accuracy, and, in some cases, even 98%. The highest scoring Recall and F1-Score was found to be Sports documents. This would make sense as sports seems, on face value, to share the least in common with the other four categories. Sports was also found to be the highest scoring category within those articles. Technology conversely generally scored the worst in recall and F1-Score.

Word Embedding KNN Model Compared to Others

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| TF-IDF + N-Gram + BoW KNN | 49% |
| TF-IDF + Feature Selection KNN | 88.5509% |
| **Document Embedding BoW + KNN** | **95.5157%** |
| SVM | 97.6744% |

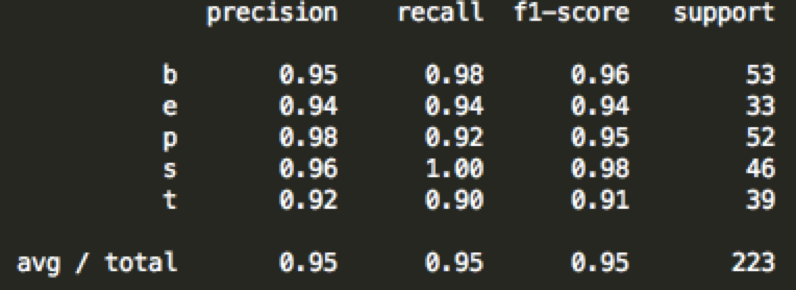
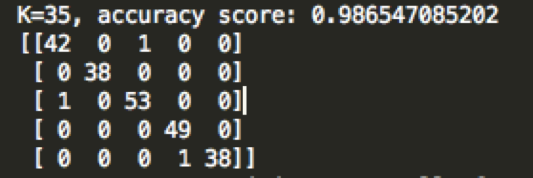


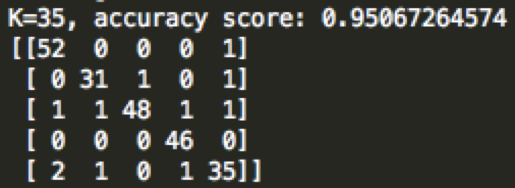
Figure 6: Precision, Recall, and F1 Score per each article type. The left most column is the first character of each category. Support is the number of article types per test set. The bottom row shows an average of 95% amongst the three.

**KNN Error Analysis**

First, the core issue with document classification is that some articles are highly ambiguous. It may be that an article shares similarities to both business and politics. The KNN approach has removed most of that error but not entirely.

Second, the algorithm is highly dependent upon the initial training set. Take for example the following two confusion matrices. Both tables utilize a K value of 35, but one scored much better than the other, nearing a 98% correct rate. The reason is solely due to the randomness of what documents are chosen to be the training and test set.





5.Complimentary Close

This paper has shown the advantage of using KNN and Document Embeddings as opposed to the traditional TF-IDF vector approach. While not yet outperforming SVM, this approach still comes within striking distance. The next phases of this algorithm will incorporate other datasets commonly used, specifically the Reuters dataset, for comparison with other mainstream algorithms. Additionally, I will introduce another metric of comparison between neighbors found and the test article, such as weighing closer neighbors as being more important than farther away neighboring articles.

6.References

* Kusner, Sun, Kolkin, and Weinberger. "From Word Embeddings to Document Distances" Washington University in St. Louis. <http://proceedings.mlr.press/v37/kusnerb15.pdf>
* Le, Quoc and Mikilov, Tomas. "Distributed Representations of Sentences and Documents" Google. <https://cs.stanford.edu/~quocle/paragraph_vector.pdf>
* Tamir, Mike Phd. "Short Introduction to Using Word2Vec for Text Classification. LinkedIn. <https://www.linkedin.com/pulse/short-introduction-using-word2vec-text-classification-mike/>
* Jin, Zhang, Chen, and Xia. "Bag-of-Embeddings for Text Classification". Joint Conferce on Artificial Intelligence. <https://www.ijcai.org/Proceedings/16/Papers/401.pdf>
* Kawa, Nura. "Text Classification" University of California, Berkley. <https://www.stat.berkeley.edu/~aldous/Research/Ugrad/Nura_Kawa_report.pdf>