
# Movie Review Corpus Vocabulary Selection

Yining Feng

January 31, 2021

**Abstract**

With the rapid development in deep learning, deep neural networks have been widely adopted in many real-life natural language applications. Under deep neural networks, a predefined vocabulary is required to vectorize text inputs. The canonical approach to select predefined vocabulary is based on the word frequency, where a threshold is selected to cut off the long tail distribution. However, such a simple approach could easily lead to under-sized vocabulary or oversized vocabulary issues. Therefore, this study intends to understand how movie classification accuracy is related to the corpus vocabulary selection and what is the minimum required vocabulary size to enhance the performance of movie classification task.

## 1. Introduction & Problem Statement

Over the past decade, deep neural networks have become arguably the most popular model choice for a vast number of natural language processing (NLP) tasks and have constantly been delivering state-of-the-art results. Because neural network models assume continuous data, to apply a neural network on any text data, the first step is to vectorize the discrete text input with a word embedding matrix through look-up operation, which in turn assumes a pre-defined vocabulary set. For many NLP tasks, the vocabulary size can easily go up to the order of tens of thousands, which potentially makes the word embedding the largest portion of the trainable parameters. For example, a document classification task like AG-news (Zhang et al., 2015) can include up to 60 thousand unique words, with the embedding matrix accounting for 97.6% of the trainable parameters, which leads to under-representation of the neural networks' own parameters.

The purpose of this study is to employ a qualitative approach to the corpus vocabulary selection problem. Specifically, the goal of this study is to answer the following questions:

1. What terms are important in at least one document?

2. What terms are prevalent in at least two or three of the documents?

## 2. Literature Review

Text classification and clustering can be done by using machine learning algorithms such as logistic regression or K-means. Vectorizing words can help determine how documents or reviews differ from one another through word frequency (Jang, Kim, & Kim, 2019). Furthermore, research has been conducted to use entire sentences instead of processing words in order to maintain language structure and semantics (Büschken & Allenby, 2016). A mix of sentence and word clustering was tried to classify news topics (Yow & Tan, 2013).

Intuitively, using the full or very large vocabulary are neither economical, as it limits neural network model applicability on computation- or memory-constrained scenarios (Yogatama et al., 2015; Faruqui et al., 2015), nor necessary, as many words may contribute little to the end task and could have been safely removed from the vocabulary. Therefore, how to select the best vocabulary is a problem of both theoretical and practical interests. Somewhat surprisingly, this vocabulary selection problem is largely under-addressed in the literature: The de facto standard practice is to do frequency-based cutoff (Luong et al., 2015), and only retain the words more frequent than a certain threshold. Although this simple heuristic has demonstrated strong empirical performance, its task-agnostic nature implies that likely it is not the optimal strategy for many tasks. Task-aware vocabulary selection strategies and a systematic comparison of different strategies are still lacking.

## 3. Research Design and Modeling Method

### 3.1 Data Description and Processing

The dataset is comprised of 42 movie review documents collected from New York Times, Washington Post, RogerEbert.com, The Seattle Times, Vanity Fair, IGN, Hollywood Reporter, Breathe Dream Go, The Guardian, Variety, NPR, and Cinemajam. Each movie review document is trimmed to 500 words and contains the movie title, plot summary, release date, actors /actresses, directors, movie review texts and so on. For data pre-processing, the first step is to transform movie review texts into lower case. This avoids having multiple copies of the same words. The next step is to remove punctuation since a punctuation does not add any extra information. Therefore, removing all instances of punctuation will help reduce the size of the input dataset. Subsequently, lemmatization, stemming, stop word removal are implemented to convert the word into its root word, remove suffices, and eliminate commonly occurring but

meaningless words, respectively. The resulting dataset includes 42 unlabeled movies reviews trimmed from 21,896 terms to 9,835 terms.

### 3.2 Methodology

#### TF-IDF

It is necessary to convert documents to fixed-length vectors of numbers since it is not practical to work with text directly when using clustering algorithms. Each encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document. One issue with simple word counts is that some words like *the* will appear many times and their large counts will not be very meaningful in the encoded vectors. An alternative approach is to calculate word frequencies, and the method called Term Frequency – Inverse Document Frequency (TF-IDF) is employed in this study. TF-IDF are the components of the resulting scores assigned to each word.

With TF-IDF, instead of representing a term in a document by its raw frequency (number of occurrences) or its relative frequency (term count divided by document length), each term is weighted by dividing the term frequency by the number of documents in the corpus containing the word. The overall effect of this weighting scheme is to avoid a common problem when conducting text analysis: the most frequently used words in a document are often the most frequently used words in all of the documents. In contrast, terms with the highest TF-IDF scores are the terms in a document that are distinctively frequent in a document, when that document is compared with other documents. *TfidfVectorizer* is deployed to tokenize a collection of text documents, build a vocabulary of known words, inverse document frequency weightings across all documents, and enable encoding new documents using that vocabulary.

#### Word2vec & Doc2vec

*word2vec* was proposed as an efficient neural approach to learning high-quality embeddings for words (Mikolov et al., 2013a). The objective function of *word2vec* is to maximize the log probability of context word ($w_O$) given its input word ($w_I$), i.e. $log\ P(w_O|w_I)$. Its input is a word (i.e. $v_{w_I}$ is a vector of one word) and the output is a context word. For each

input word, the number of left or right context words to predict is defined by the window size hyperparameter.

Paragraph vectors, or *doc2vec*, were proposed by Le and Mikolov (2014) as a simple extension to *word2vec* to extend the learning of embeddings from words to word sequences. *doc2vec* is an unsupervised framework that learns continuous distributed vector representations for pieces of texts. The texts can be of variable-length, ranging from sentences to documents. The name *doc2vec* is to emphasize the fact that the method can be applied to variable-length pieces of texts, anything from a phrase or sentence to a large document.

### K-Means Clustering

In a general sense, k-means clustering is implemented by assigning data points to a cluster centroid, and then moving those cluster centroids to better fit the clusters themselves. To run an iteration of k-means on a dataset, k number of points are first randomly initialized to serve as cluster centroids. In this study, k=5 data points are selected, and the centroid is affixed in the same place as those points. Then each data point is assigned to its nearest cluster centroid. Finally, the cluster centroid is updated to be the mean value of the cluster. Since the performance of k-means depends on the initialization of the cluster centers, a suboptimal choice of initial seed, e.g., outliers or extremely close data points, can easily cause the algorithm to converge on less than globally optimal clusters. Therefore, the assignment and updating step are usually repeated, minimizing fitting error until the algorithm converges to a local optimum.

## 4. Results

### 4.1 Analysis and Interpretation

The algorithm used for dimensionality reduction in this study is t-distributed Stochastic Neighbor Embedding (t-SNE), which is also employed for visualizing high-dimensional data. The goal is to embed high-dimensional data in low dimensions in a way that respects similarities between data points representing movie review documents. Nearby points in the high-dimensional space correspond to nearby embedded low-dimensional points, and distant points in high-dimensional space correspond to distant embedded low-dimensional points. The scatterplot of t-SNE outputs generated from text processed by *word2vec* model is featured in Figure 1 shows that

most of words that have nonzero TF-IDF scores are clustered in the center, whereas words that have zero TF-IDF score are scattered around the edges. The scatterplot of t-SNE outputs generated from text processed by *doc2vec* model is featured in Figure 2, from which it is interesting to see that most of movie review documents tend to cluster together. There are only two outliers labeled as *YF_Doc4_Snowden* and *PP_Doc6_The_Jungle_Book* that are not very related to other movie review documents.

The results of two different k-means clustering methods applied are given in Table 1. The k-means clustering method using the TF-IDF matrix puts almost all the documents contributed by the same person into a single cluster, whereas the k-means clustering method using the *doc2vec* matrix is somewhat better at distributing the documents on movies of the same genre across the five clusters. While the TF-IDF matrix based k-means clustering method assigns the review of the historical drama movie *12 Years a Slave* to the same cluster in which reviews of thriller movies such as *Parasite* and *Shape of Water* are placed (these three movies are all uploaded by the same person), the *doc2ve* matrix based k-means clustering method correctly groups historical drama movies such as*12 Years a Slave*, *Hidden Figures*, *Argo* together in the same cluster.

On the other hand, an overview of top 50 most common words in each k-means cluster generated by using the TF-IDF matrix (Figure 3) suggests that most of common terms used for clustering movie review documents are still from the list of high-frequency terms across all documents. However, many non-informative but frequently occurring terms like “one”, “back”, and “seem” are unselected, whereas more content-related but less frequent words like “computer”, “war”, “comedy” are selected. Table 2 shows 20 words selected for my documents. Although some of the words such as “space”, “magical”, “slavery” are terms that are very specific to individual document, they can connect movies in the same genre. For instance, “space” is a term ubiquitous among sci-fi movies; “magical” indicates a common theme shared by many fantasy movies; “slavery” is a popular topic among historical drama movies. On the contrary, terms such as names of actors, characters, or directors that are so specific to individual document that they cannot help group documents so keeping them is pointless. Similarly, functional words such as “after”, “take”, “which” are so common that they do not help with clustering documents based on topics either because they are meaningless.

# Appendix

| Cluster | K-Means (TF-IDF matrix) | K-Means (*doc2vec* matrix) | Movie Genres |
|---|---|---|---|
| 1 | 14 | 11 | Science Fiction |
| 2 | 7 | 8 | Comedy, Drama |
| 3 | 8 | 12 | Fantasy, Adventure |
| 4 | 8 | 8 | Action, Thriller |
| 5 | 5 | 3 | Romance, Drama |

**Table 1**. *Comparison of k-means clustering methods using the TF-IDF matrix and the doc2vec matrix, respectively. The number of movies classified into each cluster representing distinct genres is shown in each column.*

| Term | TF-IDF Score | Cluster | Documents |
|---|---|---|---|
| enigma | 16.27221174 | 1 | KS_Doc1_2001 Space Odyssey.txt, KS_Doc2_Wargames.txt, KS_Doc3_The_Matrix.txt, KS_Doc4_I_Robot.txt, KS_Doc |
| war | 8.045275722 | | |
| murder | 7.325175654 | | |
| blood | 14.65035131 | 3 | PP_Doc1_Disneys_New_Mulan.txt, PP_Doc2_Cinderella_Review_Straight-Faced.txt, PP_Doc3_Aladdin_Review_This.t |
| slavery | 7.325175654 | 2 | SD_12 years a slave.docx, SD_Birdman.docx, SD_Green Book.docx, SD_Moonlight.docx, SD_Parasite.docx, SD_Shape |
| magical | 3.374905755 | 3 | PP_Doc1_Disneys_New_Mulan.txt, PP_Doc2_Cinderella_Review_Straight-Faced.txt, PP_Doc3_Aladdin_Review_This.t |
| romance | 3.374905755 | 5 | VPD_DOC1_AMELIE.txt, VPD_DOC2_EAT_PRAY_LOVE.txt, VPD_DOC5_THE_SECRET_LIFE_OF_WALTER_M |
| racism | 8.13610587 | | |
| wikileaks | 12.20415881 | 5 | VPD_DOC1_AMELIE.txt, VPD_DOC2_EAT_PRAY_LOVE.txt, VPD_DOC5_THE_SECRET_LIFE_OF_WALTER_M |
| computer | 7.325175654 | 1 | KS_Doc1_2001 Space Odyssey.txt, KS_Doc2_Wargames.txt, KS_Doc3_The_Matrix.txt, KS_Doc4_I_Robot.txt, KS_Doc |
| american | 4.552586932 | | |
| space | 2.969440646 | 1 | KS_Doc1_2001 Space Odyssey.txt, KS_Doc2_Wargames.txt, KS_Doc3_The_Matrix.txt, KS_Doc4_I_Robot.txt, KS_Doc |
| segregation | 4.068052935 | | |
| adventure | 2.969440646 | 5 | VPD_DOC1_AMELIE.txt, VPD_DOC2_EAT_PRAY_LOVE.txt, VPD_DOC5_THE_SECRET_LIFE_OF_WALTER_M |
| disappointing | 4.068052935 | | |
| rocket | 4.068052935 | | |
| animals | 6.749811509 | 3 | PP_Doc1_Disneys_New_Mulan.txt, PP_Doc2_Cinderella_Review_Straight-Faced.txt, PP_Doc3_Aladdin_Review_This.t |
| white | 6.828880398 | 1 | KS_Doc1_2001 Space Odyssey.txt, KS_Doc2_Wargames.txt, KS_Doc3_The_Matrix.txt, KS_Doc4_I_Robot.txt, KS_Doc |
| history | 2.681758574 | | |
| dramas | 4.068052935 | | |

**Table 2.** *Top 20 term selection for the corpus vocabulary based on TF-IDF scores.*

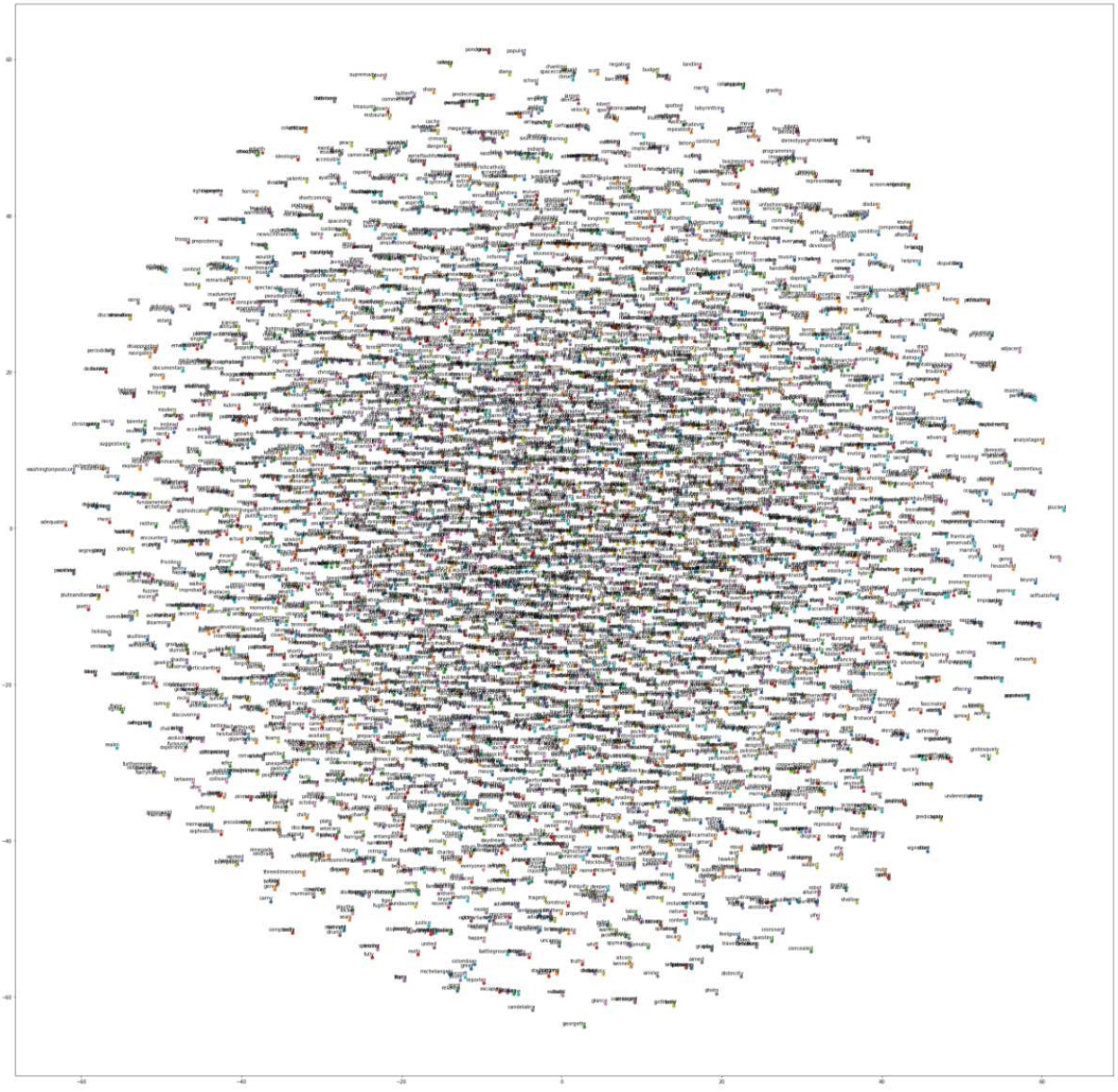

**Figure 1**. *Scatterplot of t-SNE algorithm outputs generated from text processed by word2vec model.*

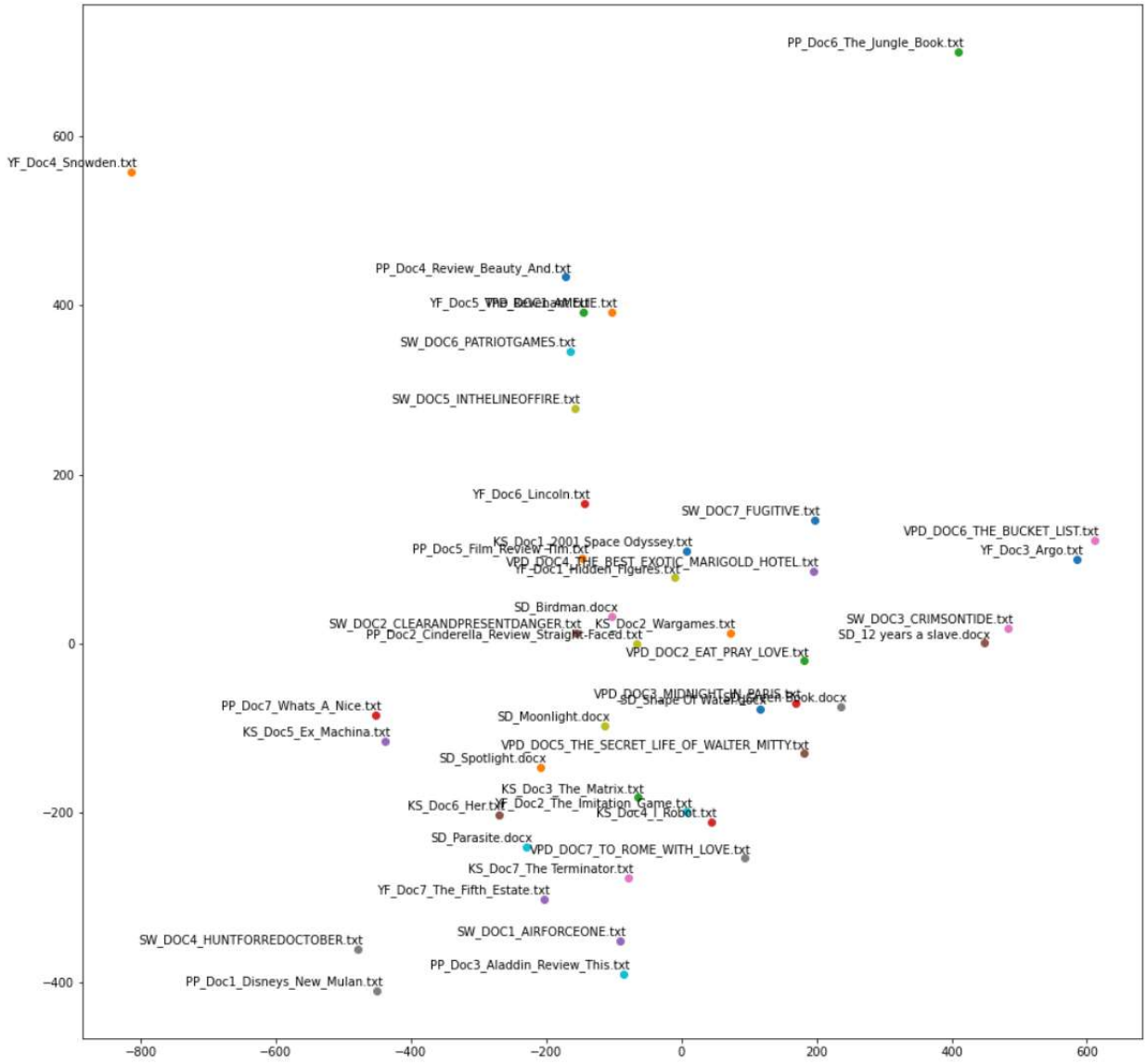


**Figure 2**. *Scatterplot of t-SNE algorithm outputs generated from text processed by doc2vec model.*

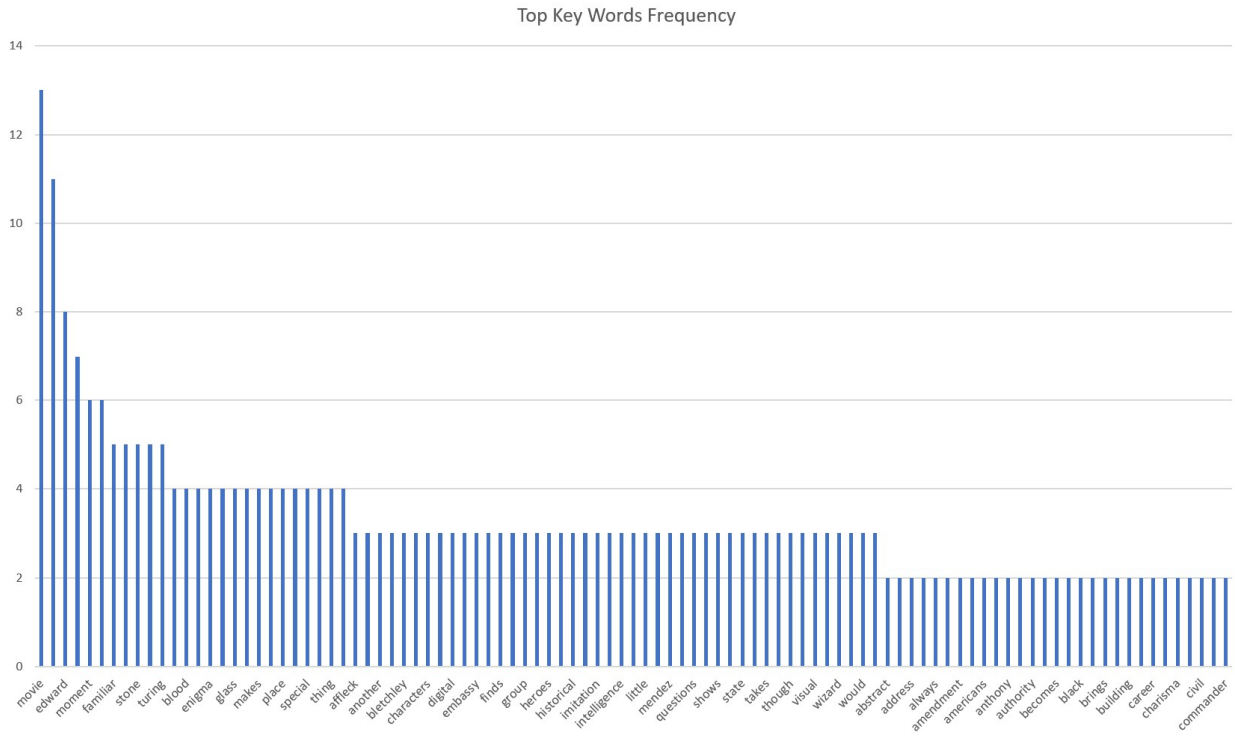


**Figure 3.** *Bar chart of top 50 most common words in k-means clusters generated by using the TF-IDF matrix.*

**Bibliography**

Büschken, J., & Allenby, G. M. (2016). Sentence-based text analysis for customer reviews. *Marketing Science, 35*(6), 953-975.

Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., & Smith, N. (2015). Sparse overcomplete word vector representations. *arXiv preprint arXiv:1506.02004*.

Jang, B., Kim, I., & Kim, J. W. (2019). Word2vec convolutional neural networks for classification of news articles and tweets. *PloS one, 14(8), e0220976*.

Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196). PMLR.

Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Yogatama, D., Faruqui, M., Dyer, C., & Smith, N. (2015, June). Learning word representations with hierarchical sparse coding. In *International Conference on Machine Learning* (pp. 87-96). PMLR.

Yow, F. K., & Tan, T. P. (2013, August). Broadcast News Story Clustering via Term and Sentence Matching. In *2013 International Conference on Asian Language Processing* (pp. 215-219). IEEE.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.