Capstone Final Report

Hongting Zhu

1. Introduction

ProQuest is an academic content aggregator as well as a research and learning hub, and ProQuest Dialog has a specialty as ProQuest's powerful search interface that provides access to numerous Pharmaceutical and Biomedical collections to allow for advanced research and pharmacovigilance. Students and professionals who study medicine rely on it for literature reviews. Pharmaceutical companies use it to support regulatory compliance, where they routinely check for any scientific findings on their products from the academy to stay alert of updates. However, the current search platform requires arduous work on inputting search queries. The search engine works only with professionally composed queries that match the indexing of the database, and due to the outdated information retrieval algorithm, the search results are less accurate as the research topics get more complicated, and cannot respond with up-to-date findings as the database collects more and more documents with more than 2 millions of records.

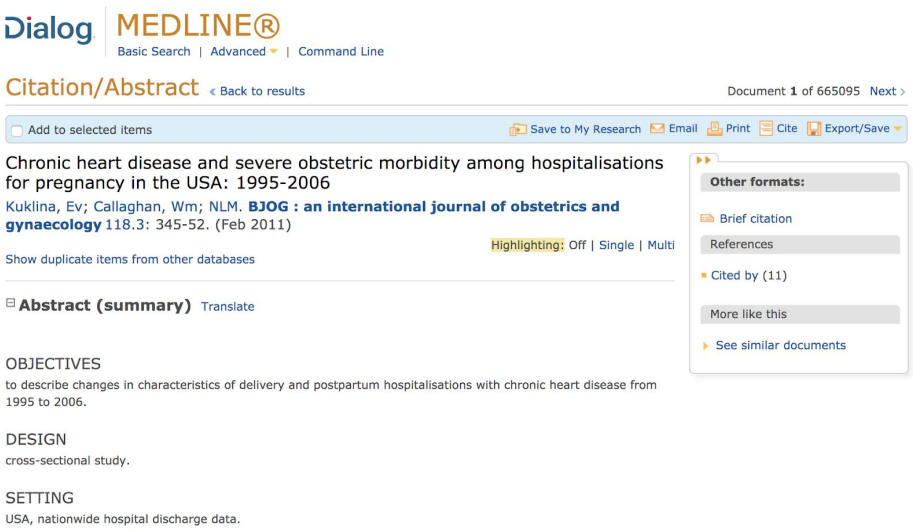


Fig 1. Sample Search Result on ProQuest Dialog Interface

Therefore, the goal is to simplify and improve the searching experience while providing powerful tools to generate precisely and encompassing results. Intelligent computing technology has been adapted to more and more products in the industry, ranging from

hardware to software. As machine learning models gain popularity in business, the integration of Natural Language Processing (NLP) into text-based applications also has much more professional interests. The pioneer in the search engine industry, Google, has upgraded its search engine to Bidirectional Encoder Representations from Transformers (BERT) and made great breakthroughs in interpreting queries and displaying better responses. [1] Results from Google prove the feasibility of an upgrade in document retrieval technique from the traditional method. This is good news, as the language models mentioned in their upgrade are easy to convert and apply to the ProQuest Dialog platform, of which a screenshot is shown in figure 1. However, the ProQuest search engine itself uses a search technique that depends on the infrastructure of the database. To change the search method would be a time-consuming task, hence the limitation on time for the team could not guarantee a meaningful outcome. It is more feasible to focus on improving search assistance functionalities that leverage the power of AI to help users search faster and more precise on pharmaceutical and biomedical collections.

We hence aim to learn from this interface, and improve upon it, i.e. make modular changes on parts of the interface. As the screenshot shows, after the user searches for a query and retrieves a useful result (correct label) in hand, the platform displays the abstract, and its relevant information including suggestions on similar documents, which is displayed on the right sidebar. Currently, the results give flawed and irrelevant results by human evaluation. The team decides to work on improving the document similarity, on training a neural net model called Doc2Vec [2] on a sample of ProQuest's corpus and then extends to the entire corpus that has 2 million records. We then integrate this model into a prototype that can run queries, retrieve document information, and show and compare similar documents. Due to the nature of this project, which uses unsupervised learning and is large in the size of the training dataset, we do not have the resources to leverage manpower to evaluate the training results. Therefore, to evaluate the performance of our models, we use several novel evaluation methods to cluster the vectorized documents and review its result. The methods include textual coherence and grant-to-linkage precision-recall, as there are no universally agreed methods to evaluate text clustering quality. We expect the Doc2Vec model to perform better in textual

coherence [3] than a baseline of the current algorithm, which we approximate by TF-IDF[1] upon the consent of ProQuest. The other evaluation methods, grant-to-linkage precision-recall should also provide a similar result.

As an MDP project, the team is expected to deliver an end product, which is a prototype that suggests similar documents using the new algorithm better than ProQuest Dialog's existing document similarity mechanism. Another solution is also embedded in ProQuest Dialog systems. The other team works on an automated tagging system in the corpus and since I am not involved, the other subteam will not be mentioned here.

2. Document Similarity

2.1 Document Vectorization

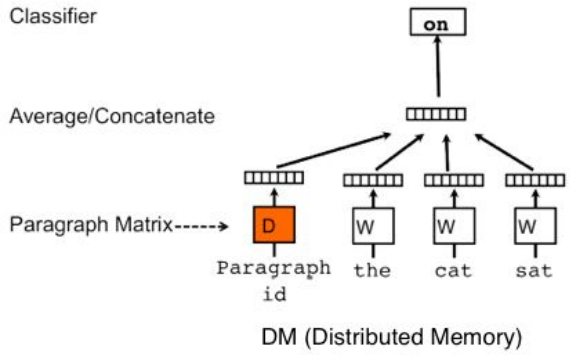


Fig. 2. A framework for learning paragraph vectors.

One of the major difficulties is the complexity of medical terms within the search engine embedded in the platform. The users need to carefully construct search queries in correct biomedical terms to retrieve the "true" documents. Our new search engine aims at giving more insights and hands the user more meaningful information based on the search result after the user clicks on one correct document to view. With one correct result, the system retrieves more relevant documents for the user based on the inter-document similarity.

---

1 Term frequency-inverse document frequency: a numerical statistic that determines how important a word is to a document in a collection or corpus.

Semantic similarity is a great indicator for the relationship between two concepts, i.e. textual items, as suggested by Le Q., and Mikolov T. [2]. The authors suggested an algorithm, Doc2Vec, to convert the representation of different texts to a fixed-length vector space model. Fig 2 explains how it is done. The traditional word vector predicts a word given the context of other words. Every word is mapped to a learned, unique vector and represented in a column in the matrix, indexed by its position in the vocabulary dictionary. Based on the basic framework, an additional paragraph token is added via a vector matrix D. The concatenation or average of the four vectors gives prediction to the fourth word. Here, the paragraph token fills out the missing information in the document context and can be seen as a kept memory, a representation of its topic or content. This is called the standard paragraph vector with distributed memory [2] (PV-DM) and works well for most of the tasks.

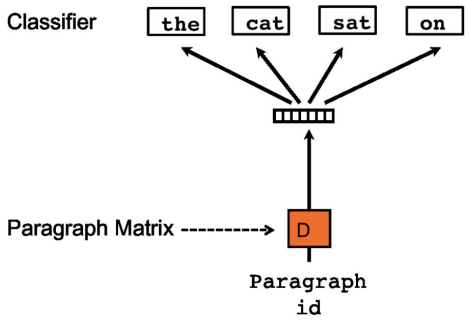


Fig. 3. the paragraph vector is trained to predict the words in a small window

Another way is shown in Fig 3. The context is ignored and the paragraph vector is forced to predict a word in a randomly sampled window. For example, in the context of the given paragraph, the model needs to fill out the sampled window with any single given word in the bag of words, "the," "cat," "sat," and "on". It is called the Distributed Bag of Words version of Paragraph Vector (PV-DBOW). Evidence shows promising outcomes with the two combined for Doc2Vec tasks.

The state-of-art model performs well on sentimental analysis tasks, which is essentially proof of its applicability into our system, where we aim at analyzing difficult academic

texts and find the inter-associations based on sentiment. As Doc2Vec is a well-developed model in a framework, our project takes advantage of the previously hard work of researchers and focuses on fine-tuning the parameters to suit our case.

3. Methods

3.1 Dataset Collection

The sponsor company supplies the team with a database of 2 million records of PubMed documents. Some of the records turn out to be inputted incorrectly and give an erroneous reading from the given file. Some of them are too short to be meaningful, i.e. only a single word, or contain only irrelevant information such as grant providers, personal contact information, or website links. Those can cause potential problems to achieve good performance on model fitting. We deem those as outliers, and cleanse, classify and build a collection of applicable and retrievable documents from the ProQuest's corpus. After the cleanup, a few dictionaries are extracted for easy and fast access to relevant training data, which are 1) a map from PubMed ids to its title and abstracts, 2) a map from PubMed ids to its grant ids. The titles and abstracts were lowered, tokenized, and concatenated to compose the dictionary.

Due to the nature of the project being large, we provide evidence of success and demonstrate a proof-of-concept for algorithms on a sample of the corpus and later apply the method to the entire corpus. A guideline for future data processing is also written in detail in the handoff document.

3.2 Training Procedure

Dov2Vec is an established model in gensim, an open-sourced library for NLP and unsupervised topic learning. It uses hierarchical softmax and the tunable parameters include corpus, epochs. We refer to Dynomant E et.al [4] for inspiration on parameter tuning. From the Doc2Vec model from gensim, we have chosen three of the available modifiable parameters for optimization of the model performance. The window size decides how large a sliding window, a.k.a. the randomly chosen window, is used to parse texts. The dm parameter sets the training on either PV-DM (dm=0) or PV-DBOW

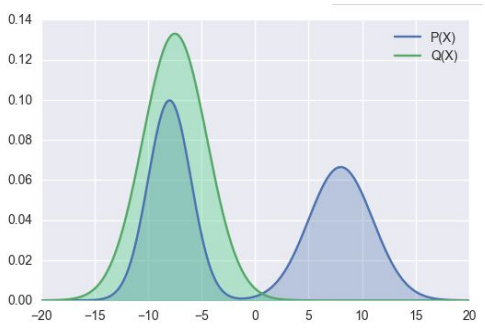
(dm=1). The vector size changes the number of dimensions of the output vector. Other common modifiable parameters we adjust include epochs and min_count. The min_count sets the threshold of the highest frequency for a word to be ignored, any word with a frequency higher than which will be put in the vocabulary. The epochs determine the iterations our training procedure will go over the input corpus.

The final models are trained on an AWS EC2 instance provided by the sponsors. The detailed infrastructure of the server is not revealed but we trained on roughly 128 threads and 512GB of RAM. We do runs on the sample set of the corpus, 15k documents. The parameters are then tested on 77k documents, and extend and apply to the entire corpus of 2 million documents. Total training time goes from 30 minutes to up to 6 hours.

3.3 Quantitative Evaluation

Since we are dealing with a large corpus, we decide to use evaluation algorithms for automated calculation rather than expensive and time-consuming manual evaluation. Several algorithms are introduced for evaluating the performance of our Doc2Vec model. To assess the quality of the document vectors, we cluster the output embeddings together based on the vector distance and form similarity clusters with k-means clustering. We then evaluate the similarity between documents within a cluster by calculating the textual coherence from the textual perspective and grant-to-article linkage, from the grant perspective, i.e., we want to see how similar the words in documents are, and how many common grants the documents share.

3.3.1 Textual coherence



$$\mathrm{JS}(p \parallel q) = \frac{1}{2}\mathrm{KL}(p \parallel m) + \frac{1}{2}\mathrm{KL}(q \parallel m)$$

where

$$m(x) = \frac{1}{2}\left(p(x) + q(x)\right)$$

$$\mathrm{KL}(p \parallel q) = -\sum_{x} p(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Fig. 4. brief explanation of JSD

Textual coherence gives a precise evaluation of how similar the texts are in a cluster. We leverage Jensen–Shannon divergence (JSD) to evaluate the similarity between two probability distributions. JSD is a symmetrized and smoothed version of the Kullback–Leibler divergence (KLD), and the formula for JSD is shown in Fig 4. given two distributions P and Q.

Before we apply JSD to our clusters, we first extract the count vector (one-hot encoding) of the top 20k words in the corpus, normalize the vectors to serve as “word probability vectors”.

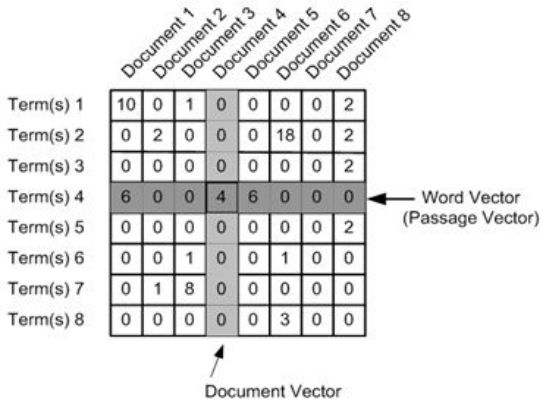


Fig 4. Count Vector

For example in Fig. 5., say these 8 documents are in a single cluster, we normalize each column so that they sum up to 1 and compare the term's "probability" distribution. Instead of 1 term, or unigram, represented as the row, we can use any n-grams as well. Then with the normalized probability vectors, we can calculate the average JSD of a cluster by comparing the probabilities vectors of all document pairs in the cluster. We make use of available mathematical models from scipy and rely on the existing implementation of JSD. Once we have the average JSD of a cluster, we must normalize the divergence since it naturally increases with cluster size. To do this, we sample a "random cluster" within the cluster size range of the solution, and subtract the average JSD of the chosen cluster for normalization. Then we can average the normalized JSD from every cluster together to get the final coherence score for the corpus. For the baseline, TF-IDF and the new method, compare their coherence score and having a higher score than TF-IDF means it outperforms the current ProQuest's algorithm.

3.3.2 Grant-to-article linkage

Grant-to-article linkage (G2A) measures the concentration of NIH grants within clusters. It is independent of the textual information and based on the fact that NIH grants sponsor research and experiments on similar topics. We first calculate a precision Pr and a cumulative recall score R while going through each cluster, and calculate a score s for each cluster. We then aggregate over the clusters to generate a precision-recall curve and then calculate the Herfindahl-Hirschman index (HHI) by taking the sum of the square of s. The variables to collect for calculation are as follows. An example of the calculation process is shown in table 1.

- Art – number of articles in the cluster
- ArtL – number of articles in clusters linked to grants
- Links – number of unique links to the ArtL
- Frac – ArtL/Art
- Sum* – cumulative sums
- R – recall = SumLink/TotLink
- Pr – precision = SumArtL/SumArt
- s – unique linkage present in each cluster

Cluster	Art	ArtL	Links	Frac	SumArt	SumArtL	SumLink	R	Pr
1	100	90	150	0.90	100	90	150	0.075	0.900
2	100	80	130	0.80	200	170	280	0.140	0.850
3	100	70	120	0.70	300	240	400	0.200	0.800

Table 1 by Boyack KW et. al [5]. "Example of cumulative precision-recall calculation based on grant-to-article linkages. Assume that the total number of linkages (TotLink) available is 2000."

4. Results

We determine that the model performs the best when we set feature vector size to be 300, epochs to be 15, min_count to be 20, and window size to be 9. We trained with a distributed bag of words (PV-DBOW).

4.1 Evaluation Analysis

4.1.1 Textual Coherence

We learn the numerical embedding vector for each article using fine-tuned Deep-learning based methods Doc2Ve and TF-IDF. The output embedding is an N vector with the same length as the training feature vector. The embeddings are clustered using K-Means and find an optimal number of clusters, which is 125 determined by the classical elbow method [7]. We then convert the corpus to a matrix of word counts using unigram/bigram/trigram with a pre-set maximum token, 20000. Summing up the averaged JSD of all 125 clusters, we have a textual coherence score for the old and new methods shown in table 2.

Embedding | ngram	Unigram	Bigram	Trigram
Doc2Vec	2.18	**2.09**	**0.26**
TF-IDF	3.91	1.53	0.24

Table 2. Result from JSD with unigram, bigram, and trigram count vectors.

As we can see, both the textual coherence score from bigram and trigram count vector for Doc2Vec is higher than TF-IDF. We should consider the fact that this measure is biased towards TF-IDF because TF-IDF inherently focuses on the singular matching words instead of semantic analysis. Therefore it is promising to see Doc2Vec outperform TF-IDF by 36.6% and 8.3% on the similarity of longer phrases within document clusters.

4.1.2 Grant-to-article Linkage

Using the clustering solution based on Doc2Vec embeddings, and one based on TF-IDF embeddings, we calculate the grant-to-article linkage (G2A) and get a precision-recall curve as shown in Fig. 5. while going through each cluster.

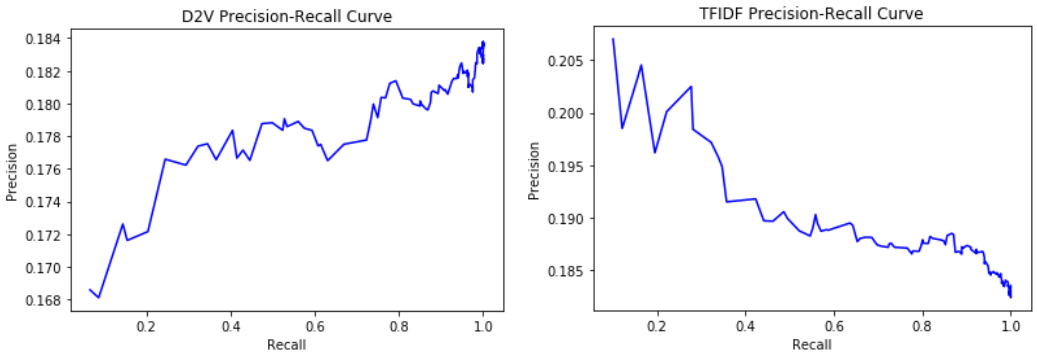


Fig 5. Comparison between the precision-recall curve for Doc2Vec(left) and TF-IDF(right)

As shown in the precision-recall curve, while TF-IDF gets a high precision in the beginning, the score drops down fast and eventually hits 0.184 as Doc2Vec goes up and reaches its ends, which is 0.184 as well. It is not as informative as the Herfindahl-Hirschman index (HHI) as there is not a noticeable difference in the precision in the end and neither of them converges. However, Doc2Vec scores 2373.41 for HHI while it is 2236.67 for TF-IDF. It is a significant 6.1% improvement in the concentration of grants in Doc2Vec embedded document clusters. Doc2Vec outperforms TF-IDF in clustering similar documents from the same grant better.

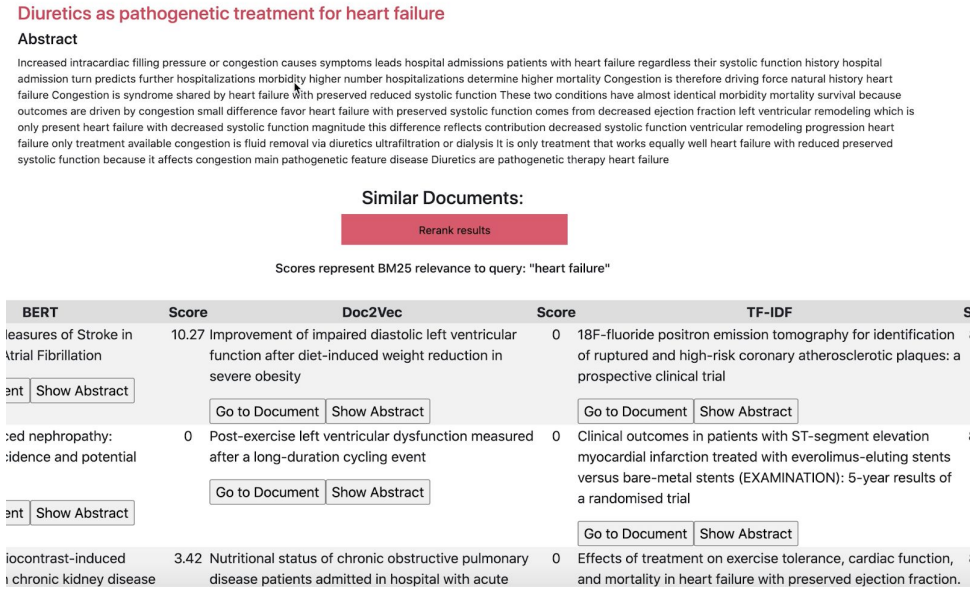


Fig 6. Screenshot of the prototype interface

4.2 Prototype on AWS

As a requested stretch goal by MDP, we implement a working interface host on the AWS server to show our training results. Fig. 6. showcases the end product, where the abstract of the article is displayed and followed by a list of similar document suggestions by Doc2Vec, TI-IDF, and BERT. BERT is added at a late stage for the mere purpose of presentation. The prototype presents a way of integrating the new model into the existing ProQuest Dialog platform. When the user searches for a document and goes into one of the documents he wants, the system automatically retrieves similar documents for the user to view and allows the user to see the abstract on the same page, and that could save a lot of time wasted on going back and sifting through the search results.

5. Conclusion

ProQuest Dialog is a powerful search engine for pharmaceutical and biomedical papers. But the document retrieval algorithm is getting outdated in current days. In this paper, we find a way to improve similar document suggestions on the Dialog interface. The NLP model Doc2Vec PV-DBOW embeds and clusters the similar documents together, and both evaluation methods return a better score for the baseline TF-IDF method, with textual coherence being 36.6% higher on bigram count vectors, 8.3% higher on trigram count vectors, and grant-to-article linkage being 6.1% higher on Herfindahl-Hirschman index. More investigation should be carried out to prove the irreplaceable effectiveness of Doc2Vec between other NLP embedding methods including BERT and PM25. Manual evaluation could also be interesting to implement as a gold-standard for future evaluation on the quality of the models.

6. Reference

1. Lan Z., Chen M., and Goodman S., et al. ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. https://arxiv.org/abs/1909.11942. Accessed: 19 October 2020
2. Le Q., and Mikolov T. Distributed Representations of Sentences and Documents. https://arxiv.org/abs/1405.4053. Accessed: 19 October 2020.

3. Boyack KW, Newman D, Duhon RJ, Klavans R, Patek M, et al. (2011) Clustering More than Two Million Biomedical Publications: Comparing the Accuracies of Nine Text-Based Similarity Approaches. PLOS ONE 6(3): e18029. https://doi.org/10.1371/journal.pone.0018029 Accessed: 18 December 2020.
4. Dynomant E., Darmoni S. J., Lejeune É., Kerdelhué G., Leroy J., Lequertier V., Canu S., & Grosjean J. (2019). Doc2Vec on the PubMed corpus: study of a new approach to generate related articles. https://arxiv.org/abs/1911.11698 Accessed: 18 December 2020.
5. Boyack KW, Newman D, Duhon RJ, Klavans R, Patek M, et al. (2011) Clustering More than Two Million Biomedical Publications: Comparing the Accuracies of Nine Text-Based Similarity Approaches. PLOS ONE 6(3): e18029. https://doi.org/10.1371/journal.pone.0018029 Accessed: 18 December 2020.
6. Elbow method (clustering). (2020). Retrieved 18, December, 2020, from https://en.wikipedia.org/wiki/Elbow_method_(clustering).