Virtual Health Medical Journals Search & Recommender System

Subramanian Padmanabhan,
College of Business & Economics, MSBA
California State University, East Bay
spadmanabhan3@horizon.csueastbay.edu

***Abstract*—An outbreak of respiratory disease caused by a novel coronavirus has harmed human lives. *Medical Professionals* use journals to get insights about how viruses behave and tackle pandemics. This requires reading *articles* & extracting important topics.**

This project utilizes Text Mining & Machine Learning Techniques to recommend Articles Textually, Contextually & based on Topics similarity for the Medical Research using abstracts from the *Virtual Health Library* database.

I. Project Overview and Problem Statement

A. *Background Information*

The economic and social disruption caused by the pandemics has changed human habits, decimated jobs & increased poverty. Medical Researchers need to provide pathway to control the crisis and overcome a pandemic, and finally be prepared to oversee future outbreaks.

One of the best options to achieve this is referring the pool of medical research articles of best minds in the respective fields, available all over internet.

B. *Project Overview*

This project aims to ease access to those topics and articles of a related journal. The project uses the Virtual Health Library as a source data and using different natural language processing algorithms, it recommends the best possible articles based on context, keywords, text, and similarity**.**

C. *Purpose & Problem Solving*

The primary purpose is to build a web application platform for article recommendation that will easily help medical researchers to search for articles. The results of the search, either with a keyword or a journal will, allow user to choose from the Articles based on Categorization of Results

II. Strategy and Execution Plan

A. *Strategy*

Several methods are to be tried to find an effective topic modelling algorithm. First step is to fetch the relevant medical journals, clean them using text mining frameworks and understand the word placement using graphs.

Latent Dirichlet Allocation (LDA) [1] models, Latent Semantic Indexing [25] and Bert-Nli [8] models are used for topic modelling [2] to extract hidden topics. This is done as performance and response time vary for each.

B. *Potential Impact of Solving Problem*

Project aims to solve and ease the understanding of diverse set of medical journals that has different data for researchers. Topic modelling of medical journals will allow others to look through multiple topics and organize, understand, and summarize them.

The outcome of the project enables more consistent interchange of ideas among medical teams leading to powerful decisions to manage medical problems in future.

III. Data Quality & Preprocessing

Total 3623 abstracts are used to train the model, with each abstract in a text format with collection of sentences. Each abstract is then broken down into words for the natural language processing models. However, to avoid bias and overfitting, we randomly shuffle them before running on the model.

. The data set covers a range of topics in medical world and is robust enough to be recommend as article for a search request. It is crucial to select the amount of dataset to avoid model crash. Since, genism models takes time to execute such vast data, 3623 random abstracts are selected.

A. Data Cleaning

To perform a clear analysis, it is essential to clean up the unstructured data using different methods, helping us to increase the accuracy of the prediction, and to reach reasonable inferences.

It was observed that some abstracts had headings that end with colon. These headers did not carry any significance and hence removed these headings within each abstract. Some abstracts also were in a language other than English and same was removed to make sure the data was stable and relevant.

After preparing a list of abstracts and shuffling the same, using combination of regular expressions and other data processing function, a header clean abstract data frame in language: English was created for further processing.

Furthermore, filtering stopwords [6] was essential as it removes commonly used words that do not add much information to the text. Using the corpus package, the list of stop words is removed by checking that list against words within the text.

Using the stemming method [14] from the Stem package, these word stems are created for each unique word in from all abstracts. Also, URLs are the address locators for online resources but are unnecessary while performing any text analysis. They are removed by using a regular expression [15] that replaces URLs with nothing (‘’).

Characters like full stop and comma do not add any value to the model building. These characters are removed by checking a binary method.

A few of the least familiar words from each document were removed using NLTK’S *Frequency Distribution* [7] function because they may not find a match in other documents and thus, we could build an efficient model that used abstract details.

B. Word Tokenize

Once the abstracts are cleaned, the words in the text are separately obtained using the *word tokenize* [16] method from the NLTK Tokenize package.

These inflected words need to be reduced to their root forms. These words may not be a valid word but assist in achieving a more precise understanding of the text.

C. *Corpora Dictionary*

A corpora dictionary [21] is created by passing these tokenized words to a function. A corpus contains each word's token's id, to work on text documents. *Genism* model requires the words, i.e., tokens to be converted to their unique ids.

Here the list of words is passed to the object [17] from Genism Library where each unique word is mapped to its unique integer id.

D. *Document Term Matrix*

The tokenized list of words is sent to a bag of words object [18] and it results in frequency of each word, thus creating a Document Term Matrix for the complete bag of words. Each unique word's occurrence in a document is displayed as a matrix and then fed to build a topic prediction model. We can say that this matrix is input to the model.

IV. Model Implementation

Latent Dirichlet Allocation (LDA) is a popular algorithm for topic modeling with excellent implementations in the Python's Genism package. This module is very scalable, robust, and optimized in terms of performance.

A. LDA Multicore

This module allows both LDA model estimation from a training corpus and inference of topic distribution on new, unseen documents. LDA Multicore [1] provides a faster implementation, especially for exceptionally large corpora as in this case.

Using all CPU cores to parallelize and speed up model training, the model performance and response time was a key to extracting information from these huge texts. Following are the parameters that are used:

- *corpus* – cleaned corpus of abstracts obtained from data preprocessing.
- *num_topics* – The number of randomly requested latent topics to be extracted from the training corpus.
- *chunk size* – Number of documents to be used in each training chunk.
- passes – Number of passes/epochs through the corpus during training.
- *id2word* – This is a dictionary mapping from word IDs to words. It is used to determine the vocabulary size as well as for debugging and topic printing [17].

B. Coherence Model

Topic coherence measurement is widely used metric to evaluate topic models. Coherence model from *Gensim* library [5] is used to compute the coherence values for each number of topics. Similarly, log perplexity score [22] is calculated using LDA model functionality.

The high value of topic coherence score model [12] will be considered a good topic model. At each step, the coherence score is calculated to give the value for each topic. The topic with highest score would then be used in the model to find the topic distribution.

Following are the parameters that are used:

- model – pre trained topic model is provided.
- texts – tokenized cleaned texts.
- dictionary – dictionary mapping of id word.
- coherence – coherence measure to be used. In this case it is 'c_v'.

In this phase, evaluating the LDA multicore model with 1000 epochs was done to evaluate the model performance. The model that performed optimally was used for further data analysis. After training model using corpora dictionary, document term matrix, number of epochs and topics, it is saved for further analysis.

C. Latent Semantic Indexing (LSI) Model

LSI is based on the principle that words that are used in the same contexts tend to have similar meanings. A key feature of LSI is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts.

LSI helps overcome synonymy by increasing recall, one of the most problematic constraints of Boolean keyword queries [23]. Synonymy is often the cause of mismatches in the vocabulary used by the authors of documents and the users of information retrieval systems.

As a result, Boolean or keyword queries often return irrelevant results and miss information that is relevant. Thus, semantic relatedness of text is preferred over the keyword search. In practice, LSI is faster to train than LDA.

- *Document term matrix developed from cleaned corpus*
- *num_topics* – The number of randomly requested latent topics to be extracted from the training corpus.
- *chunksize* – Number of documents to be used in each training chunk.
- *power_iters* – Number of passes/epochs during training the model.
- *id2word* – This is a dictionary mapping from word IDs to words. It is used to determine the vocabulary size as well as for debugging and topic printing [17].
- *extra_samples* – extra samples to use besides number of topics to improve accuracy.

Once the model is trained, new text article is first converted into a bag of words dictionary vector and passed to the lsi model. Now, similarities of new text article vector are obtained against all the 3623 indexed documents while training the model.

D. Bert NLI

For better accuracy and model predictions, bert-based-nli pre-trained model [8] is used to train the corpus. This is a sentence-transformers model. It maps sentences & paragraphs to a 768-dimensional dense vector space called sentence embeddings and can be used for tasks like clustering or semantic search.

This is a type of Hugging Face transformers [24] typically used for natural language processing. Input for this model is the list of abstracts that are cleaned during the data preprocessing. All the 3623 text abstracts are converted to sentence embedding using this sentence transformer.

To find the cosine similarity, the sentence embeddings of training data and new test articles are passed to *cosine similarity* library [29]. The higher the similarity, better is article recommendation.

V. Model Evaluation

Three Models are built to recommend journals with respect to the User search. The search items could be a keyword, a single sentence, or a big article.

If the search is a keyword, then only the Bert Model's (Based on Textual similarity) and LSI model's (Based on Contextual Similarity) results are displayed.

As LDA model uses a set of words and distributes it to form topics, it expects a string of sentences as input. When the user search is an article or sentence of big length them all 3 models present their topo most articles with biggest cosine similarity or matrix score, respectively.

A. Benchmark Models & Comparison

Recommending articles or responding to a search is a common technique. Two of the big technology companies are *AWS [38]* & Google who do this work efficiently.

Amazon uses SageMaker LDA algorithm uses tensor spectral decomposition [26] where model parameters are first found then topics for each document are calculated. *It* is a highly trained model.

Following is the working principle of *SageMaker LDA*

1. After creating the *SageMaker LDA* session, instances and setting up hyperparameters the model is trained for the input corpus.
2. After model training, it is deployed on the new article to extract topic distributions corresponding to each of the input documents.
3. This gives the probability distribution list across all the selected number of topics for the new test article

Google Search engine returns results where the keyword or synonym of the search query is present. The results are most of the time beneficial and widely used by every Internet user to expand on their research.

However, the user must open each result, read, and then decide as to which article is relevant. Also, in case we put a huge article with many sentences and search on Google, sometimes it does not return a result.

Compared to these 2 examples, the project here simplifies user search by recommending articles and giving indications of the same, such as:

1. Article Similarity score to given input search
2. Article's main keywords
3. Article Summarization
4. Article Categorization based on Text, Context & Topic match

B. Model Building & Behavior

a. The Bert Model uses sentence transformers library & Cosine similarity to find vector relationship of words. This is a pretrained model trained on huge datasets based on Transformer (Encoder & Decoder Technique) [39].

b. The LSI model works using matrix similarity of words to recommend article. Here, we develop a large matrix of term-document (document-term-matrix) association data and construct a "semantic" space wherein terms and documents that are strongly associated are placed near one another.

Singular-value decomposition (SVD) allows the arrangement of the space to reflect the major associative patterns in the data, and ignore the smaller, less important influences [31].

As a result, terms that did not actually appear in a document may still end up close to the document if that is consistent with the major patterns of association in the data [9].

Position in the space then serves as the new semantic indexing. Retrieval proceeds by using the terms in a query to identify a point in the space, and documents in its neighborhood are returned to the user. Below is the mathematical matrix representation of the same,

c. The LDA model uses number of topics to suggest an article. The number of topics used to build actual model corresponds to higher coherence score. Similarly, perplexity score should be lower for more optimized model. Cosine similarity is later used to determine the similarity between abstracts and suggest the article.

C. Model Results Interpretation

The project, here, goes a step further and allows user to make his decision quicker and makes it flexible for user to choose the recommended articles based on categorizations such as Contextual, Textual & Topic similarity.

Any type of search be it a word, sentences, small or big article, there is always a relevant article recommended.

Apart from the above category of results, there are keywords and article summaries presented, allowing the user to decide on the article to be read.

The three model results were evaluated to understand many aspects of the document segregation and how each word influenced the surrounding text, each word's significance, topic modelling method & performance.

1. LDA Model

Once the number of Topics was finalized through steps mentioned in above sections, the final LDA model was built and trained on a huge data set 3623 odd records. This model is then used to recommend articles for Test article predictions based on topic similarity

2. LSI Model

Using similar document term matrix and corpus, LSI model is built that is recommending the articles based on the Matrix Similarity.

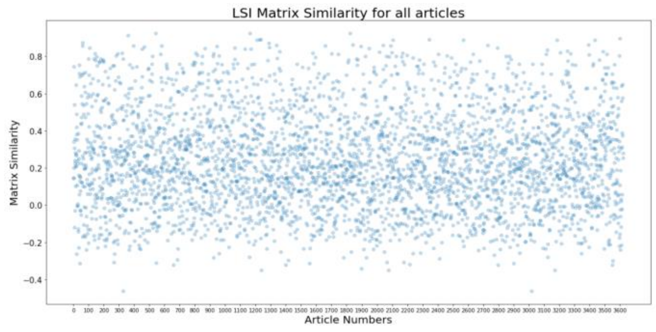


Figure 1. LSI Matrix similarity for word 'virus'

For example, if the text article is word "virus," then above figure shows the scatter plot LSI matrix similarities against all the articles, closer to 1 is the better similarity match. Also, from above plot, we can infer that for this word, most of the articles have similarity between 0 to 0.4.

3. Bert NLI Model

Using hugging face transformers, the article was recommended using pre-trained Bert model and document term matrix. The recommended journals were based on Textual similarity

D. LDA Model- Accuracy Increase

As mentioned earlier, perplexity score & coherence score are metrics used for Model evaluation & accuracy increase.

Perplexity metric [32] as measuring how probable some new unseen data is given the model that was learned earlier. Topic Coherence [33] measures score a single topic by measuring the degree of semantic similarity between high scoring words in the topic.

By increasing the iterations, the accuracy of suggested article become more relevant. For the same, we improved the batch size parameter or chunk size [34] and used the above metrics to find the correct number of topics to tune the Model.

The coherence graph is a plot of coherence score against of topics, and visualization shows us the peak score of several topics.

The Topic with highest score was used as the value for the number of topics parameter for model building. Below is the coherence plot for trained model with 1000 epochs:

- Number of topics:7 and 1000 epochs

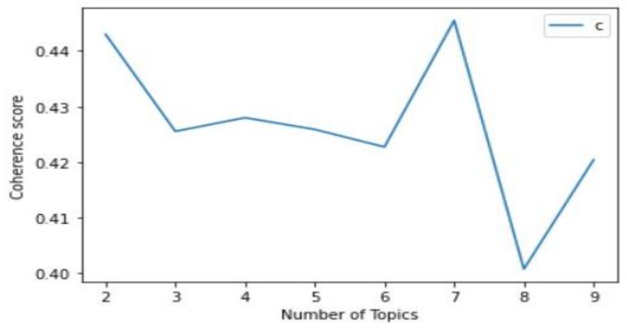
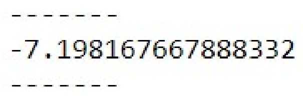


Figure 2. Coherence Plot for 1000 epochs

The perplexity score [19] is measure of how a model predicts a sample; this is achieved through log perplexity method of the model where the document term matrix is passed as input. The perplexity function from the Genism model helps in estimating the score. Below are the perplexity scores for:

- Model that was trained for 1000 epochs



```
-------
-7.198167667888332
-------
```

Figure 3. perplexity score for 1000 epochs

E. LSI Model- Accuracy Increase

Hyperparameter tuning is one of the methods to further increase the accuracy of the model keeping performance in balance. For LSI model, two important parameters are tuned while building a model: power_iters and extra_samples [40].

These two parameters affect the accuracy of the front-end multi-pass algorithm. Increasing the number of power iterations is improving the accuracy of model but lowers performance.

F. Word Cloud

The *word cloud* library [20] was used to plot all the most representative words in topics of the model and gave an overview of the words that can be expected to influence the article recommender for a test article.

According to below figure, we can infer that articles are based on medicinal studies, patients, pandemics, and other healthcare related topics.

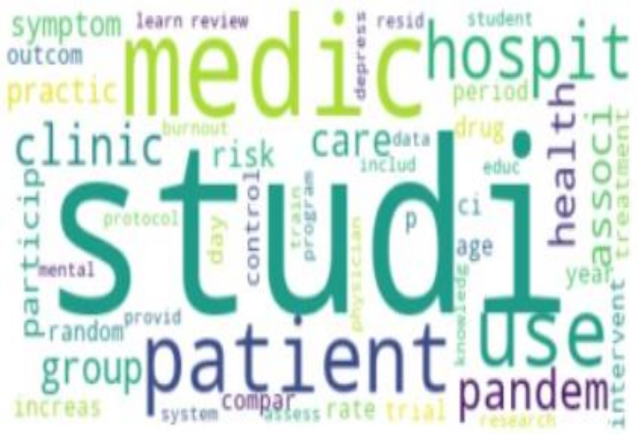


Figure 4. Word Cloud

VI. WEB APPLICATION

Once the models are built using the above techniques, the models are integrated into a web application using Flask Library [35]. The web application consists of 3 pages as follows:

1. Home Page- Where the users perform the search

2. Article Categorization Page- where user is presented with results based on Textual, Contextual & Topic Similarity

3. Article Detail Page- where user gets detail of complete article that was selected.

A. Tools & Libraries

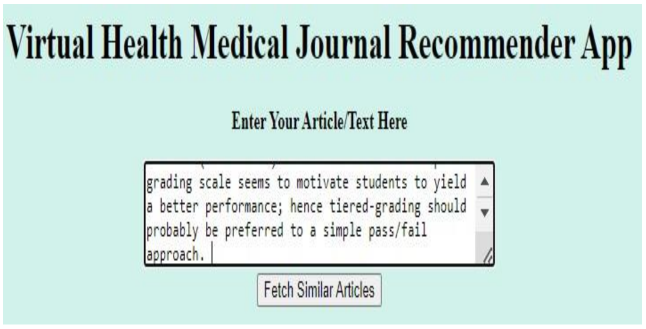
The Flask Library is a widely used package to develop web applications. Its methods defined in each HTML page triggers the appropriate function to fetch data.

LexRankSummarizer from the sumy package [36] is used here to summarize the texts of recommended article. It finds the relative importance of all words in a document and selects the sentences which contain the most of those high-scoring words.

RAKE (Rapid Automatic Keyword Extraction) algorithm [37] is used to extract the most important keywords from an article. It is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurrence with other words in the text.

B. Home Page

The Home page of the web application has a form where the user is allowed enter the text, sentence, and article of any size. When user searches here, the LDA, LSI & Bert model are executed to fetch their top ranked articles.


Virtual Health Medical Journal Recommender App

Enter Your Article/Text Here

grading scale seems to motivate students to yield a better performance; hence tiered-grading should probably be preferred to a simple pass/fail approach.

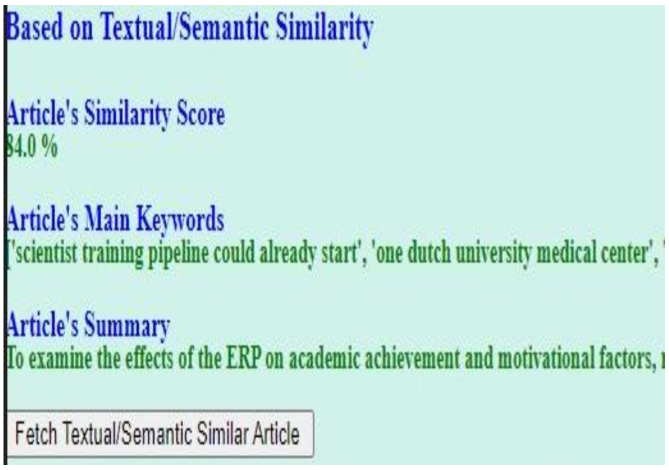
Figure 5. Home Page – Non-Keyword Search

C. Article Categorization Page – Non-Keyword Search

The *Article Categorization Page* of the web application presents results in 3 separate Categories.

1. Based on Textual/Semantic Similarity

   Here articles are presented after we compare two texts and decide whether they are similar in meaning using Bert model. This model required less parameter tuning and given accurate results as it is pretrained.


Based on Textual/Semantic Similarity

Article's Similarity Score
84.0 %

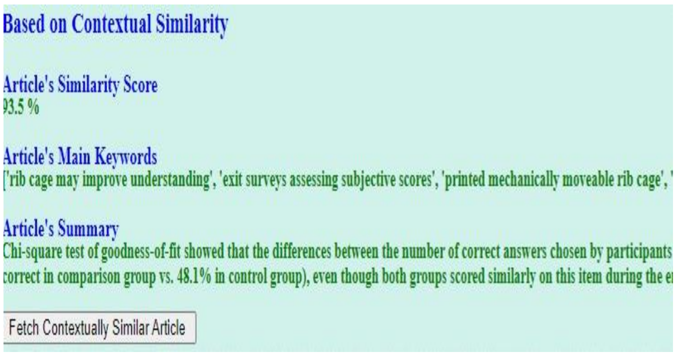
Article's Main Keywords
['scientist training pipeline could already start', 'one dutch university medical center',

Article's Summary
To examine the effects of the ERP on academic achievement and motivational factors,

Figure 6. Article Categorization Page - Textual/Semantic

2. Based on Contextual Similarity

   Here articles are presented after we compare two texts and match on the surrounding context using LSI model


Based on Contextual Similarity

Article's Similarity Score
93.5 %

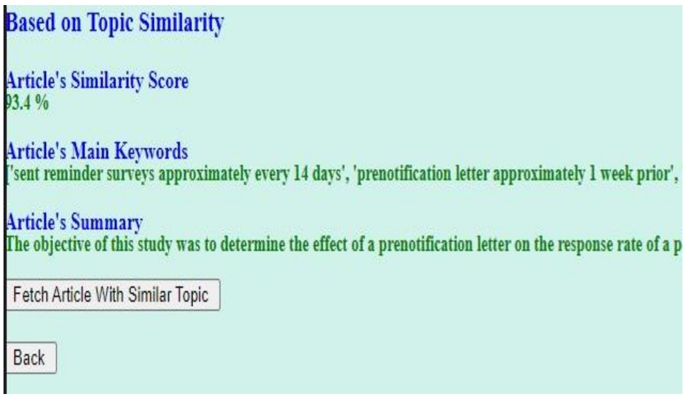
Article's Main Keywords
['rib cage may improve understanding', 'exit surveys assessing subjective scores', 'printed mechanically moveable rib cage',

Article's Summary
Chi-square test of goodness-of-fit showed that the differences between the number of correct answers chosen by participants
correct in comparison group vs. 48.1% in control group), even though both groups scored similarly on this item during the

Figure 7. Article Categorization Page – Contextual

3. Based on Topic Similarity

   Here articles are presented after we compare two texts and match on the similar Topics with weights of words matching using LDA model


Based on Topic Similarity

Article's Similarity Score
93.4 %

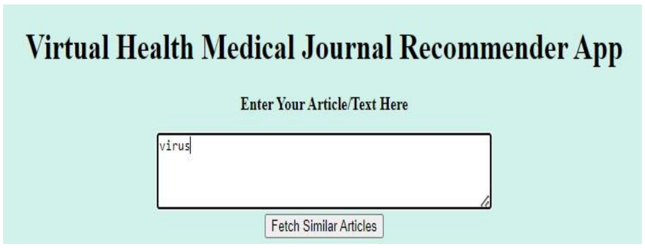
Article's Main Keywords
['sent reminder surveys approximately every 14 days', 'prenotification letter approximately 1 week prior',

Article's Summary
The objective of this study was to determine the effect of a prenotification letter on the response rate of a

Figure 8. Article Categorization Page – Topic Similarity

D. Article Categorization Page – Keywords Search

The *Article Categorization Page* of the web application presents results in 2 separate Categories whenever there is single keyword or multiple keywords searched.



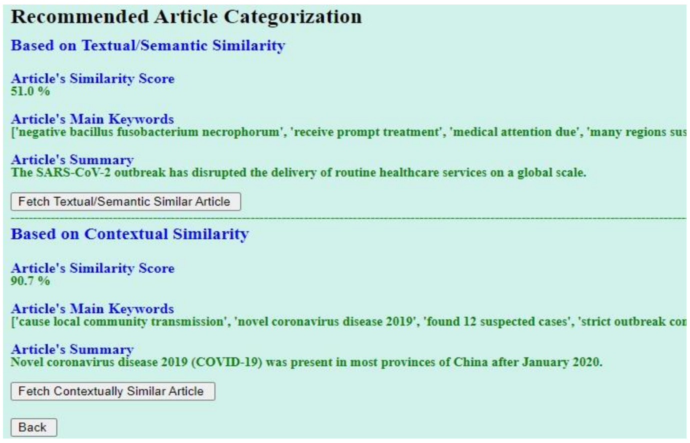
Virtual Health Medical Journal Recommender App

Enter Your Article/Text Here

virus

Figure 9. Home Page – keyword search

The LDA model works on using documents and understanding the weight of words to build topics. Since a smaller search item would not have enough information, only the LSI & Bert model would recommend the articles.



Recommended Article Categorization

Based on Textual/Semantic Similarity

Article's Similarity Score
51.0 %

Article's Main Keywords
['negative bacillus fusobacterium necrophorum', 'receive prompt treatment', 'medical attention due', 'many regions sus

Article's Summary
The SARS-CoV-2 outbreak has disrupted the delivery of routine healthcare services on a global scale.

Based on Contextual Similarity

Article's Similarity Score
90.7 %

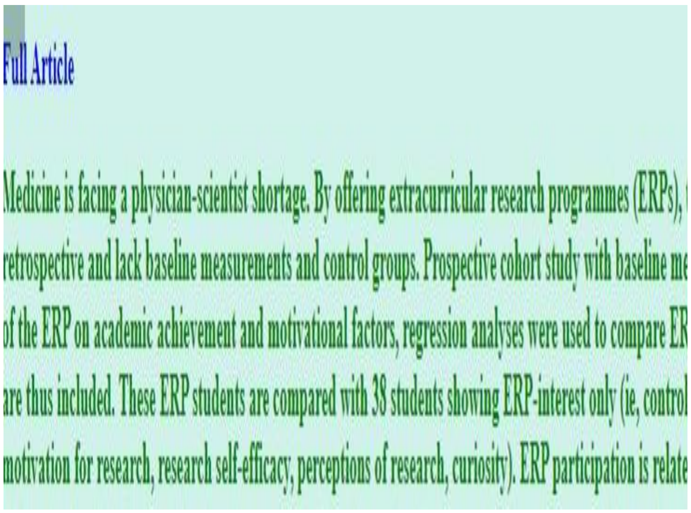
Article's Main Keywords
['cause local community transmission', 'novel coronavirus disease 2019', 'found 12 suspected cases', 'strict outbreak cor

Article's Summary
Novel coronavirus disease 2019 (COVID-19) was present in most provinces of China after January 2020.

Figure 10. Article Categorization Page – keyword search

E. Article Detail Page

The *Article Detail Page* of the web application presents results the complete detail of the article. The user can choose Article of specific Type match and click the respective button. On doing so, the result page with all the details of the chosen article is presented.



Full Article

Medicine is facing a physician-scientist shortage. By offering extracurricular research programmes (ERPs),
retrospective and lack baseline measurements and control groups. Prospective cohort study with baseline me
of the ERP on academic achievement and motivational factors, regression analyses were used to compare ER
are thus included. These ERP students are compared with 38 students showing ERP-interest only (ie, control
motivation for research, research self-efficacy, perceptions of research, curiosity). ERP participation is relate

Figure 11. Article Categorization Page – keyword search

The final page has complete article presented with all important keywords, text summarization and Similarity score of that article vs the input search

VII. Model Justification

The main goal of the project is to build a user interface that will easily recommend articles to medical researchers. The input could be a keyword, a sentence, a paragraph, or a big journal.

Many metrics were used to find a similarity between the test article and article present in the model. Various experiments during the phase of Model Building, Model tuning, interpreting results and increasing the accuracy of the same

The articles recommended assist the medical professional with more useful data for better research and understanding. Also, recommendations from three different models would help medical researchers gather more information.

A. Benefits of Model Results

The results of the 3 models are evaluated using their core modules that decide on suggestion of article to user. The user can pick which article he wants to research further on.

a. The most important words that add significant weight to the article is displayed. LDA model uses this method and returns results for every search except a keyword or text search.

The topic distribution, most important detail, of document is checked to understand which topic it is more aligned. This is determined using the weights of the words in that topic to understand the document better.

b. The article with context like the user queried article and surrounding data present in it that is matching overall context of recommended articles is pulled up for display to user. This is an LSI model search result.

c. The article with textual similarity with any other article, using the words similarity like the user queried article and words present in the test article or that have synonyms in the recommended articles are pulled up for display to user. This is a Bert model search result.

Every article is presented with important keywords, Similarity score % and Summarization of the entire article. User can first read these and then decide to read the article.

B. LDA Model - Cosine Similarity Accuracy

Following the development of LDA model data for test journals, we used a metric called Cosine Similarity [4] that determines the similarity of the data objects in a dataset. These data objects are treated as a vector. Below is the formula for cosine similarity,

$$\mathbf{cos}(\boldsymbol{x}\,,\boldsymbol{y}) = \boldsymbol{x} * \ \boldsymbol{y} \,/\, ||\boldsymbol{x}|| * ||\boldsymbol{y}||$$

where,

x = topic distribution list vector of training data

y = topic distribution list vector of new test article

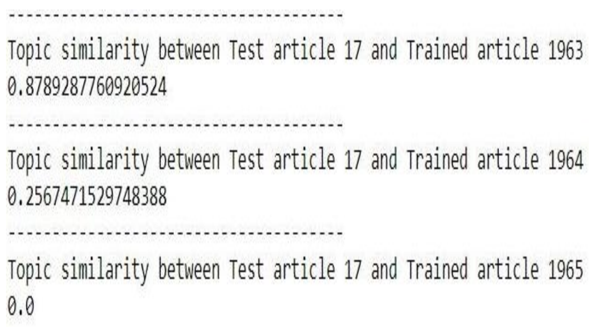
x * y = product (dot) of the vector's 'x' & 'y.'

||x|| and ||y|| = length of the two vectors 'x' & 'y.'

||x|| * ||y|| = cross product of the two vectors 'x' & 'y.'

From the topic distribution list, the values of the weight of words are sent as a list of vectors to the Cosine Similarity module which in turn performs the calculation. The cosine similarity function has a returning value between 0 to 1.

We have set the benchmark value of 0.8 to confirm whether that documents are equal to or more than 80% similar and are a good representation of each other. If probability is greater than 0.8, we can conclude that the topics are closely like each other.



```
------------------------------------
Topic similarity between Test article 17 and Trained article 1963
0.8789287760920524
------------------------------------
Topic similarity between Test article 17 and Trained article 1964
0.2567471529748388
------------------------------------
Topic similarity between Test article 17 and Trained article 1965
0.0
```

Figure 12. Topic Similarity Cosine score

As shown in above figure, for example, Cosine similarity for new test article 17 and trained article 1963 is 0.87 (87%), therefore we can infer that there is similarity between these two articles to some extent.

On the other hand, it is 0.25 (25%) for trained article 1964 which is less than the set threshold of 0.80, hence, this article is not recommended. Finally, article 1965 is not at all like new test article 17 because cosine similarity is 0. Thus, cosine similarity plays a key role in text similarity recognition.

C. Topic Labeling & Article Recommendation Feature

The model has a method of printing topics with which we can retrieve the words associated with the topic for a particular article. For this we need to pass the article index to this function. Using the most representative words we can predict a meaningful label manually and associate with related articles.

In this project, based on the efficiency of cosine similarity to text similarity calculation, articles will be recommended using three different models as opposed to single model.

D. LSI Model - Matrix Cosine Similarity Feature

For LSI model, metric used to fetch similarity between two articles is matrix similarity [27] module from gensim. This compute similarity against a corpus of documents by storing the index matrix in memory.

The similarity measure used is cosine between two vectors. Internally, vectors are stored as dense NumPy arrays and context is considered during calculation.

E. Significance of LSI over standard keyword search

Some articles may not be recommended by standard boolean full text search, because they do not share any familiar words with new text article.

However, after applying LSI, we can observe that they received quite high similarity score which corresponds to better context. In fact, semantic generalization is the reason topic modelling is performed in the first place. [28]

VIII. CONCLUSION

Having a resource of data for medical experiments is vital for existence and to overcome any pandemic. The COVID-19 virus is a proof of how important it is for the medical workers to be a step ahead to tackle the problem and diagnose the issues that the public face.

Providing a solution to any problem starts with having enough knowledge, accurate & relevant data, and communication. This application would assist the medical world in providing a varied range of information to understand and kickstart any research idea.

This application not only recommends articles but also provides information in a categorized format and with keywords, summaries.

All the results can then be presented to scholars, medical experts, professors, and medical professionals to add value and receive feedback to improve the projects' ability to deliver information with higher accuracy and relevant business improvements.

IX. IMPROVEMENTS

Finally, the results can be improved with more model Training, usage of varied and massive datasets. For this we need to experiment with model iterations, parameter tuning and several topics to build model.

The web application can be enhanced further by adding details of the Author of each article and how widely the Author's research papers are used for enhancement in the medical world. This would give users more confidence to believe and implement the ideas in recommended articles

Apart from suggesting just the topmost ranked article, with the similarity score, from each categorization, we can suggest more articles from each category just like the similar recommendations suggested on Netflix or Amazon applications. These suggestions would cater to the interest of the user and would help expand the research.

More time needs to be spent on trying all Topic modelling algorithms that are used in the industry. Researching on what the top technology companies are using and implementing the same for this project could yield better results.
Also, we need to interpret the coherence and perplexity score more carefully to ensure we get a highly predictive model. There is aa scope to use other metrics proposed by Data scientists or Natural Language Processing experts.

More metrics & graphical analysis can be used to interpret the results, thereby assisting with more validation of the results obtained. The project could be evaluated using the Tools mostly used by medical experts.

REFERENCES

[1] Rehurek Radim, “Multicore LDA in Python: from over-night to over-lunch”, September 9 2014, https://rare-technologies.com/multicore-lda-in-python-from-over-night-to-over-lunch/

[2] Aravind CR., “Topic Modeling using Gensim-LDA in Python,” Analytics Vidhya, July 26, 2020, https://medium.com/analytics-vidhya/topic-modeling-using-gensim-lda-in-python-48eaa2344920

[3] Navlani, A.,”Latent Semantic Analysis using Python,” Data Camp, October 9 , 2018,

https://www.datacamp.com/community/tutorials/discovering-hidden-topics-python

[4] Rao, S., “Cosine Similarity,” geeksforgeeks, October 6, 2020, https://www.geeksforgeeks.org/cosine-similarity/

[5] Prabhakaran Selva,” Topic Modeling with Genism (Python),March 26, 2018, https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

[6] Filtering stopwords in a tokenized sentence, Filtering stopwords in a tokenized sentence | Python 3 Text Processing with NLTK 3 Cookbook

[7] Edward Loper, “NLTK Tutorial: Modelling Probabilistic Systems” (ed.ac.uk)

[8] Yang Gao, Nicolo Colombo, Wei Wang, “Adapting by Pruning: A Case Study on BERT”, bert_nli/adapt-by-prune.pdf at master · yg211/bert_nli · GitHub

[9] Scott Deerwester, Susan T. Dumais. George W. Furnas, Thomas K. Landauer, Richard Harshman, “Indexing by Latent Semantic Analysis” Indexing by latent semantic analysis (bham.ac.uk)

APPENDIX

[10] Article Recommender Source code

Complete Project Source Code

The above file is the Python Jupyter notebook with all the code and analysis and inferences from the Project developed by the authors of this paper.

[11] Virtual Health Library Dataset

http://red.bvsalud.org/en/

[12] Topic Modelling

Topic Modeling and Latent Dirichlet Allocation (LDA) in Python | by Susan Li | Towards Data Science

Grouping documents according to themes and extracting valuable insights are need of the hour. The above link gives detailed information of how to manage data with less time consumed to build probabilistic topic models and use statistical algorithms that analyze words in original text documents to uncover the thematic structure

[13] Coherence Score

https://www.baeldung.com/cs/topic-modeling-coherence-score

To efficiently use a Topic modelling technique and extract relevant topics, it was key to know how to interpret the result and set a threshold. Coherence score measurement is widely used in Natura Language Processing domain and the above link gave a deep understanding of its usage.

[14] LDA Multicore vs LDA

https://radimrehurek.com/gensim/models/ldamulticore.html

Latent Dirichlet Allocation (LDA), one of the most used modules in genism but it still runs only in single process, without full usage of all the cores of modern CPUs. Because in recent years CPUs with multiple cores has become standard even for low-end laptops, a multicore implementation can be very useful and might save a lot of time waiting, especially for very large dataset.

[15] Porter Stemmer

Python Programming Tutorials

The idea of stemming is a normalizing method. Many variations of words carry the same meaning, other than when tense is involved. The reason we stem is to shorten the lookup and normalize sentences.

[16] Regular Expressions – re.sub

re — Regular expression operations — Python 3.11.0 documentation

This module provides regular expression matching operations like those found in Perl.

[17] Word Tokenization

What is Tokenization | Methods to Perform Tokenization (analyticsvidhya.com)

Tokenization is splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens

[18] Corpora Dictionary

corpora.dictionary – Construct word<->id mappings — gensim (radimrehurek.com)

This module implements the concept of a Dictionary – a mapping between words and their integer ids.

[19] Converting Dictionary to Bag of Words

gensim.corpora.Dictionary.doc2bow (tedboy.github.io)

Convert *document* (a list of words) into the bag-of-words format = list of *(token_id, token_count)* 2-tuples

[20] Perplexity Score

Perplexity To Evaluate Topic Models (qpleple.com)

The most common way to evaluate a probabilistic model is to measure the log-likelihood of a held-out test set.

[21] Word Cloud Library

Python Word Clouds Tutorial: How to Create a Word Cloud | DataCamp

A visualization filled with lots of words in varied sizes, which represent the frequency or the importance of each word

[22] Corpora Dictionary

corpora.dictionary – Construct word<->id mappings — gensim (radimrehurek.com)

This module implements the concept of a Dictionary – a mapping between words and their integer ids.

[23] Model Evaluation

Evaluate Topic Models: Latent Dirichlet Allocation (LDA) | by Shashank Kapadia | Towards Data Science

[24] Boolean Keyword queries

Full-text search - Wikipedia

[25] Hugging Face Transformers

Transformers (huggingface.co)

Used for natural language processing

[26] Latent Semantic Indexing (LSI model)

Similarity Queries — gensim (radimrehurek.com)

Demonstrates querying a corpus for similar documents

[27] Amazon SageMaker Algorithm
How LDA Works - Amazon SageMaker
Amazon SageMaker LDA is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of distinct categories

[28] Gensim Matrix Similarity
gensim.similarities.MatrixSimilarity (tedboy.github.io)

[29] Similarity Queries
Similarity Queries — gensim (radimrehurek.com)

[30] Scikit Learn Cosine Similarity
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

[31] Latent Semantic Indexing an Example
Microsoft Word - LSI-Eg.doc (cuhk.edu.hk)

[32] Standard Metrics for LDA Model – Perplexity score
Standard Metrics for LDA Model Comparison

[33] Evaluation of Topic Modeling: Topic Coherence
Evaluation of Topic Modeling: Topic Coherence

[34] Chunk Size LDA Model
Chunk size LDA MODEL

[35] Flask Web App
Flask Web App

[36] Flask Web App
Lexrank summarizer

[37] Flask Web App
Rapid Automatic Keyword Extraction

[38] AWS SageMaker
Machine Learning – Amazon Web Services

[39] BERT Cosine Similarity
Sentence Similarity With BERT | Towards Data Science

[40] Power Iterations and Extra Samples - LSI
models.lsimodel – Latent Semantic Indexing — gensim (radimrehurek.com)