# Assignment 2

Yuan Zhou

**Abstract**

This research aims to build a good machine learning model to help us categorize new articles topics. We utilized three different algorithms to vectorize our documents and trained with the same Random Forest classifier. We collected and evaluated the performance metrics and concluded on the best vectorization approach.

## 1. Introduction

COVID-19 has attracted the attention of media from all industries. The amount of news, reports, and researches about it is overwhelming, and it is critical to well categorize the articles for further study. We wish to focus on the booming technology trends in IT, health, and business fields. Thus, we need the help of machine learning to achieve text classification. By experiment, we aim to find the best vectorization approach and generate a robust classifier to categorize the topics of collected articles in the future.

## 2. Literature Reviews

Most text classification algorithms require the input data as fixed-length, numerical feature vectors. Thus, how to generate the proper representation of data could determine the final performance result. The most common vector representation for texts is the bag-of-words or bag-of-n-grams conducted by Harris in 1954.[1] Its simplicity and efficiency soon won popularity. After that, a more precise algorithm TF-IDF for measure word importance was conducted[2]. In 2014, Tomas Mikolov, inspired by the previous work in learning vector representations of

words using neural networks, made his famous breakthrough Doc2Vec. [3]

## 3. Methodology

### 3.1 Assigning labels to documents in the corpus

We used a classification API provided by uclassify.com to assign labels to our corpus. The API has ten classes of topics: *Arts, Business, Computers, Games, Health, Home, Recreation, Science, Society*, and *Sports*; it could predict the probabilities of each class given one document. We used the highest probable topic to label our document. Besides, we have merged all topics that are not *Business, Computers,* or *Health* into the category *Other*. Thus, we got a corpus where documents are classified into four classes: *Computers*, *Health*, *Business*, and *Other*.

### 3.2 Vectorization of documents

In this research, we applied three approaches to vectorization: analyst judgment, TF-IDF, and Doc2vec.

The analyst judgment we did in detail is of 3 steps. First, each document is converted to a dictionary, where the keys are unique words, and the values are the count of occurrence for the terms, and we have a list of such dictionaries for the corpus. Second, similarly to the first step, we generated a dictionary of keywords appearance counts in the whole corpus. Then we extract the top X number of keywords that have the most occurrence and vectorize each document based on top words' occurrence.

TF-IDF stands for term frequency-inverse document frequency. Its calculation could be represented as:

1. *TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).*
2. *IDF(t) = log_e(Total number of documents / Number of documents with term t in it).*
3. *TF-IDF(t) = TF(t) * IDF(t)*

TF-IDF vectorization calculates terms' tf-idf values, extracts the X of them with the highest values, and then uses these terms to convert docs to vectors of float numbers.

Doc2vec method was established on top of Word2vec, a numeric representation for each word that can capture the relations between words. Unlike TF-IDF, the vector representation generated by Doc2vec is not associated with the terms in the document but from the underlying model's weights and embeddings.

#### 3.3 Classifier training and test

With numeric vectors converted by each approach for the training/testing dataset, we trained the RandomForest classifiers with the same hyperparameters to get the performance metrics, such as the F-1 score, to evaluate the performance of models fed with different vectors. The variance between performance metrics could reveal the best vectorization approach for text data. We also experimented to find the impact of dimensions on the models' performance by training with 50, 100, 200, 500, 700, and 1000 dimensions.

## 4. Results & Evaluation

#### 4.1 Vectorization Terms comparison

We compared the terms picked for vectorization by Analyst Judgment and TF-IDF. According to Appx-Table-1, both Analyst Judgment and TF-IDF share some terms, such as *covid*, *compani*, *verg*, *share*, *tech*, *view*, and *video*; TF-IDF has excluded some frequent but ambiguous words, such as *like*, and included some specific terms, such as *trump*. Another finding from the table that is worth noting is that as the vectors' dimension increases, the top terms from TF-IDF become less similar to the most frequent words selected by Analyst Judgment.

#### 4.2 Corpus Reviews

From Appx-Table-2, about 74% of documents fall into the categories we want to research, which is an acceptable rate. We could improve the corpus' quality by collecting more articles, and by utilizing the classification model, we trained to filter out the documents in the 'Other' class.

#### 4.3 Classification models performance comparison

In Appx-Table-3, we recorded the F-1 score of the models trained by different vectorization approaches and vector dimensions. From the table we can tell:

- When the vector dimension is the same, the performance comparison is: *Doc2vec* > *Analyst Judgment* >= *TF-IDF*. As it has the highest F1-score in every column, we believe that Doc2vec is the most efficient vectorization approach among the candidates.
- From 50 to 500, increasing dimension improved the performance of models, and the opposite happens after dimension exceeds 500. Our theory to explain the result is that there must be a sweet spot of dimension to include most differentiating terms while excluding useless ones.

## 5. Conclusion

From our experiment and analysis, Doc2vec is a superior text vectorization approach than TF-IDF and ad-hoc analyst judgment in general. We have also generated a classification model that has about an F-1 score of 0.70. Such a result proved the feasibility of using the current corpus to classify articles topics. The model developed in this research could help us filter out the irrelevant document more efficiently by applying the topic class prediction.

## References

[1] Harris, Zellig S. "Distributional structure." Word 10, no. 2-3(1954): 146-162.

[2] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." In Proceedings of the first instructional conference on machine learning, vol. 242, pp. 133-142. 2003.

[3] Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." In International conference on machine learning, pp. 1188-1196. 2014.

## Appendix

| Rank \ Vectorization | Analyst Judgment | TF-IDF (50) | TF-IDF (100) | TF-IDF (200) | TF-IDF (500) | TF-IDF (1000) |
|---|---|---|---|---|---|---|
| 1 | covid | verg | verg | verg | verg | verg |
| 2 | new | share | share | share | view | view |
| 3 | share | view | compani | view | share | share |
| 4 | verg | compani | view | compani | trump | trump |
| 5 | tech | new | new | trump | save | save |
| 6 | compani | covid | million | save | video | appl |
| 7 | view | million | covid | new | compani | video |
| 8 | video | tech | tech | million | appl | million |
| 9 | like | said | said | covid | million | guid |
| 10 | one | save | save | video | covid | visit |

Appx-Table-1 Top 10 terms by vectorizations approaches

| Class | Docs Count |
|---|---|
| Computers | 166 |
| Health | 75 |
| Business | 58 |
| Other | 103 |

Appx-Table-2 distribution of docs per topic class

| **Model \ Dimension** | 50 | 100 | 200 | 500 | 700 | 1000 |
|---|---|---|---|---|---|---|
| Analyst Judgment | 0.586 | 0.582 | 0.637 | 0.64 | 0.615 | 0.601 |
| TF-IDF | 0.548 | 0.554 | 0.594 | 0.611 | 0.635 | 0.602 |
| Doc2vec | 0.624 | 0.663 | 0.655 | 0.699 | 0.673 | 0.638 |

Appx-Table-3 F-1 score of models trained with different dimension