Doc2Vec Vector Representations Can Provide Significant Improvement on Text Classification Compared to Well-Established Methods

Christopher Festa

[cfesta@gmail.com](mailto:cfesta@gmail.com)  
https://www.linkedin.com/in/chris-festa/  
+1 (516) 287-7496

Northwestern University

February 14, 2021

**Abstract**

Well established term frequency-inverse document frequency (TF-IDF) and term count vectors have widely been used by data scientists because they are both prominent and perform well on document classification with algorithms such as the classic K-means algorithm. The aim of this study is to show that the Doc2Vec algorithm can significantly improve the performance of document classification over the more well-established TF-IDF and term count methods. A set of experiments was performed on the news category dataset from Kaggle[[1]](#footnote-1), containing documents from the Huffington Post and the associated category of each document capturing the theme of the article. Results suggest that use of vectors generated by the Doc2Vec algorithm will cluster documents into the correct category more accurately than other established algorithms and also predict clustering of new documents more accurately. Sample comparisons are provided using the results of supervised classification achieved by the classic K-means algorithm and the Rand Index score.

*Keywords:* Doc2Vec, TF-IDF, Term Counters, K-Means, Clustering, Text Classification

**Introduction and Problem Statement**

A reference term vector is an algebraic model used to represent text in documents so that the text can be used in clustering and classification applications. Advancements have been made in reference term vector representations claiming to perform better in Natural Language Processing (NLP) models, significantly expanding its business applications and use cases (Kim et al. 2019). This study evaluated the effectiveness of three different approaches to creating a reference term vector using a multi-label news category dataset from Kaggle[[2]](#footnote-2) to identify how much of an improvement select newer methods provide. This dataset contained news headlines and short descriptions with labels that correspond to the document theme (e.g., politics, comedy, sports, entertainment, etc.). The three approaches used in this study are as follows:

* **Term Count Vectors**: a qualitative approach used by selecting terms from the corpus
* **Term Frequency – Inverse Document Frequency (TF-IDF)**: a statistical approach that reflected how important a word is to a document in the corpus
* **Doc2Vec**: an unsupervised approach that found similarity between sentences, paragraphs, and documents in the corpus

The effectiveness of each approach was tested using a K-means model and allowed for the evaluation of different vector representations and their capabilities in natural language processing by evaluating how well the K-means model was able to predict the theme of each document. According to Novotný and Ircing (2018), a simple classification algorithm such as the classic K-means clustering method is generally accepted for data clustering if it is given an appropriate feature vector. This study concluded by evaluating the similarity measure between the actual and predicted themes using the sklearn package implementation of Rand Index[[3]](#footnote-3) adjusted for chance.

**Literature Review**

Popular use of the internet has put increasingly higher demands on use cases for text classification similar to the multi-label news category corpus used in this study. One simple example that could be applied to this study’s corpus is performing a category search or clustering on the document set. Historically, the bag-of-words (BOW) approach achieved satisfactory results using supervised machine learning methods. However, the BOW approach is known to have a low tolerance for the semantics of words and fails to achieve adequate performance on text classification. Newer methods such as term vector representations have become more prominent since they can be learned via language modeling or encoding word meaning with a probabilistic approach. To apply mathematical techniques, text first must be transformed into a numerical representation also known as a vector which enables deep learning techniques to perform the text classification (Zhang and Zhong 2016). In this study, three term vector techniques were tested: (1) term count vectors, (2) term frequency-inverse document frequency (TD-IDF) vectors, and (3) Doc2Vec vectors. The term vectors were then evaluated using the classic K-means algorithm from the sklearn package[[4]](#footnote-4) to assess their efficacy in deep learning.

The most basic method to convert text into vectors is using a count vectorizer. This approach converts the document into a vector of counts using a given vocabulary list. The vocabulary list is typically created manually after inspecting the terms across the corpus and identifying the terms that are prevalent. The terms are then assigned an index in the vector and the number of times it appears in the document is stored in that index. This is repeated for each document (Bhanot 2019).

The main drawback using a count vectorizer is that it does not consider the importance of the word across the corpus. For example, if a three-document corpus contains a term that occurs in all three documents, the term will provide no useful information in differentiating between the three documents. Spärck Jones (1972) introduced in a paper that query terms that are too frequent across the corpus are not suitable terms and should be given less weight than terms that occur in fewer documents. This later became known as inverse document frequency (IDF), the inverse of the frequency of documents that contain the term, irrespective of how many times it occurs in each document. IDF remains at the center of most ranking methods used in search engines today. IDF combined with term frequency (TF) can then be used to calculate a weighted score of the term per document that is based on the frequency of the term within a document and the inverse of the frequency of documents that contain the term (Robertson 2004). The simple formulas that underpin the TF-IDF statistical measure are:

|  |  |  |
| --- | --- | --- |
|  | Term Frequency Formula (TF) | (Equation 1) |
|  | Inverse Document Frequency Formula (IDF) | (Equation 2) |
|  | Term Frequency-Inverse Document Frequency (TF-IDF) | (Equation 3) |
| (Borcan 2020) | | |

represents the number of documents in the corpus, is a given document in our corpus, is the collection of all documents, is a given word in a document, and is the natural logarithm. Equation 1 computes the term frequency (TF), where is the frequency of word in document . Equation 2 computes the inverse document frequency (IDF), where is the frequency of word in corpus . This number will be lower with more appearances of the word in the corpus. Finally, equation 3 computes the TF-IDF weighted score by multiplying equation 1 with equation 2.

Text analytics generally rely on the well-established count vectorizer or TF-IDF approach, however these approaches are not without drawbacks. Both rely on a bag-of-words philosophy, which purports that the basis that a document is simply a collection of words. This approach is linear; the semantics of the terms and how they interact with nearby terms and the document as a whole is lost. This is leads to the third approach explored in this study, Doc2Vec. Doc2Vec is newer amongst the three approaches and is an extension of word-to-vector (Word2Vec) which was published in 2013 by a team of researchers led by Tomas Mikolov at Google. In Word2Vec, a word is regarded as a single vector and the element values of a word are affected by other words surrounding the target word. Doc2Vec is the natural extension of Word2Vec at the document level; each document has its own vector values in the same space as that for words and is an unsupervised learning algorithm that maps text data to a fixed-length vector with the goal of capturing the semantics within the document (Kim et al. 2019). Doc2Vec has been known to achieve higher classification accuracy where sentiment classification is important such as IMDB movie reviews, news categorization, and forum discussions.

Measuring the performance of these three vector representations in this study will be evaluated by how well the documents in the corpus are classifying the predicted labels into clusters and how accurate the prediction is compared to the actual label using the classic K-means algorithm. K-means was first proposed over 50 years ago and it remains one of the most popular used algorithms for clustering (Jain 2010). Vectors are used as features in the K-means algorithm and will have a significant influence on the performance of the K-means clustering algorithm. If the vectors are suitable, the clusters are will be compact and isolated; simple clustering algorithms such as K-means will be able to find them. For example, in Figure 1a and Figure. 1b the impact of using suitable features on how well the K-means algorithm can identify two clusters is easily identifiable. The dashed line in Figure 1a shows the linear cluster separation boundary, and the K-means algorithm fails to find the two clusters. However, in Figure 1b, the K-means algorithm can easily detect the two clusters. For the purpose of this study, it is expected that the best preforming vector will look more like Fig 1b (Jain 2010).

![Chart, scatter chart

Description automatically generated]()

Figure 1: The importance of suitable features in clustering algorithms can be visualized by how well the separation of clusters are represented in a scatter plot. In figure 1a, poor separation of clusters is displayed, whereas Figure 1b depicts a clear separation. The dashed line in figure 1a represents the linear cluster separation boundary and the K-means algorithm fails to find the two clusters. In figure 1b the points in the scatter plot are clustered together, therefore the K-means algorithm are accurately identifying the two clusters (Jain 2010).

The final comparative measure of cluster performance in this study is how well the K-means algorithm predicts the cluster partition by comparing the actual clusters versus the predicted clusters. When comparing two clusters, the Rand index (RI) and adjusted Rand index (ARI) are commonly used to evaluate a model’s classification performance (D’Ambrosio et al. 2020). Rand index is a function that computes the similarity between the actual and predicted by considering all pairs of samples and counting pairs that are assigned in the similar or different clusters in the predicted and actual clustering. Afterwards, the raw Rand index score is adjusted for chance into the adjusted Rand index score (D’Ambrosio et al. 2020).

**Data**

The corpus used in this study is the multi-label news category dataset from Kaggle. The corpus contains 202,372 news headlines from the year 2012 to 2018 obtained from Huffington Post. Each record contains the authors of the article, a link to the article, the headline of the article, a short description of the article, the date the article was published, and the category to which the article belongs. This study used the headline and short description from the article as the text for each document. Category is defined as the label that is the target cluster partition derived from the combined text of the headline and short description, which will be converted into a vector representation across the corpus. Forty-one categories were defined within the selected data set including politics, education, environment, sports, entertainment, etc. The study did not consider the entire corpus; only the nine most frequently identified categories were used to evaluate the cluster partitions. These nine selected categories expedited algorithm training time while maintaining sufficient differentiation to evaluate the performance of each approach:

* Business
* Entertainment
* Food & Drink
* Healthy Living
* Parenting
* Politics
* Queer Voices
* Style & Beauty
* Travel

The corpus was further reduced to 20,000 using a random sample.

Before the creation of any vector representation, the documents were preprocessed and cleaned using a predefined approach. Each document was split into tokens and evaluated for punctuation, non-alphabetic and stop words using the natural language toolkit (NLTK) library[[5]](#footnote-5). All tokens determined to be in these conditions were removed from the document. The remaining tokens were converted to lowercase and were lemmatized using the WordNetLemmatizer method from the NLTK library[[6]](#footnote-6). For reference, lemmatization is the process of converting a word to its base form by considering the context and converting the word to its meaningful base form. For example, lemmatization would convert the word ‘walked’ into ‘walk’.

**Methods**

Splitting a dataset into train and test segments is an important part of evaluating a model’s performance in machine learning algorithms such as K-means. In this study, the 20,000 documents obtained were split into a train and test dataset; 16,000 documents were included in the train dataset and 4,000 documents in the test dataset. The train dataset was used to train the term count, TF-IDF, and Doc2Vec vectors and was also used to train the K-means model allowing for the smaller test dataset to be used for testing and evaluating the performance. The hyperparameters for each approach were tuned for best performance using GridSearchCV[[7]](#footnote-7) from sklearn which performed an exhaustive search of each parameter to identify the best combination. The pipeline included the vector representation and the K-means classification algorithm. The best combination of parameters identified from the search was used as the default parameters for this corpus.

The term count vectors were created by selecting terms that occurred over 5 times across the corpus which formed the vocabulary used in the sklearn implementation of the CounterVectorizer[[8]](#footnote-8). A ngram\_range parameter with a lower boundary of 1 and an upper boundary of 3 was used for the purposes of this study. An n-gram is a string of n words in a row. For example, when using the default setting of 1, the term 'data scientist' would treat 'data' and 'scientist' as two different terms, where an increased ngram\_range with an upper boundary of 2 would treat 'data scientist' as one term. The TF-IDF vectors for this study were created using the sklearn implementation of TfidfVectorizer[[9]](#footnote-9) using a similar ngram\_range parameter of (1,3) and would not have more than 8,000 features. The vocabulary used for the TF-IDF vector ignored terms that have a document frequency less than 5 and if a document frequency was higher than ninety-five percent of the documents. Finally, the Doc2Vec vectors were specified to have a length of 500 and used the gensim implementation of Doc2Vec paragraph embeddings[[10]](#footnote-10) with the hyperparameters as follows:

* window = 3, the maximum distance between the current and predicted word in a sentence
* min\_count = 1, ignores all words with a lower frequency than 1
* epochs = 15, the number of iterations over the corpus

Using these parameters, the objective in this study was to identify partitions or clusters from the text of the news category dataset from Kaggle. Specifically, given a vectorized representation of the headline and short description of each document, how accurately can each algorithm cluster and predict the category of each article in the dataset? To test these hypotheses, the three vector representations reviewed in the previous sections and the K-means algorithm were used to create a scatter plot of each representation, which tested how well the model partitioned the documents into clusters. The hyperparameters used to train the K-means model were:

* init = k-means++, selects initial cluster centers for k-mean clustering in a smart way to speed up convergence
* max\_iter = 100,000, maximum number of iterations of the k-means algorithm for a single run.

The best performing method would result in a similar visual representation to Figure 1b in the previous section. Finally, the predicted category of the documents in the train and test datasets were computed, then an adjusted Rand Index score for each algorithm would be calculated. The highest score would represent a vector that provided better features for use in the K-means algorithm.

**Results and Analysis**

The code used in this study can be found on GitHub[[11]](#footnote-11). In Tables 1 and 2, the distinction between how important terms are determined in a term count vector versus the TF-IDF vector are illustrated. Term count vectors placed a higher priority on terms that are prevalent throughout a category, whereas TF-IDF ignored these terms because they were too frequent in the category. A similar illustration on term counts and prioritization for the Doc2Vec algorithm does not provide additional insight due to its focus on sentences and paragraphs rather than words and short phrases.

|  |  |  |  |
| --- | --- | --- | --- |
| **Category: Politics** | | | |
| **Count Vector** | | **TF-IDF** | |
| **Word** | **Count** | **Word** | **Score** |
| trump | 1517 | convictions | 0.185388 |
| donald | 682 | marijuanarelated | 0.116534 |
| donald trump | 525 | erasing | 0.116534 |
| president | 461 | offenses | 0.109493 |
| trumps | 386 | haunt | 0.104856 |
| clinton | 374 | public opinion | 0.104856 |
| house | 342 | opinion | 0.097426 |
| hillary | 307 | prosecutors | 0.093461 |
| obama | 300 | public | 0.071451 |

Table 1: Common terms such as politicians as seen above are prioritized in a term count vector, whereas politicians are less important in TF-IDF.

|  |  |  |  |
| --- | --- | --- | --- |
| **Category: Business** | | | |
| **Count Vectorizer** | | **TF-IDF** | |
| **Word** | **Count** | **Word** | **TF-IDF Score** |
| business | 123 | customer | 0.070865 |
| people | 76 | customer service | 0.066358 |
| women | 70 | customer loyalty | 0.061446 |
| company | 49 | loyalty | 0.055463 |
| workers | 43 | service | 0.053266 |
| companies | 42 | manage customer loyalty | 0.036291 |
| change | 41 | loyalty build relationships | 0.036291 |
| years | 34 | management customer service | 0.036291 |
| industry | 33 | loyalty management customer | 0.036291 |

Table 2: Common terms such as ‘business’, ‘people’, and ‘women’ as seen above are prioritized in a term count vector; TF-IDF prioritizes the type of experience one would expect from a business.

The PCA scatter plots of the clusters identified from the K-means model using the three different vector representations used in this study are represented in Figures 2, 3, and 4. The triangle in the scatter plots represent the centroid point indicating the center of each cluster. Observing Figure 2 and Figure 3, most of the centroid’s points overlap and is similar to Figure 1a in the previous section where the K-means algorithm fails to identify the nine clusters. The TF-IDF scatter plot and associated centroid points in Figure 3 more accurately identified the clusters than term count vectors in Figure 2 but did not provide a significant improvement. However, the Doc2Vec scatter plots in Figure 4 depicts the centroids with clear separation and all nine clusters were identified with minimal overlap. Therefore, the comparison approach and method in this study demonstrates supporting evidence that Doc2Vec vector representation can provide a significant improvement in text classification.

![Chart

Description automatically generated with medium confidence]()

Figure 2: The PCA cluster plot of term count vectors does not perform well in identifying partitions or clusters. Several clusters overlap indicating that term count vectors are not suitable features in the K-means algorithm.

![Chart

Description automatically generated]()

Figure 3: The PCA cluster plot of TF-IDF vectors does not perform well in identifying partitions or clusters. Several clusters overlap indicating that term count vectors are not sufficient features in the K-means algorithm.

![Chart, scatter chart

Description automatically generated]()

Figure 4: The PCA cluster plot of Doc2Vec vectors performs well in identifying partitions or clusters. Compared with figure 2 and figure 3, Doc2Vec are sufficient features to use in the K-means algorithm. The nine clusters are clearly defined in the scatter plot.

The final objective of this study was to evaluate how well the K-means model predicted the category of each document using the given text. The predicted label would serve as an effective indicator for model performance on new documents and how well it could be classified into a news article category. Observing Table 3, the adjusted Rand Index on both the training and test datasets provided significant improvement using a Doc2Vec vector. Therefore, a Doc2Vec vector provides improved feature performance for use in clustering algorithms such as K-means.

|  |  |  |
| --- | --- | --- |
| **Vector Type** | **Training Adjusted Rand Index** | **Test Adjusted Rand Index** |
| Term Count Vector | -0.0101 | -0.0114 |
| TF-IDF Vector | 0.0148 | 0.0156 |
| Doc2Vec Vector | 0.0994 | 0.1329 |

Table 3: There is a significant performance improvement using the Doc2Vec vector in the K-means model when predicting the categories of documents based on the corpus used in this study.

**Conclusion**

In this study, three different vector representations of text data were examined and compared: (1) term counts, (2) TF-IDF, and (3) Doc2Vec. Term counts and TF-IDF are the more traditional methods that have worked well historically; however, semantics of the terms, their interaction with nearby terms and the document as a whole are not accurately represented or contextualized. Newer methods such as Doc2Vec account for the context and how nearby words interact with each other. This study provides evidence that for the given task of multi-label text classification, Doc2Vec vectors are a superior choice in terms of identifying clusters of similar documents and model performance in predicting the labels compared to traditional methods.

**Future Work**

Several open questions remain. First, the size of the corpus was limited to only nine categories and 20,000 documents. How would the performance of the three vector representations be impacted if the corpus were to expand to all forty categories and the full 202,372 documents in the corpus? Second, there are other newer methods such as BERT which are gaining popularity in natural language processing. Will newer methods such as BERT and GloVe vectors provide more accurate results than Doc2Vec in text classification? Lastly, the K-means algorithm was used to predict the category of each document from the dataset. Using a deep neural network such as a recurrent neural network has been known to perform well on multi-label text classification. A future study would require moving this study to the cloud where graphics processing units (GPUs) to train the models could be leveraged.

References

Bhanot, Karan. 2019. “Different Techniques to Represent Words as Vectors (Word Embeddings).” Medium. June 7, 2019. https://towardsdatascience.com/different-techniques-to-represent-words-as-vectors-word-embeddings-3e4b9ab7ceb4.

Borcan, Marius. 2020. “TF-IDF Explained And Python Sklearn Implementation.” Medium. June 8, 2020. https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275.

D’Ambrosio, Antonio, Sonia Amodio, Carmela Iorio, Giuseppe Pandolfo, and Roberta Siciliano. 2020. “Adjusted Concordance Index: An Extensionl of the Adjusted Rand Index to Fuzzy Partitions.” *Journal of Classification*.

Jain, Anil K. 2010. “Data Clustering: 50 Years beyond K-Means.” *Pattern Recognition Letters* 31 (8): 651–66.

Kim, Donghwa, Deokseong Seo, Suhyoun Cho, and Pilsung Kang. 2019. “Multi-Co-Training for Document Classification Using Various Document Representations: TF–IDF, LDA, and Doc2Vec.” *Information Sciences* 477: 15–29.

Misra, Rishabh. 2018. “News Category Dataset.” Unpublished. https://doi.org/10.13140/RG.2.2.20331.18729.

Novotný, Jaromír, and Pavel Ircing. 2018. “The Benefit of Document Embedding in Unsupervised Document Classification.” In *Speech and Computer*, 470–78. Lecture Notes in Computer Science. Cham: Springer International Publishing.

Pereira, Rafael B, Alexandre Plastino, Bianca Zadrozny, and Luiz H.C Merschmann. 2018. “Correlation Analysis of Performance Measures for Multi-Label Classification.” *Information Processing & Management* 54 (3): 359–69.

Robertson, Stephen. 2004. “Understanding Inverse Document Frequency: On Theoretical Arguments for IDF.” *Journal of Documentation* 60 (5): 503–20.

Spärck Jones, Karen. 1972. “A Statistical Interpretation of Term Specificity and Its Application in Retrieval.” *Journal of Documentation* 28: 11–21.

Zhang, Heng, and Guoqiang Zhong. 2016. “Improving Short Text Classification by Learning Vector Representations of Both Words and Hidden Topics.” *Knowledge-Based Systems* 102: 76–86.

(Novotný and Ircing 2018)

(Pereira et al. 2018)

(Spärck Jones 1972)

(Borcan 2020)

(Misra 2018)

1. The news category dataset can be found at: <https://www.kaggle.com/rmisra/news-category-dataset> (Misra 2018) [↑](#footnote-ref-1)
2. The news category dataset can be found at: <https://www.kaggle.com/rmisra/news-category-dataset> (Misra 2018) [↑](#footnote-ref-2)
3. The sklearn package implementation of the Rand Index can be found at: <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html> [↑](#footnote-ref-3)
4. The sklearn package implementation of the K-means algorithm can be found at: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> [↑](#footnote-ref-4)
5. The natural language toolkit (NLTK) library can be found at: <https://www.nltk.org/> [↑](#footnote-ref-5)
6. The NLTK implementation of the WordNet Lemmatizer can be found at: <http://www.nltk.org/api/nltk.stem.html?highlight=lemmatizer> [↑](#footnote-ref-6)
7. The sklearn package implementation of GridSearchCV can be found at: https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html [↑](#footnote-ref-7)
8. The sklearn package implementation of CountVectorizer can be found at: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html> [↑](#footnote-ref-8)
9. The sklearn package implementation of TfidfVectorizer can be found at: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> [↑](#footnote-ref-9)
10. The gensim implementation of Doc2Vec can be found at: <https://radimrehurek.com/gensim/models/doc2vec.html> [↑](#footnote-ref-10)
11. GitHub repository for the code that performed the analysis used in this study: <https://github.com/chrisfesta/NWU_MSDS453> [↑](#footnote-ref-11)