School of Computer Science and Informatics

Cardiff University

2020

CMT400 – Master's Dissertation – 60 Credits

Research Paper Recommendation System

Author: Subodh Gholve

Supervisor: Luis Espinosa-Anke

Moderator: Alun D Preece

Degree Programme: MSc Computing and IT Management with Placement

Acknowledgements

I would like to specially thank my supervisor Luis Espinosa-Anke for his extreme support throughout the project during the most difficult time in my life, understanding me on a personal level and responding to my queries even during Christmas holidays.

I would like to thank my friends Giulia Maria Bellan, Manali Lokhande and Taehui Kim for being a family and supporting me during this pandemic whilst I have not been able to visit home for a very long time.

I would like to thank my team at my workplace (Atradius) for their understanding, shedding off workload form my shoulders, providing feedback and allowing me time to focus on the thesis.

Table of Contents

Table of Figures

Abstract

Whilst there are ample of studies being conducted to end this global pandemic; this project offers an opportunity to help Covid-19 researchers for their extremely helpful contributions towards making things "normal". Covid-Aware, the paper recommendation tool is developed to help the researchers in finding the research papers most relevant to their paper of interest. This does not use user's evaluation and the recommendations are calculated using the information within the dataset of the chosen paper. The recommender is developed using principles of content-based filtering, one of the most common approaches to recommender systems. To achieve this, I have used Cosine similarity function from Scikit-learn machine learning libraries in Python, Flask as backend and a simple HTML frontend to display the results. The dataset being used is a cleaned version of the CORD-19 Open Research Dataset provided by Semantic Scholar for use by global researchers. This dataset contains Abstract IDs and Paper Abstracts for the research papers. The recommendation tool takes an Abstract ID as input to compare the respective Paper Abstract with the rest of the dataset to find the Top 10 most relevant documents.

1. Introduction

As technology has been advancing at a rapid pace, we see a massive increase in information. To handle such huge data, we automate the analytical model building using machine learning. There has been good growth in the recommender systems area over the last decade and is continually improving. The interest remains high as with the rise in digital services, personalized recommendations have been very useful for users to come across suitable content. Recommendation systems are very popular on the internet on various e-commerce sites, on video-on-demand services, etc. Lately, we have seen advertisements being personalised too. Although there have been major improvements in the recommendation systems, there is good scope for further improvements in them (Adomavicius, Tuzhilin, 2005). It is not easy for the content providers to be distinct. Hence, recommender systems are developed as a solution. The purpose of a Recommendation System is to unite the information and the users: this helps both the parties by providing useful information for the users and by increasing the popularity of the recommender providers (Ma. K., 2016).

As it has been mentioned in the paragraph above, hundreds of thousands of studies, similar but not the same in details and progress, now populate the internet, making it hard for people to research according to one's preferences. The aim of the study is to build a tool to research studies and papers on the topic of the virus of Covid-19 using a recommendation system, in order for the research to be personalised and suitable.

1.1. Rationale of the study-

The pandemic of Covid-19 has affected every area of our lives, included technology. Together with this event, the explosion and expansion of the world wide web have caused an out spacing of the extraction of useful information during studies and research (Agarwal et al., 2005). According to a study carried out by Paul Resnick on Recommender Systems (Resnick, 1997), there has been a sudden increase in the number of research papers, also due to the incredible progress of digitalisation in the last years.

In order to find the papers suitable to our research, a recommendation system has to go through thousands and thousands of options, so many of these have ways to personalise the search (Agarwal et al., 2005), recommending users with similar research carried out by others with related interests: this is called Content-Based recommendation system or Content-based filtering. A second approach is called Collaborative filtering. This latter one analyses information collected from users' habits, recommending potential interesting items (Agarwal et al., 2005). The differences between these systems will also be explained in its entirety in the section dedicated.

The choice to develop an application based on recommendation systems has been driven by mainly two thoughts. Firstly, how helpful it would be to many students, doctors and the people in the first line fighting during these difficult times; secondly, because recommendation systems are almost everywhere, on every website of streaming or e-commerce, whereas not many have been used for research.

1.2. Scope and limitation of the dissertation-

This paper mainly focuses on the explanation and description of the development of my Research Recommendation system. As the topic is prevalently mathematics and of coding, no limitations were found in the progress of this research. However, there are limitations in the application of this service: as it has been developed, this tool can only be used as a standalone service which would not be very efficient. Nevertheless, when integrated with a full-fledged application / website, it would provide as an added functionality and deliver a higher level of efficiency.

1.3. Structure of the dissertation-

This dissertation will be divided into chapters and sub-chapters. There are 13 chapters in total. Chapter 5 will provide a background of recommendation systems, indispensable for the understanding of the approach and the implementation of the tool. The last 5 chapters will be focussed on personal reflections, future work and evaluation. Finally, there will be a section dedicated to the references to everything that has helped me writing this dissertation.

2. Aim and Objectives

The aim of the project is to research on Recommender Systems and find an appropriate way to create an application which helps Covid-19 researchers to find Research papers. The returned query should also provide top 10 similar research papers from the dataset so that the Researchers are made aware about similar papers without having to change their search query. The returned results for relevant papers should optionally have corresponding relevancy scores to have an idea how similar the papers are. There are a lot of recommendation systems approaches but not all of them are appropriate in our case.

Hence, to achieve this objective, we need to –

- Research for a dataset with Abstracts and use it as the data source for our software.
- Research on how recommendation systems work and the best approach to design one for recommending Research papers.
- Choose suitable programming language with appropriate libraries to efficiently process the dataset and return Abstracts relevant to the entered Research Paper ID.
- Design a basic Frontend for the user to input an abstract ID and see the expected results with the similarity score.
- Choose a suitable Web Framework for our backend.

3. Problem Statement

To find research papers on a digital library or a website is not an easy task. Researchers spend hours finding the right and relevant research papers as the search queries usually only return the most relevant research paper based on their search query. It is one of the main problems faced by researchers. The user needs to find research papers of their interest. As a solution to this, a recommendation system can help the users by recommending them appropriate research papers.

The problem faced here is to create an application or a piece of software that instantly looks through the entire library / dataset and returns the most relevant research papers to the user.

A number of questions would arise when we build this recommender from scratch, especially when there is no user information as most of the recommender systems work based on user's history like browsing, purchases, wish lists, etc. Therefore, to solve this problem, we will break it into multiple problem statements which will help us set the expectations clearer.

- How to recommend relevant research papers based solely on one input?
- What are the best frontends, backends and programming languages appropriate to process this data?
- How to calculate the similarity between multiple research papers?

Going through a lot of research papers in this area, it can be derived that recommendation systems are mostly common in movie recommendation systems and ecommerce websites as they have a very high number of visitors. There have been relatively less studies based on research paper recommender systems, but it has been well discussed. The top points that can be noted by reviewing these research papers are –

1. We have a lot of techniques to use when it comes to recommender systems like Content-based filtering, collaborative filtering, Hybrid Filtering, etc.

2. A lot of articles have approved of Content-based Filtering to be the best approach for recommendation systems for Research papers.

4. Training –

I am very new to Machine Learning or programming in general. Consequently, starting up with an NLP based project with such high time constraints was a very difficult task. Apart from understanding theoretical concepts like Cosine similarity, I had to first understand the concepts of Flask, NLP and Machine Learning using relevant Youtube tutorials, official documentations and other public articles. Putting it all together was a big challenge as well. I had some basic understanding of numpy and pandas studied during first semester of this degree, which helped me pick up the pace in my training and create a whole usable application. To be able to version-control my code, I also had to understand the concept of versioning and to learn using Github in an effective way.

5. Background

5.1. Famous Recommender Systems:

5.1.1. Video on Demand services:

A movie recommender is a very important system on Video on Demand services or more specifically OTT (Over the top) media services like Netflix or Amazon Primevideo. The reason being that considers the users' watching history and suggests them TV shows/movies based on what they have watched and liked. It is vital, as it is quite tricky, to find something of their likings amongst the vast numbers of available videos (Miller et al., 2003). An example of Netflix's recommender system is shown in Figure 1 below where the user has watched a TV show and the recommender system has suggested other TV shows related to that series. This could be based on multiple factors like genre, plot, cast, etc. As we can see, this recommendation system considers what you have watched previously and tells you why they have recommended these titles.

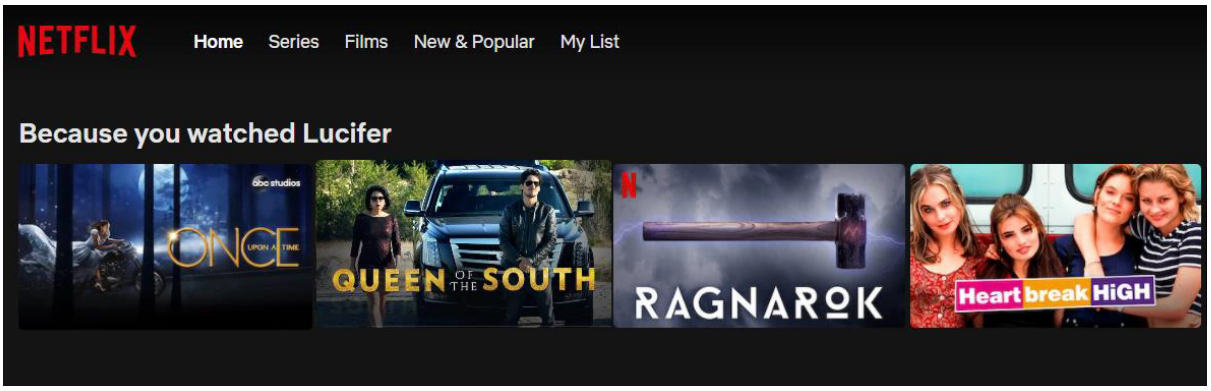


Figure 1. Netflix movie recommender - personalised list.

5.1.2. E-commerce Websites:

As Video on Demand services mostly use a 'personalised' recommendation list, there is another critical factor which is relevant. The relevancy factor matters equally in E-commerce websites (Ma. K., 2016). Using eBay as an example, if the user tries to search for a PC game then the user would be recommended gaming categories with the type of games related to what the user had searched for and viewed as shown in the Figure 2 below. This is an example of personalised list.

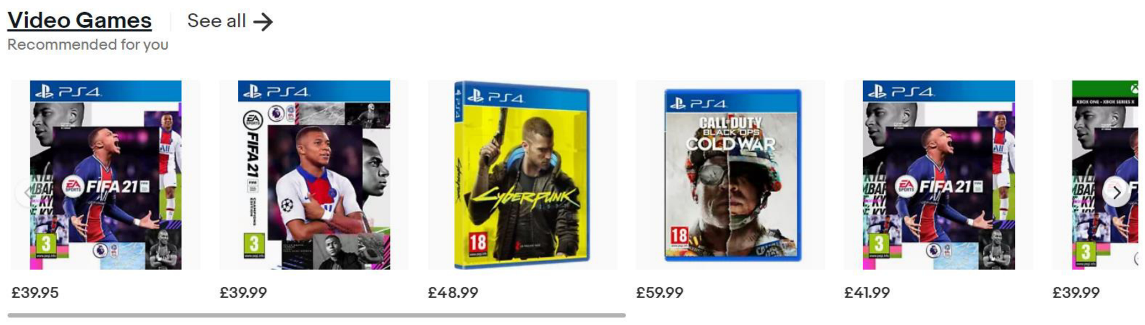


Figure 2. Personalised list of eBay.

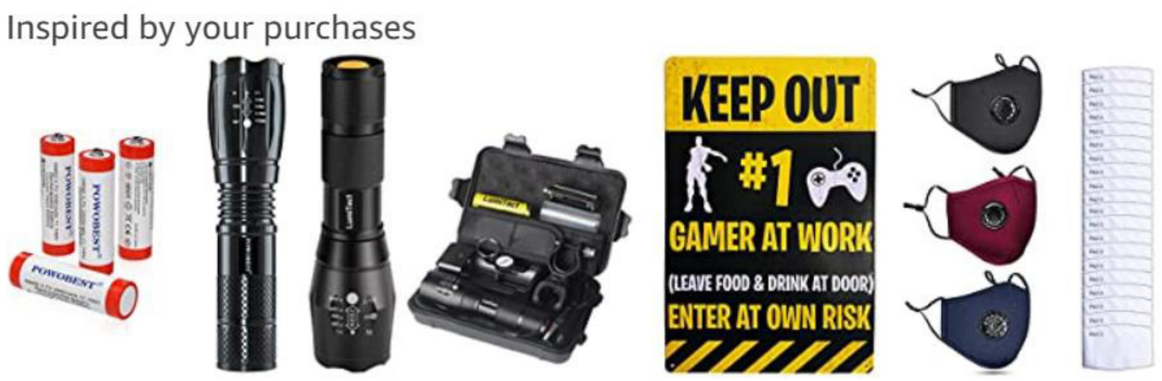


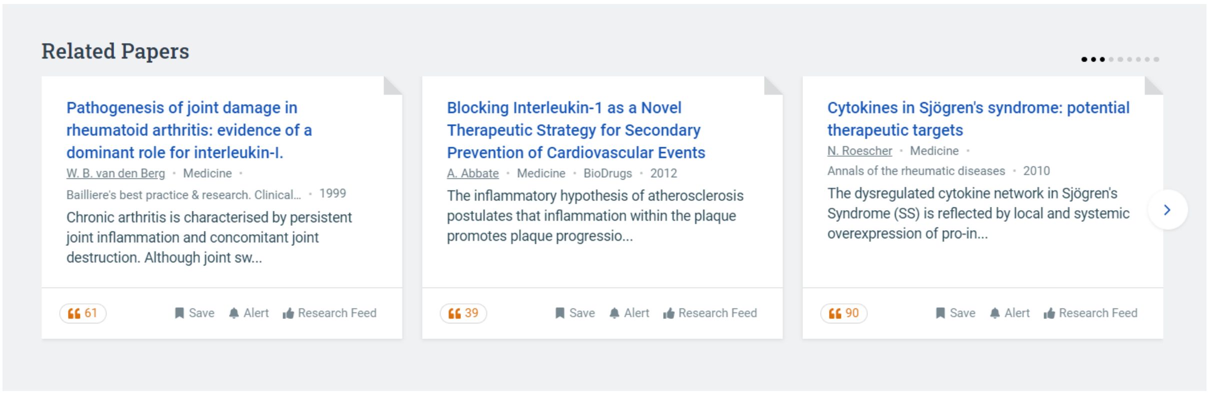
Figure 3. Relevant recommendation list of Amazon.

As you can see in Figure 3, this list is relevant to what the users have previously purchased and is recommended for the user.

After comparing and observing, I personally believe that the recommendation lists of Amazon consist of personalisation and relevancy, but eBay mostly focuses on personalisation, making Amazon much more convenient for the shoppers as they recommend items based on what you are currently viewing, what you have purchased in the past and items usually bought together.

5.1.3. Research paper websites –

As I have already mentioned, there are much lesser websites which contain recommendation systems in research paper platforms, but some websites do have the implementation to suggest relevant papers.


Related Papers

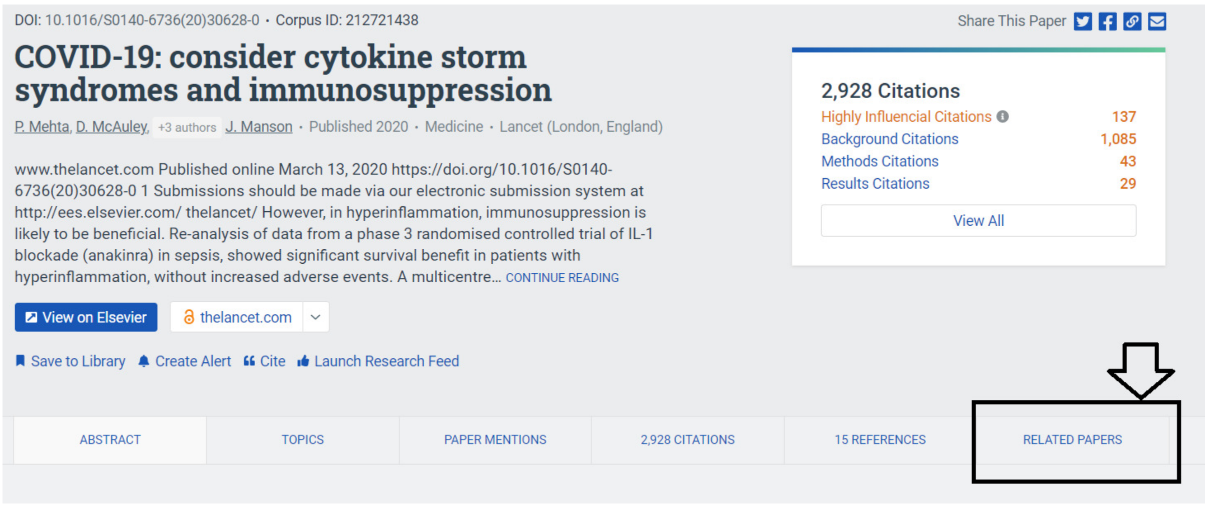
Pathogenesis of joint damage in rheumatoid arthritis: evidence of a dominant role for interleukin-I.
W. B. van den Berg · Medicine · Bailliere's best practice & research. Clinical... · 1999
Chronic arthritis is characterised by persistent joint inflammation and concomitant joint destruction. Although joint sw...

Blocking Interleukin-1 as a Novel Therapeutic Strategy for Secondary Prevention of Cardiovascular Events
A. Abbate · Medicine · BioDrugs · 2012
The inflammatory hypothesis of atherosclerosis postulates that inflammation within the plaque promotes plaque progressio...

Cytokines in Sjögren's syndrome: potential therapeutic targets
N. Roescher · Medicine · Annals of the rheumatic diseases · 2010
The dysregulated cytokine network in Sjögren's Syndrome (SS) is reflected by local and systemic overexpression of pro-in...

Figure 4. Recommender system in Research papers sites (semanticscholar)

In Figure 4 above it can be noticed that some research paper websites have a recommendation paper system implemented. This snapshot has been taken from semanticscholar. It is a fairly less emphasized section on the page as it focuses more on the abstract, topics, etc as shown in Figure 5. However, it is one of the most famous applications in terms of research paper recommender tools.


DOI: 10.1016/S0140-6736(20)30628-0 • Corpus ID: 212721438

COVID-19: consider cytokine storm syndromes and immunosuppression

P. Mehta, D. McAuley, +3 authors J. Manson • Published 2020 • Medicine • Lancet (London, England)

www.thelancet.com Published online March 13, 2020 https://doi.org/10.1016/S0140-6736(20)30628-0 1 Submissions should be made via our electronic submission system at http://ees.elsevier.com/ thelancet/ However, in hyperinflammation, immunosuppression is likely to be beneficial. Re-analysis of data from a phase 3 randomised controlled trial of IL-1 blockade (anakinra) in sepsis, showed significant survival benefit in patients with hyperinflammation, without increased adverse events. A multicentre... CONTINUE READING

2,928 Citations

Highly Influencial Citations	137
Background Citations	1,085
Methods Citations	43
Results Citations	29

Figure 5. Semanticscholar Paper Recommendation.

5.2. Survey of Recommendation Classes

There has been a lot of research in the recommender systems area using a range of approaches. Most of them can be categorised as Stereotyping, content-based filtering, collaborative filtering, Co-occurrence, Graph-based, Global relevance and Hybrid (Beel, Breitinger et Langer, 2015). However, the most common approaches for building a recommendation system are –

1. Collaborative Filtering.
2. Content-based Filtering.
3. Hybrid Filtering.

Since hybrid filtering is not relevant to our context, I will focus more on Collaborative and content-based filtering.

5.2.1. Collaborative based Filtering

The theory of collaborative filtering marks back to 1994 which was explained by Resnick et al. The theory was explained as - "*the users will like what like-minded users will like*". So, this theory is based on a concept that when two different users rate the same item in the same way, they are considered as like-minded and might also like another item based on what else the other user has liked (Resnick et al., 1994). For example, if user A has rated 4 stars for item K and user B has also rated 4 stars for item K, then user B is likely to rate item L as (say) 5 stars if user A has already rated it 5 stars. So, the user B will be recommended similar items if they match tastes with user A based on how alike their ratings are.

This type of filtering is the most famous amongst recommender systems as it based on user's choices according to the history of their actions. User-based collaborative filtering was the first introduced type of collaborative filtering and then Amazon came up with item-based collaborative filtering in 2000 (Ma. K., 2016).

Let us briefly discuss both the ways.

5.2.1.1. User-based Collaborative Filtering –

This type of filtering is based on users. It is assumed that when a user likes items which are liked by another user, they will have similar preferences. Hence, the first thing to do would be to find other people with similar preferences, for which we will have to figure out which users have liked similar items. (Ma. K., 2016); (Singh et al., 2020)

To make this easier, let us consider two users *a* and *b,* and *N(a)* and *N(b)* are items set which are liked by the respective users. Here the similarity is given by –

$$sim_{ab} = \frac{|N(a) \cap N(b)|}{|N(a) \cup N(b)|}$$

The likeability of User *a* for item *x* can be given by –

$$p_{ax} = \sum_{b \in S(a,k) \cap N(x)} sim_{ab} p_{vx}$$

	Item A	Item B	Item C	Item D
User 1	Likes	-	Likes	Likes
User 2	-	Likes	Likes	-
User 3	Likes	-	*Recommended*	Likes
User 4	Likes	-	Likes	*Recommended*

Table – User-based Collaborative Filtering.

The Table above explains how User-based Collaborative Filtering works. User 3 and User 4 have 2 common likes with User 1. Hence, both the users are recommended a third item based on what User 1 likes.

5.2.1.2. Item-based Collaborative Filtering -

Item-based Collaborative Filtering is somewhat similar, but the difference is that instead of looking for similar users, we are looking for similar items. Hence, in item-based Collaborative Filtering, it is considered that people will like items which are similar to the ones they have liked in the past. So here we find the items similar to what the user has already liked.

Let us consider x and y are items and N(x) and N(y) are the user sets who like the items respectively.

Similarity is then given by –

$$sim_{xy} = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|}$$

The likeability of User *u* for item *x* can be given by –

$$p_{ux} = \sum_{y \in S(x,k) \cap N(u)} S_{xy} p_{uy}$$

	Item A	Item B	Item C
User 1	Likes	-	Likes
User 2	-	Likes	-
User 3	Likes	-	*Recommended*
User 4	Likes	-	Likes

Table – Item-based Collaborative Filtering.

The table above shows how if item A is liked by a group of users, then item C is also liked by them. According to this assumption, Item C is recommended to User 3 in the table above.

This is how Collaborative filtering works, but when compared to Content-based Filtering, Content-based method does not consider user ratings or preferences.

5.2.2. Content-based Filtering.

One of the most commonly used recommendation system methods is the Content-based filtering system. One key element of CBF is the user modelling process, wherein the interests of users are deduced from the items that users viewed or liked. (Beel, Breitinger et Langer, 2015).

In recommendation systems, the content-based filter is a vital approach. The main concept here is to recommend results most similar to what the user has looked for before (Ma. K., 2016).

Content-based filtering holds a high advantage in front of some other approaches. Mainly, Content-based filtering permits the recommendation system to have a user-based personalization which helps the recommender to figure out the best recommendations for every user particularly, instead of being limited (Singh et al., 2020).

To prototype this methodology, the Vector Space Model (VSM) method is used. It draws the similarity of an item using the information in the document and utilizes the concept of TF-IDF (Term Frequency-Inverse Document Frequency). TF-IDF was the foremost well-known weighting plot (70%) among those approaches for which the plot was stipulated (Singh et al., 2020).

A major factor of CBF is user modelling process: here the users' interests are inferred from their history of their previous interactions. This information is usually textual. The user's history of interactions can be related to their searches, purchases or similar actions depending on the application. This specific history of items is represented using its features. They could be n-grams, words or phrases. In general, only the most descriptive characteristics are used for this modelling. After these characteristics are identified, they are contained as a vector which stores these characteristics and the weights assigned to the item. In the recommendation process, these vectors are compared with the user model (Beel, Breitinger et Langer, 2015).

As previously mentioned, TF-IDF is one of the best information retrieval methods. TF-IDF is given as follows –

Let us assume that N is the total number of documents, N_x is the total number of words in document, d_x. $N_{y,x}$ are the number of times the word y appeared in document d_x. The $TF_{y,x}$ is defined as (Adomavicius, Tuzhilin, 2005) (Philip et al., 2014).

$$TF_{y,x} = \frac{N_{y,x}}{N_x} \tag{1}$$

But the words that are repeated in most of the documents are not particularly helpful in finding out the relevant ones. This is where IDF comes into picture, we utilise the Inverse Document Frequency in along with TF which is given by – (Adomavicius, Tuzhilin, 2005) (Philip et al., 2014).

$$IDF = \log \frac{N}{n_y} \tag{2}$$

Where, n_y = number of documents containing term t_y.

From (1) and (2), we get the TF-IDF by,

$$w_{y,x} = TF_{y,x} \times IDF \tag{3}$$

But for the weights to be in [0,1] interval and for documents to be signified by vectors of equal length, the weights in (3) are normalised by cosine normalisation as (De Gemmis, Lops, Semeraro, 2011)-

$$w_{y,x} = \frac{TF\text{-}ID(t_y, d_x)}{\sqrt{\sum_{s=1}^{|k|} TF\text{-}ID(t_s, d_x)^2}} \tag{3}$$

Finally, the content can be given by –

$$Content(d_x) = (w_{1x}, w_{2x}, \ldots., w_{kx}) \tag{4}$$

(Adomavicius, Tuzhilin, 2005); (Ma. K., 2016).

The vectors of these weights for each word can be used to be projected in the multidimensional space to find the similarities between them. This measure of similarity is calculated using Cosine Similarity.

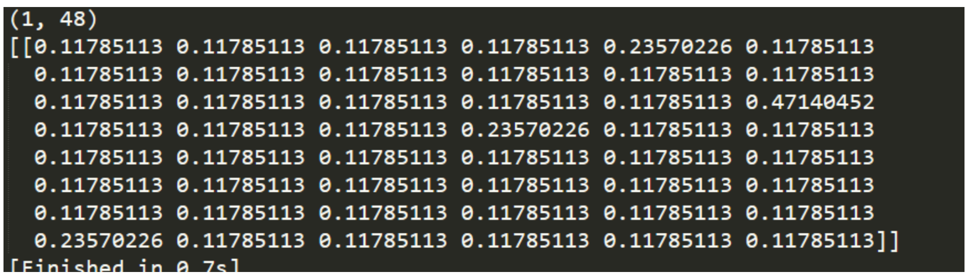
Count Vectorizer -

Apart from TF-IDF, there is another vectorizer used in a lot of recommendation systems, which is Count Vectorizer in Sklearn.

Count Vectorization is one of the most basic ways to represent the textual data numerically (Heidenreich H., 2018). It returns an encoded vector with a length of the whole vocabulary and integer count for the number of instances every word showed in the document. The CountVectorizer from scikit-learn can also be used in a way similar to TFIDF to achieve the vectorization. The difference in between these two can be explained by the following example.

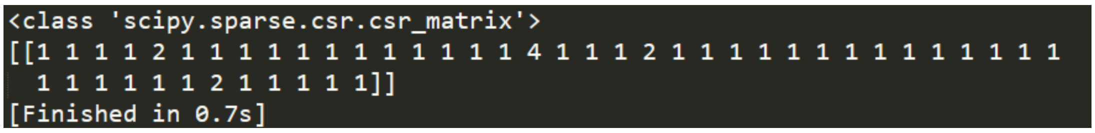
I performed a comparative analysis of both the Vectorizers to see how they vectorize the textual data. The Vectorizers token counts responding to an abstract can be seen in the Figures 6 and 7.

Text input – "Middle East respiratory syndrome coronavirus (MERS-CoV) was first identified in a human with severe pneumonia in 2012."



```
(1, 48)
[[0.11785113 0.11785113 0.11785113 0.11785113 0.23570226 0.11785113
  0.11785113 0.11785113 0.11785113 0.11785113 0.11785113 0.11785113
  0.11785113 0.11785113 0.11785113 0.11785113 0.11785113 0.47140452
  0.11785113 0.11785113 0.11785113 0.23570226 0.11785113 0.11785113
  0.11785113 0.11785113 0.11785113 0.11785113 0.11785113 0.11785113
  0.11785113 0.11785113 0.11785113 0.11785113 0.11785113 0.11785113
  0.11785113 0.11785113 0.11785113 0.11785113 0.11785113 0.11785113
  0.23570226 0.11785113 0.11785113 0.11785113 0.11785113 0.11785113]]
[Finished in 0.7s]
```

Figure 6. TFIDF tokens.



```
<class 'scipy.sparse.csr.csr_matrix'>
[[1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 4 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 2 1 1 1 1 1]]
[Finished in 0.7s]
```

Figure 7. CountVectorizer tokens.

In the final line of both the images it can be seen that Count Vectorizers assign a value of 1 to each word and does not segregate between words, assigning equal weight to all the words.

TFIDF has a different tokenization mechanism which assigns lower weight to the repeated words compared to the unrepeated ones.

5.2.2.1. Cosine Similarity-

Cosine similarity is basically the measure of cosine angle between two vectors. In our case, it compares two documents on a normalized level. This is done by finding the dot product between two vectors. Let us consider a 2D plane to explain a dot product of two vectors a and b. "*The dot product between two vectors corresponds to the projection of one of the vectors on the other.*" (Grimaldi, E., 2018).

Hence, the dot product between two similar vectors is equal to their product, which is 1 or less than 1. But they are completely different if they are perpendicular and their dot product is zero (as cos 90 = 0). (Philip et al., 2014); (Grimaldi, E., 2018); (Singh et al., 2020)

$$a.b = [a_1 a_2 \ldots\, a_n].\begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{bmatrix} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \sum_{i=1}^{n} a_i b_i$$

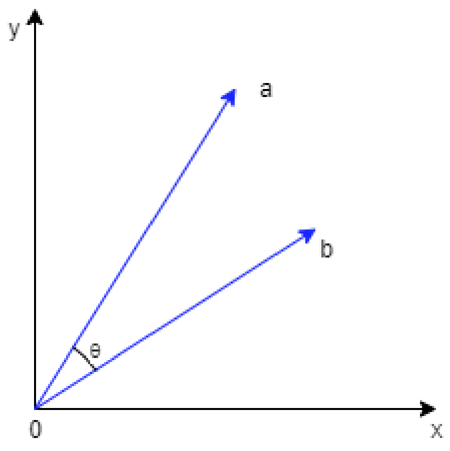


Figure 8. Two vectors in a 2D plane – Cosine Similarity.

As we have illustrated this example in a vector plane in Figure 8, the angle between a and b is θ. The similarity is measured based on this angle, lesser the angle, more the similarity. Hence, if this angle θ between the vectors is 0, they are identical. (Philip et al., 2014); (Grimaldi, E., 2018); (Havan, E., 2019) ;(Singh et al., 2020)

Therefore, the similarity can be given by -

$$similarity = \cos(\theta) = \frac{a.b}{\|a\| \times \|b\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} {a_i}^2}\sqrt{\sum_{i=1}^{n} {b_i}^2}}$$

According to the definition, if the vectors are identical the similarity will be equal to 1, if they are perpendicular, it will be 0. Hence, the value will always be between 0 and 1 which will tell us how similar the vectors are. In our case, it will tell us how similar the two research papers are.

5.3. Flask

Flask is one of the most popular Python web application frameworks. It is a light weight WSGI (Web Server Gateway Interface) web application framework. It is designed in a way that getting started is fast and simple, while being able to scale up complex apps (Flask, 2020). A quick start template is available on their official website which creates a minimal "Hello World" application to get the application running and then build on top of that.

It has two components based on Pocoo Projects which are Werkzeug and Jinja.

1. Werkzeug

Werkzeug (Greek for tool or work-stuff) is a WSGI web application library. Flask wraps Werkzeug for handling details of WSGI. Werkzeug comes with many useful features like an interactive debugger, a full-featured request object, a response object, http utilities to handle http data and requests, a routing system to match URLs to endpoints, a threaded WSGI server, a test client for HTTP request simulation (Werkzeug, 2020).

2. Jinja (2.0)

Python uses a full-featured template engine which is Jinja2. It has an optional integrated sandboxed execution environment and a full unicode support (Jinja, 2020).

Jinja2 is a powerful template engine for Python, with a range of features, some of which are –

- Sandboxed execution mode. Every facet of this template execution is observed and whitelisted or blacklisted, according to the preference (Jinja, 2020).
- Powerful automatic HTML escaping system. This is to prevent cross-site scripting (Jinja, 2020).
- Template inheritance makes it viable to utilize the same layout for all templates (Jinja, 2020).

- High performance due to real-time compilation. Jinja2 will translate the template sources on initial load into Python p-code for best runtime performance (Jinja, 2020).
- Ahead-of-time compilation (optional) (Jinja, 2020).
- Simple to troubleshoot through a debug system (Jinja, 2020).
- Configurable syntax. Jinja2 can be reconfigured to well suit output formats like LaTeX and JavaScript (Jinja, 2020).

Flask offers suggestions but it is the developer's wish to decide the tools and libraries wanted by them. Many extensions have been provided by the community which makes it easier to have new functionalities. (Jinja, 2020).

6. Data Sources

I have used an open-source dataset available online for public use. Since the recommender system being created is specifically for COVID-19 based research papers, the dataset used is the 'Cord-19: Covid-19 Open Research Dataset' (Wang et al., 2020). This dataset is regularly updated, and a fresh copy is available regularly on their website. The dataset when downloaded comes with a cluster of files which can be used as per the user's choice and application. It contains files which contains some information like Paper ID, Paper title, Paper Abstract, date of publishing, etc. This raw data is in JSON format.

The dataset which I will be using is one of the downloaded versions of these regular datasets and it has been pre-cleaned by my dissertation supervisor and have dropped all the columns apart from Paper ID and Paper Abstract. The rows with Null values have also been cleaned so we have a dataset with just 2 columns and almost 60,000 rows all related to COVID-19 research papers.

7. Approach

To meet certain expectations and have this application functioning properly, we need to make sure that a number of criteria are set out to be implemented. These criteria need to be predefined to be well executed. They can be divided into Functional and Non-Functional Requirements as below.

Functional Requirements:

The application must have a capability to take an Abstract ID as an input from the user. This Abstract ID should be used to Vectorize and calculate the similarity with other Abstracts in the dataset.

1. The application should be able to use the provided dataset in .tsv format.
2. The application should be able to process the whole dataset and vectorize it, using either TF-IDF or Count Vectorizers.
3. The application should be able to provide the abstract for the ID provided.
4. The application should use the vectors to convert the data into cosine similarity matrices.

When the Web Framework initializes, it should be able to:

1. Load the web templates successfully on the dedicated IP and port.
2. Collect the necessary parameters that are entered by the user and returned by the html page.
3. Pass the values and necessary parameters to the respective functions to find the abstract value and the recommendations list.
4. Return the outputs to the results page template.

Useful to have (optional) features –

- Be able to return a relevancy score along with the similar abstracts.
- An interactive responsive web-design to make the application look nice.

- A SQL database to store the data and allow the user to query.

Non-functional requirements –

1. Usability – A GUI (Graphical user interface) should be implemented for a user to easily use the application.
2. Performance – The application should collect the data, pre-process and process it fast and efficiently using minimal time and resources.

My approach –

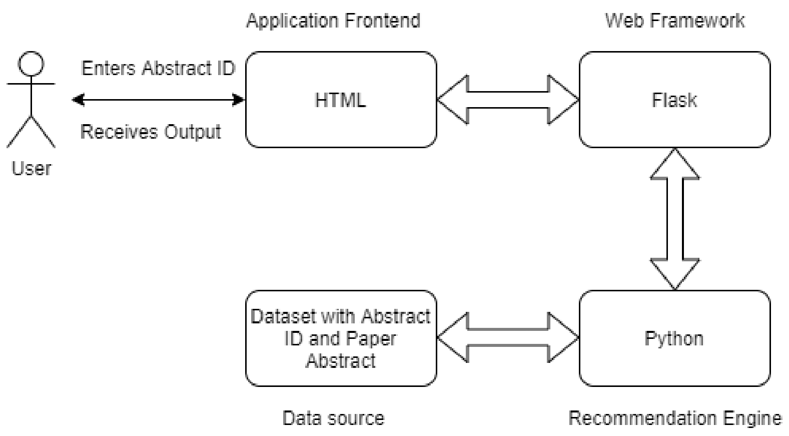


Figure 9. Basic Application architecture.

To meet the specified requirements, the application will be developed using Flask Web Framework and Python Recommendation Engine. To understand this workflow, Figure 9 can be referred to.

The user will be allowed to enter a valid input using an HTML frontend. This HTML frontend will interact with Flask framework to allow the input to be processed in the recommendation engine which is Python. The recommendation engine will process this input and return a list of the top 10 similar abstracts after comparing with the rest of the abstracts in the dataset. This returned list will be published on another HTML template through Flask.

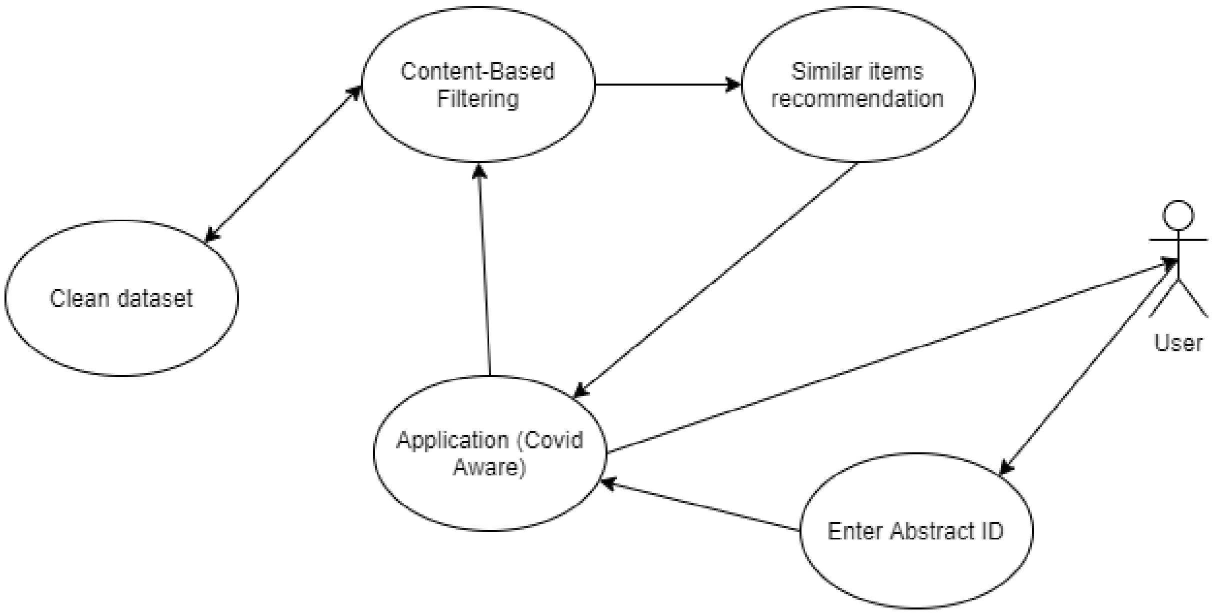


Figure 10. Application Workflow

Figure 10 represents the application workflow of how the whole application works and how it interacts with data. The abstract ID will be passed from the user to the application. This data will be compared and filtered while accessing the data source. The items which are found similar will be returned to the user.

8. Implementation

This prototype application has been implemented using HTML, Flask and Python technologies. For the ease of understanding, the explanation is divided into Frontend and Backend categories.

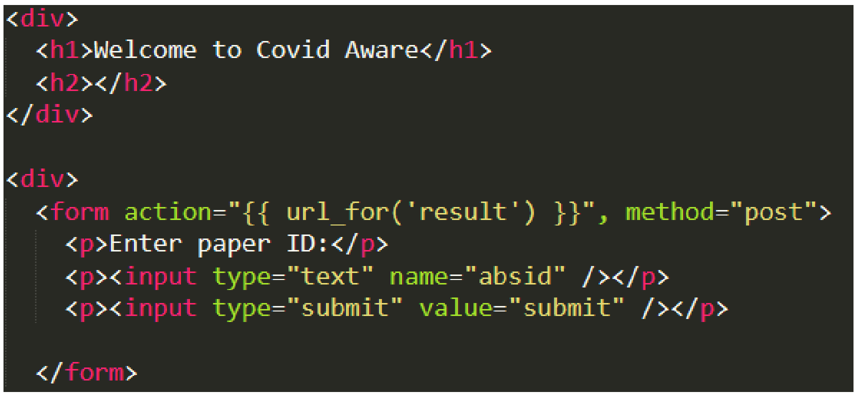
- Frontend – All the HTML web pages which are mainly responsible for the graphical user interface for data collection and for representing the values returned after processing of the data.
- Backend – This consists of Flask and Python, which are responsible for processing the collected data, access the data source, process for similar items and return the recommendations to the Frontend.

Frontend –

The Frontend is responsible for making an environment available to the user to be able to view the interface and input their Abstract IDs. It is also responsible to display the paper abstract and their relevant paper abstracts to the user after they search for one. To achieve this, the Frontend uses two different templates.

Data collection template –

A separate home.html file is used to create a basic user interface in order to allow the user to enter the specific Abstract ID to search for.



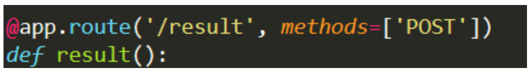
```
<div>
  <h1>Welcome to Covid Aware</h1>
  <h2></h2>
</div>

<div>
  <form action="{{ url_for('result') }}", method="post">
    <p>Enter paper ID:</p>
    <p><input type="text" name="absid" /></p>
    <p><input type="submit" value="submit" /></p>

  </form>
```

Figure 11. home.HTML file for data collection.

In Figure 11, the h1 tag represents the heading we can see on our page. Then, I have created a form for accepting user input which uses a post method to send the data to Flask. The 'action' parameter in the form tag instructs the next action to be done which in this case is to send the data to a result page in Flask. Figure 12 shows in short how it is defined in Flask. This result page allows POST method as it can be seen. However, the Flask side of things will be discussed in detail in the next section.



```
@app.route('/result', methods=['POST'])
def result():
```

Figure 12. Flask code reference for 'result' page.

The result of this basic HTML template which is the home page of the application can be seen in Figure 13.

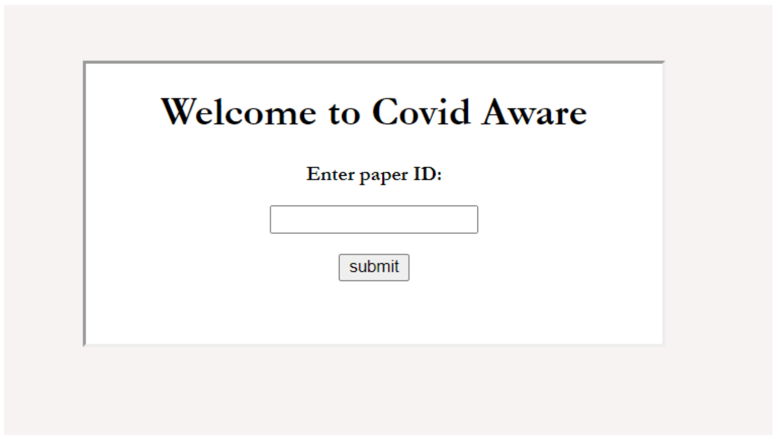
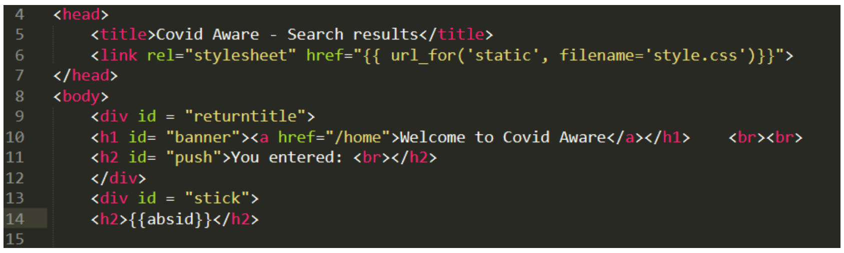


Figure 13. Home page of Covid Aware

Result display template –

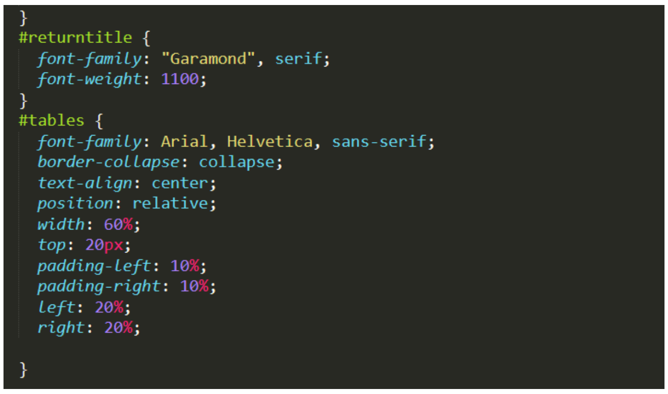
This template has the primary function to display the results returned by Flask which would be in the form of variables.



```
4   <head>
5       <title>Covid Aware - Search results</title>
6       <link rel="stylesheet" href="{{ url_for('static', filename='style.css')}}">
7   </head>
8   <body>
9       <div id = "returntitle">
10      <h1 id= "banner"><a href="/home">Welcome to Covid Aware</a></h1>    <br><br>
11      <h2 id= "push">You entered: <br></h2>
12      </div>
13      <div id = "stick">
14      <h2>{{absid}}</h2>
15
```

Figure 14. Top section of the results landing page.

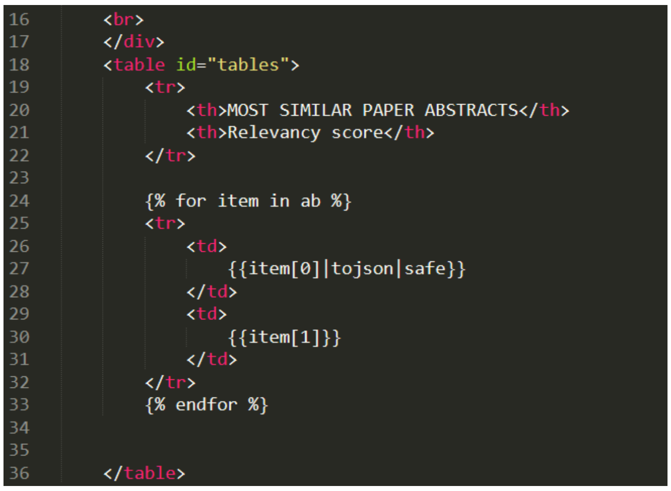
Figure 14 above represents the top section of the results page. This is the page returned by the 'result' action mentioned above. The head section in this code shows us the title of the html web page. In line 6, we are linking a CSS stylesheet which allows us to separately configure some styling to our HTML pages. Lines apart from line 14 represent some static data which will not change based on the input. However, in line 14, 2 curly braces can be noticed. These brackets are used to fetch input passed by Flask. The {{absid}} parameter here is a variable from Flask which contains the returned Paper Abstract for the Abstract ID entered by the user. An 'id' parameter to a few tags can be noticed in the following lines in the body section. These parameters are used to reference customisation in the CSS stylesheets. An example of this is shown in Figure 15.



```
}
#returntitle {
  font-family: "Garamond", serif;
  font-weight: 1100;
}
#tables {
  font-family: Arial, Helvetica, sans-serif;
  border-collapse: collapse;
  text-align: center;
  position: relative;
  width: 60%;
  top: 20px;
  padding-left: 10%;
  padding-right: 10%;
  left: 20%;
  right: 20%;

}
```

Figure 15. CSS styling example.



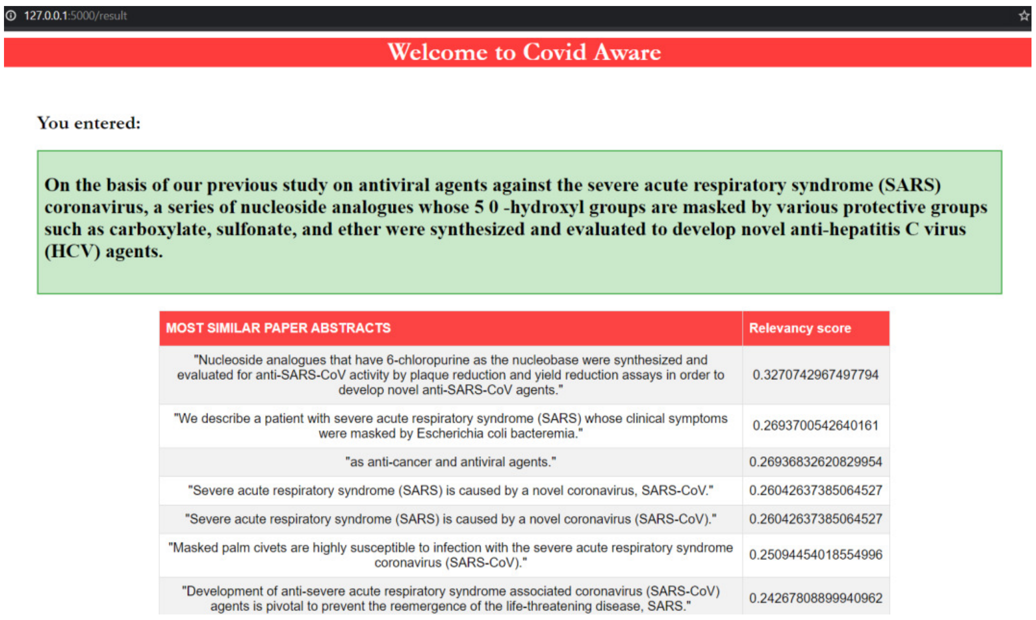
```
        <br>
        </div>
        <table id="tables">
            <tr>
                <th>MOST SIMILAR PAPER ABSTRACTS</th>
                <th>Relevancy score</th>
            </tr>

            {% for item in ab %}
            <tr>
                <td>
                    {{item[0]|tojson|safe}}
                </td>
                <td>
                    {{item[1]}}
                </td>
            </tr>
            {% endfor %}


        </table>
```

Figure 16. Bottom section of the results landing page.

Figure 16 above represents how the relevant paper abstracts are displayed on the results web page. A table has been created with headings ‘Most similar paper abstracts’ and ‘Relevancy Scores’ which will have the similar abstracts and score in their respective columns. To add the data to these columns which is received in a single variable in the form of an array, a ‘for’ loop has been implemented to fetch a single tuple from the array and populate the table one by one. An example of the output (input abstract ID 5b889b5e979a0b32228441c2fb350b5357134712) for the results page can be seen in Figure 17.



127.0.0.1:5000/result

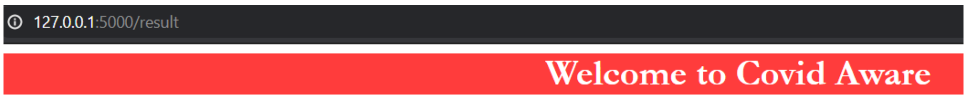
Welcome to Covid Aware

You entered:

On the basis of our previous study on antiviral agents against the severe acute respiratory syndrome (SARS) coronavirus, a series of nucleoside analogues whose 5 0 -hydroxyl groups are masked by various protective groups such as carboxylate, sulfonate, and ether were synthesized and evaluated to develop novel anti-hepatitis C virus (HCV) agents.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Nucleoside analogues that have 6-chloropurine as the nucleobase were synthesized and evaluated for anti-SARS-CoV activity by plaque reduction and yield reduction assays in order to develop novel anti-SARS-CoV agents."	0.3270742967497794
"We describe a patient with severe acute respiratory syndrome (SARS) whose clinical symptoms were masked by Escherichia coli bacteremia."	0.2693700542640161
"as anti-cancer and antiviral agents."	0.26936832620829954
"Severe acute respiratory syndrome (SARS) is caused by a novel coronavirus, SARS-CoV."	0.26042637385064527
"Severe acute respiratory syndrome (SARS) is caused by a novel coronavirus (SARS-CoV)."	0.26042637385064527
"Masked palm civets are highly susceptible to infection with the severe acute respiratory syndrome coronavirus (SARS-CoV)."	0.25094454018554996
"Development of anti-severe acute respiratory syndrome associated coronavirus (SARS-CoV) agents is pivotal to prevent the reemergence of the life-threatening disease, SARS."	0.24267808899940962

Figure 17. Result page



127.0.0.1:5000/result

Welcome to Covid Aware

Figure 18. Result page URL.

From figure 18 above, it can be noticed that the returned URL contains '/result' appended to the URL. It denotes that this is the result page of the application. The page is divided into 2 sections which display the abstract and a table for relevant abstracts and the relevancy score.

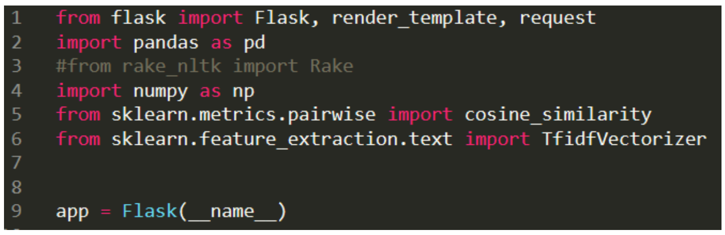
Backend –

The Backend side of the code contains the Flask Web Framework and the NLP (Natural Language Processing) based Recommendation Engine written in Python.

Flask –

Flask is our web framework used to develop this application. To start with, Flask provides a minimal application template which starts up as a basic Hello World program.

For my implementation, I have started with the minimal code and developed on it. As it can be seen in Figure 19 below, in line 1, Flask class has been imported which allows us to create an instance of it. This instance becomes our Web Server Gateway Interface (WSGI) application.



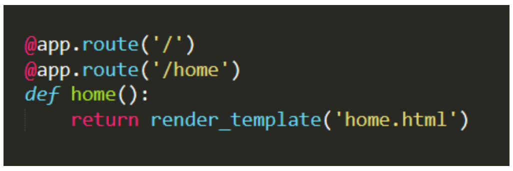
```
1  from flask import Flask, render_template, request
2  import pandas as pd
3  #from rake_nltk import Rake
4  import numpy as np
5  from sklearn.metrics.pairwise import cosine_similarity
6  from sklearn.feature_extraction.text import TfidfVectorizer
7
8
9  app = Flask(__name__)
```

Figure 19. Flask import.

Apart from importing Flask, I have also imported render_template and request. Render_template is used to render a defined web page in response to a triggered URL. The 'request' subclass allows Flask to collect the user input from the rendered HTML template.

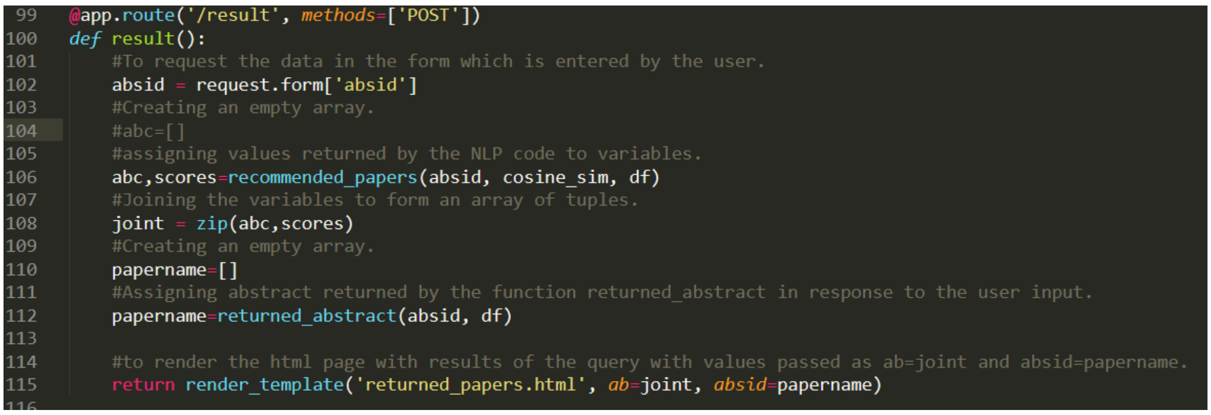
In line 9, I have created an instance of the Flask class which is our application. The name of the application here is '__name__'. This is always the application name if the application uses a single module. This name is used to find resources on the filesystem (Flask, 2020). The flask object acts as a central registry for URL rules, template configuration, view functions, etc.

Then we have defined an app.route for the home page. Two values have been passed to the app.route here because it indicates if either of the default route ('/') or home is requested, the same function will be triggered. We can see in the Figure 20 below, that whether either of these are resolved, the function home() will be triggered, and it will render a html page which is 'home.html'. Therefore, when referred back to the html home page, it will be resolved when this function is triggered.



```python
@app.route('/')
@app.route('/home')
def home():
    return render_template('home.html')
```

Figure 20. route decorator in Flask.



```python
@app.route('/result', methods=['POST'])
def result():
    #To request the data in the form which is entered by the user.
    absid = request.form['absid']
    #Creating an empty array.
    #abc=[]
    #assigning values returned by the NLP code to variables.
    abc,scores=recommended_papers(absid, cosine_sim, df)
    #Joining the variables to form an array of tuples.
    joint = zip(abc,scores)
    #Creating an empty array.
    papername=[]
    #Assigning abstract returned by the function returned_abstract in response to the user input.
    papername=returned_abstract(absid, df)

    #to render the html page with results of the query with values passed as ab=joint and absid=papername.
    return render_template('returned_papers.html', ab=joint, absid=papername)
```

Figure 21. route for results.

Referring to the Figure 21, the app.route here resolves the URL '/result' and triggers a function called result(). This function is an important aspect of our application.

In line 102 - the value which was collected by the user's input was stored under the name 'absid', This value is being fetched using a request method which allows us to retrieve the form data as the form in our home page had delivered these values to the result route. The values returned by the NLP code have been assigned to variables, where the required values parameters are passed to the function in line 106. Then these 2 variables are joined together to be sent to the results page in one array.

Another variable (papername) is declared to store the paper abstract for the returned Abstract ID. Required parameters are passed. In line 115, the html page named 'returned_papers.html' is rendered with the 'joint' and 'papername' variables passed as 'ab' and 'absid'.

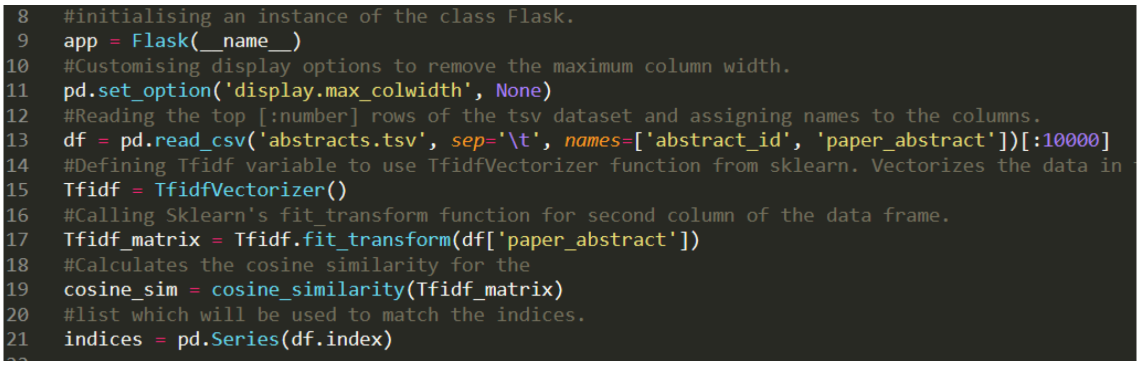
This covers the Flask and the webpage side of events. But the NLP section is the heart of the recommender system as it calculates the similarities and relevancy. The NLP section contains the logic to find the cosine similarity using either Tfidf Vectorizer or Count Vectorizer.

To find the cosine similarity –

Primarily, the column width has been changed to none so that the whole abstract in the column can be viewed without being limited to a predefined number of characters. Pandas has been imported as 'pd'. Hence, it can be seen that set_option is a pandas function.

In line 13 (refer Figure 22), the dataset in tsv format is assigned to the dataframe df and names have been assigned to the respective columns. In this example, I am only reading top 10000 rows for development purposes as it is faster to process lesser rows, making less matrices.

The next step is to use the Vectorizer. Since many articles have used CountVectorizers for recommendation systems, for testing purposes I have implemented this using both Tfidf Vectorizer and Count Vectorizer to see which one was giving better results.



```
#initialising an instance of the class Flask.
app = Flask(__name__)
#Customising display options to remove the maximum column width.
pd.set_option('display.max_colwidth', None)
#Reading the top [:number] rows of the tsv dataset and assigning names to the columns.
df = pd.read_csv('abstracts.tsv', sep='\t', names=['abstract_id', 'paper_abstract'])[:10000]
#Defining Tfidf variable to use TfidfVectorizer function from sklearn. Vectorizes the data in
Tfidf = TfidfVectorizer()
#Calling Sklearn's fit_transform function for second column of the data frame.
Tfidf_matrix = Tfidf.fit_transform(df['paper_abstract'])
#Calculates the cosine similarity for the
cosine_sim = cosine_similarity(Tfidf_matrix)
#list which will be used to match the indices.
indices = pd.Series(df.index)
```

Figure 22. TFIDF Vectorization and Cosine similarity.

After Vectorizing, fit transform function is used. This function basically implies fit first then transform function but done more efficiently when passed together as fit_transform. The fit function learns the vocabulary and IDF from the dataset and the transform function transforms the documents to document-term matrix. It uses the document frequencies learned by fit. I have passed just the paper_abstract column of the df, hence it only fits and transforms this column.

The Tfidf_matrix is then used to create a cosine similarity matrix by using the cosine_similarity function by sklearn. Cosine similarity is the metric we are using to see how similar two abstracts are. The cosine_sim is a numpy array containing computed cosine similarity between all abstracts. Since they are all similar to themselves, we expect a diagonal full of 1s.

The cosine similarity is found by the equation as discussed earlier.

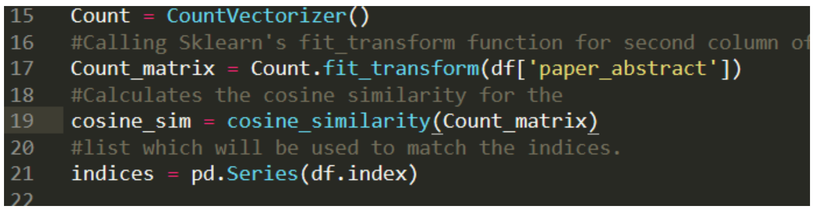
$$similarity = \cos(\theta) = \frac{a.b}{\|a\| \times \|b\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} {a_i}^2}\sqrt{\sum_{i=1}^{n} {b_i}^2}}$$

The similarity matrix when calculated returns a score matrix, but since the vectors (in our case Tfidf_matrix) passed as input are both the same, the matrix will have a diagonal of ones. For example - The cosine similarity matrix for 4 rows would look as something like this –

$$\begin{bmatrix} 1 & 0.139 & 0.302 & 0.246 \\ 0.139 & 1 & 0.431 & 0.455 \\ 0.302 & 0.431 & 1 & 0.046 \\ 0.246 & 0.455 & 0.046 & 1 \end{bmatrix}$$

This is because we are passing only one column from df to calculate the similarity and they are 100% similar to themselves. It can also be observed that Row 1 of Column 0, and Column 1 of Row 0 also have the same results.

Similar results can be achieved using Count Vectorizer. The only difference in the implementation would be to use Count Vectorizer instead of Tfidf Vectorizer as shown in Figure 23 below. The difference between both approaches is minimal as already explained. However, as Tfidf Vectorizer gives more importance to less repetitive words, the results seen are much more accurate than Count Vectorizer. Therefore, I have decided to implement Tfidf Vectorizer. A comparison between the two has been shown in the Analysis section.

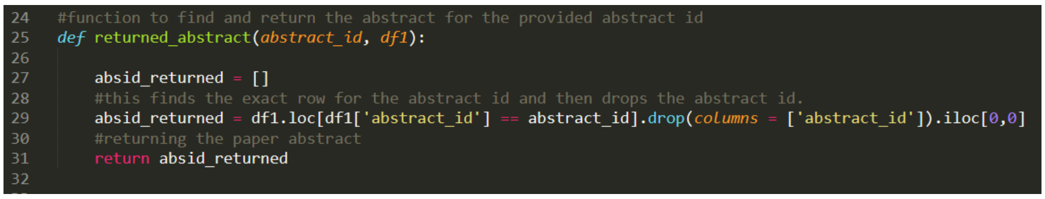


```python
Count = CountVectorizer()
#Calling Sklearn's fit_transform function for second column of
Count_matrix = Count.fit_transform(df['paper_abstract'])
#Calculates the cosine similarity for the
cosine_sim = cosine_similarity(Count_matrix)
#list which will be used to match the indices.
indices = pd.Series(df.index)
```

Figure 23. Using Count Vectorizer.

Function to return the Abstract –

In order to return the Abstract corresponding to the input Abstract IDs, the application uses a function called returned_abstract which can be seen in Figure 24 below. The absid_returned numpy array has been defined to select only the row which matches the abstract ID and then the column with abstract ID is dropped. This leaves us with the expected paper abstract. This array is returned to Flask to display on the html page as already discussed.



```python
#function to find and return the abstract for the provided abstract id
def returned_abstract(abstract_id, df1):

    absid_returned = []
    #this finds the exact row for the abstract id and then drops the abstract id.
    absid_returned = df1.loc[df1['abstract_id'] == abstract_id].drop(columns = ['abstract_id']).iloc[0,0]
    #returning the paper abstract
    return absid_returned
```

Figure 24. To Return Paper Abstract.

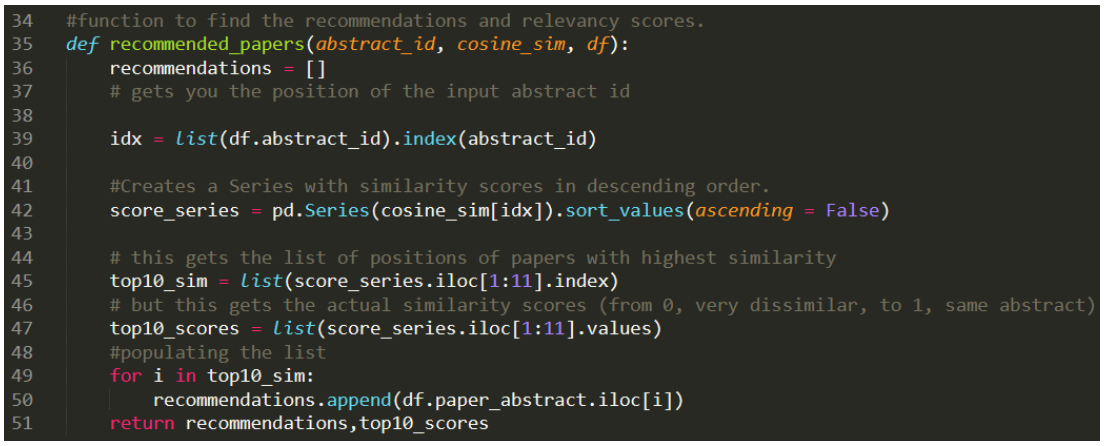
Function to recommend papers and relevancy scores –

This is the core function of our recommendation system which, when called with input parameters, will return a list of recommendations and relevancy scores.

Here idx contains the index / position of the input Abstract ID. Then, in line 42 of Figure 25 below, a series of scores is created to store the scores obtained by the cosine similarity for the index value of the input Abstract ID. These are sorted in descending order (highest value first).

The list of similarity scores is ready, so all that is required now is to list the corresponding abstracts. To achieve this, I have 2 lists at line 45 and 47, which create a list of positions of score_series (or the papers with the highest similarity). Line 47 creates a similar list but with similarity scores instead of their positions. Both these variables contain only the top 10 values as we will be displaying only the top relevant papers on our Application.

Finally, a for loop is used to populate the list by actual Abstracts corresponding to the positions in 'top10_sim' list.

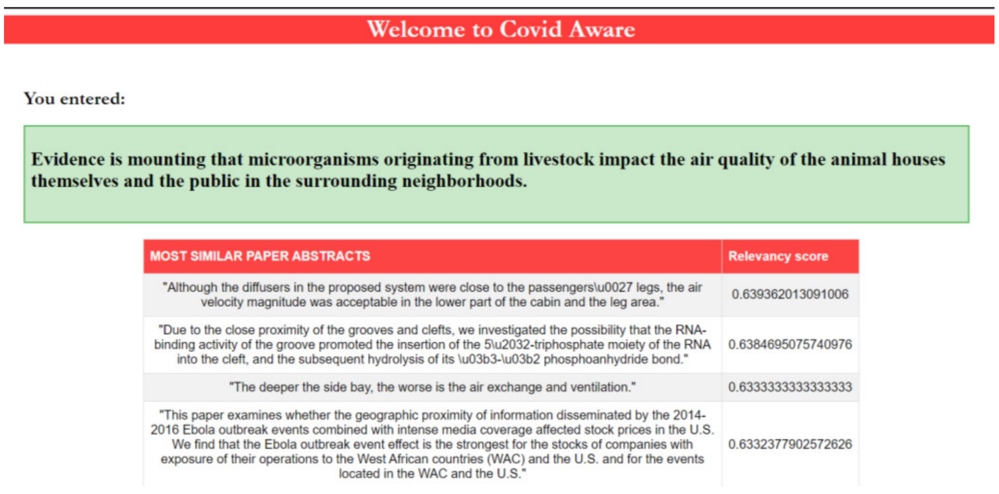


```
34  #function to find the recommendations and relevancy scores.
35  def recommended_papers(abstract_id, cosine_sim, df):
36      recommendations = []
37      # gets you the position of the input abstract id
38  
39      idx = list(df.abstract_id).index(abstract_id)
40  
41      #Creates a Series with similarity scores in descending order.
42      score_series = pd.Series(cosine_sim[idx]).sort_values(ascending = False)
43  
44      # this gets the list of positions of papers with highest similarity
45      top10_sim = list(score_series.iloc[1:11].index)
46      # but this gets the actual similarity scores (from 0, very dissimilar, to 1, same abstract)
47      top10_scores = list(score_series.iloc[1:11].values)
48      #populating the list
49      for i in top10_sim:
50          recommendations.append(df.paper_abstract.iloc[i])
51      return recommendations,top10_scores
```

Figure 25. Recommendation function.

9. Analysis

To compare the output of the application with both Count and TFIDF Vectorizers, I attempted to manually observe which of these methods return the most relevant abstracts as there is no definite way to compare the outputs. Hence, in my first example, as it can be seen in the Figure 26 below, that Count Vectorizer method has returned a few similar abstracts, but they are irrelevant to the input abstract. The input abstract has keywords like “microorganisms”, “livestock”, etc. But the most relevant paper talks about something completely irrelevant. Whereas in the TFIDF Vectorizer, as it can be seen in the Figure 27, has returned a document with 2 matching keywords which are “livestock” and “animal”. This makes it comparatively more relevant to what Count Vectorizer had returned. The snapshots have been cropped to show only the first 4 results on each page as they are sufficient for a comparison.



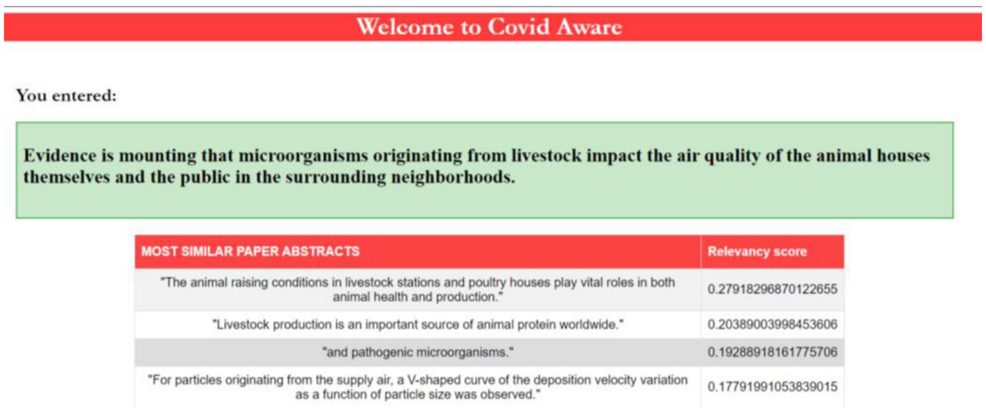
Welcome to Covid Aware

You entered:

Evidence is mounting that microorganisms originating from livestock impact the air quality of the animal houses themselves and the public in the surrounding neighborhoods.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Although the diffusers in the proposed system were close to the passengers\u0027 legs, the air velocity magnitude was acceptable in the lower part of the cabin and the leg area."	0.639362013091006
"Due to the close proximity of the grooves and clefts, we investigated the possibility that the RNA-binding activity of the groove promoted the insertion of the 5\u2032-triphosphate moiety of the RNA into the cleft, and the subsequent hydrolysis of its \u03b3-\u03b2 phosphoanhydride bond."	0.6384695075740976
"The deeper the side bay, the worse is the air exchange and ventilation."	0.6333333333333333
"This paper examines whether the geographic proximity of information disseminated by the 2014-2016 Ebola outbreak events combined with intense media coverage affected stock prices in the U.S. We find that the Ebola outbreak event effect is the strongest for the stocks of companies with exposure of their operations to the West African countries (WAC) and the U.S. and for the events located in the WAC and the U.S."	0.6332377902572626

Figure 26. Count Vectorizer Example 1.



Welcome to Covid Aware

You entered:

Evidence is mounting that microorganisms originating from livestock impact the air quality of the animal houses themselves and the public in the surrounding neighborhoods.

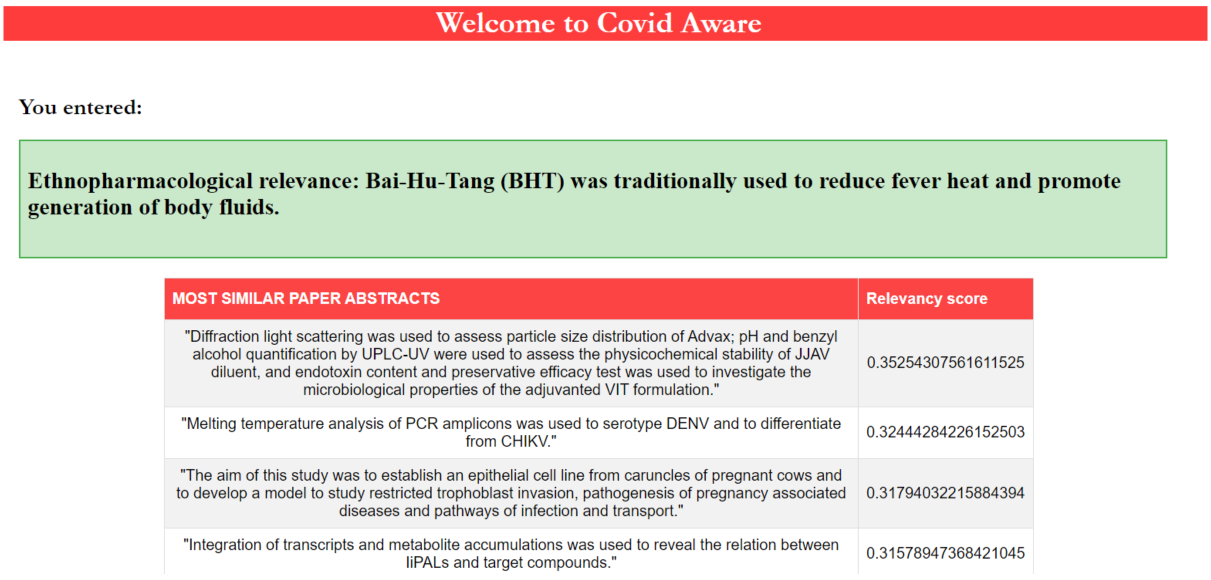
MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"The animal raising conditions in livestock stations and poultry houses play vital roles in both animal health and production."	0.27918296870122655
"Livestock production is an important source of animal protein worldwide."	0.20389003998453606
"and pathogenic microorganisms."	0.19288918161775706
"For particles originating from the supply air, a V-shaped curve of the deposition velocity variation as a function of particle size was observed."	0.17791991053839015

Figure 27. TFIDF Vectorizer Example 1.

To confirm this observation, I tried passing another abstract ID in example 2 (refer Figures 28 and 29).

Count Vectorizer – Does not seem to have any important repetitive words. The results are based rather on the comparison of words like 'was' and 'to'.

TFIDF Vectorizer – The first relevant abstract itself returns the core words of the abstract which are "Ethnopharmacological relevance", making it a much more relevant and effective system in our case.



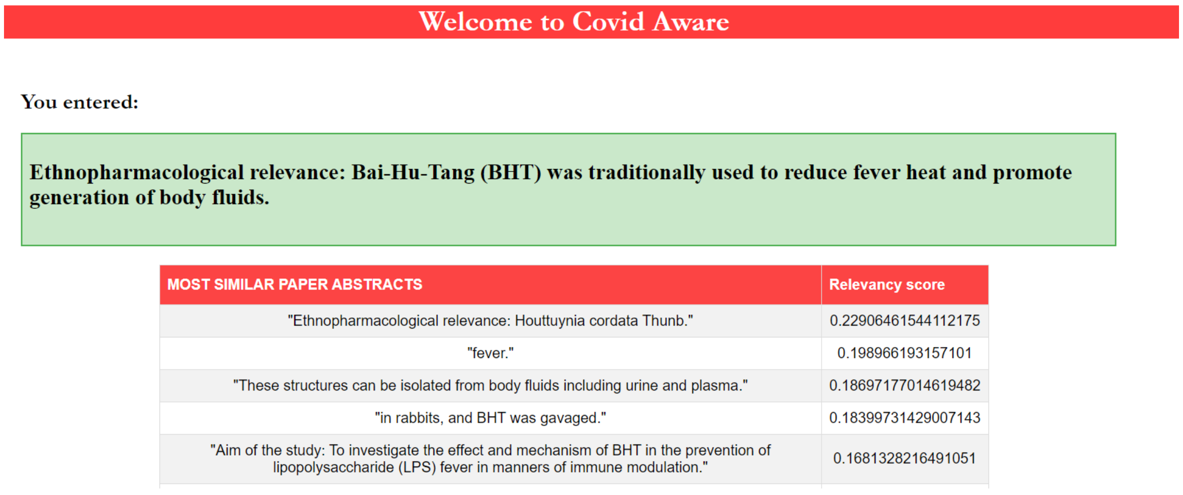
Welcome to Covid Aware

You entered:

Ethnopharmacological relevance: Bai-Hu-Tang (BHT) was traditionally used to reduce fever heat and promote generation of body fluids.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Diffraction light scattering was used to assess particle size distribution of Advax; pH and benzyl alcohol quantification by UPLC-UV were used to assess the physicochemical stability of JJAV diluent, and endotoxin content and preservative efficacy test was used to investigate the microbiological properties of the adjuvanted VIT formulation."	0.35254307561611525
"Melting temperature analysis of PCR amplicons was used to serotype DENV and to differentiate from CHIKV."	0.32444284226152503
"The aim of this study was to establish an epithelial cell line from caruncles of pregnant cows and to develop a model to study restricted trophoblast invasion, pathogenesis of pregnancy associated diseases and pathways of infection and transport."	0.31794032215884394
"Integration of transcripts and metabolite accumulations was used to reveal the relation between liPALs and target compounds."	0.31578947368421045

Figure 28. Count Vectorizer Example 2.


Welcome to Covid Aware

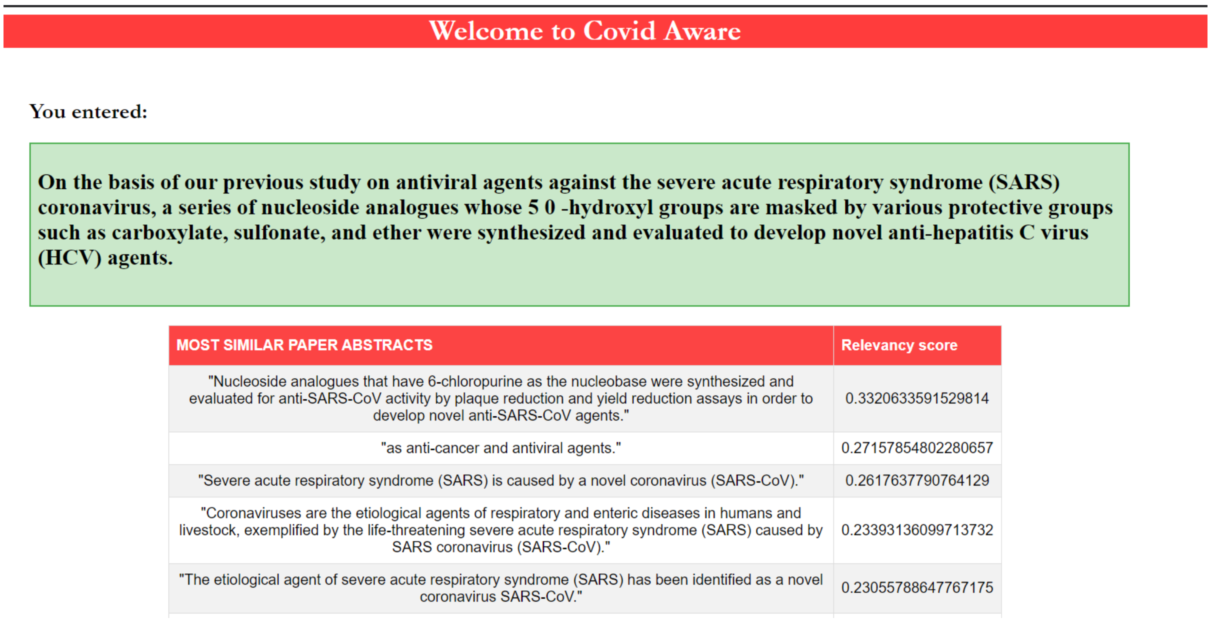
You entered:

Ethnopharmacological relevance: Bai-Hu-Tang (BHT) was traditionally used to reduce fever heat and promote generation of body fluids.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Ethnopharmacological relevance: Houttuynia cordata Thunb."	0.22906461544112175
"fever."	0.198966193157101
"These structures can be isolated from body fluids including urine and plasma."	0.18697177014619482
"in rabbits, and BHT was gavaged."	0.18399731429007143
"Aim of the study: To investigate the effect and mechanism of BHT in the prevention of lipopolysaccharide (LPS) fever in manners of immune modulation."	0.1681328216491051

Figure 29. TFIDF Vectorizer Example 2.

However, there have been examples where both the methods have seemed to perform well. Referring to Figure 30, an Abstract ID was entered for the third test where TFIDF has found similar abstracts related to keywords like “SARS” and nucleoside analogues. Whereas, in reference to Figure 31, CountVectorizer has also returned few results related to “SARS”. Hence, it can be seen that CountVectorizer provides better results when a large text input has been passed.

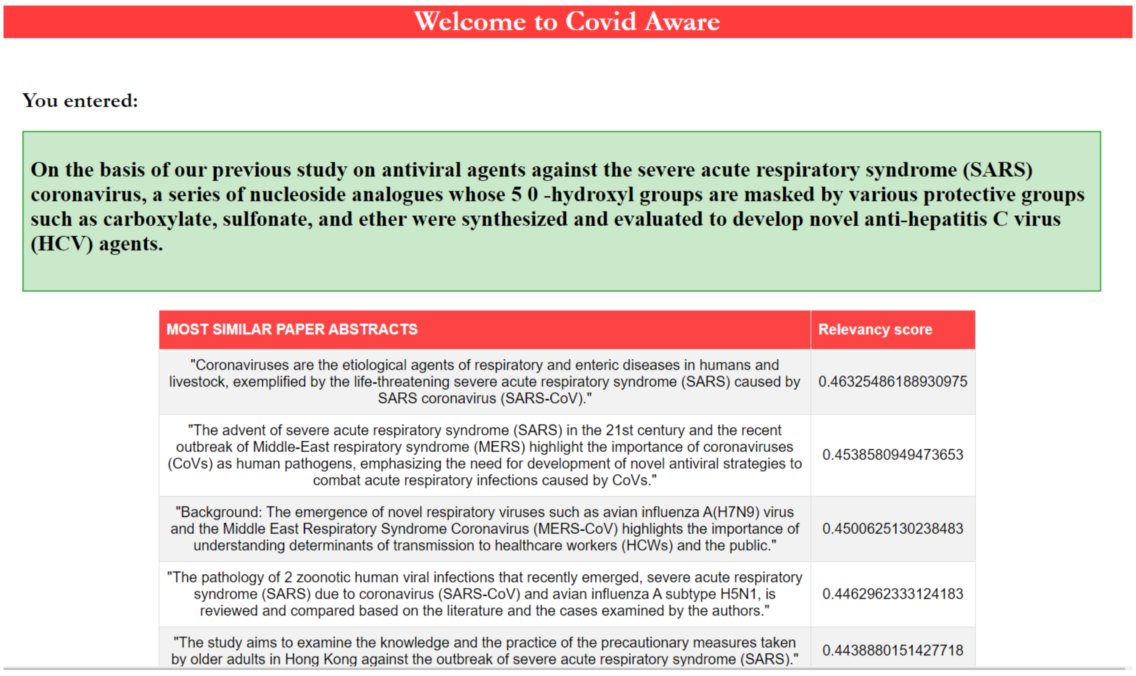

Welcome to Covid Aware

You entered:

On the basis of our previous study on antiviral agents against the severe acute respiratory syndrome (SARS) coronavirus, a series of nucleoside analogues whose 5 0 -hydroxyl groups are masked by various protective groups such as carboxylate, sulfonate, and ether were synthesized and evaluated to develop novel anti-hepatitis C virus (HCV) agents.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Nucleoside analogues that have 6-chloropurine as the nucleobase were synthesized and evaluated for anti-SARS-CoV activity by plaque reduction and yield reduction assays in order to develop novel anti-SARS-CoV agents."	0.3320633591529814
"as anti-cancer and antiviral agents."	0.27157854802280657
"Severe acute respiratory syndrome (SARS) is caused by a novel coronavirus (SARS-CoV)."	0.2617637790764129
"Coronaviruses are the etiological agents of respiratory and enteric diseases in humans and livestock, exemplified by the life-threatening severe acute respiratory syndrome (SARS) caused by SARS coronavirus (SARS-CoV)."	0.23393136099713732
"The etiological agent of severe acute respiratory syndrome (SARS) has been identified as a novel coronavirus SARS-CoV."	0.23055788647767175

Figure 30. TFIDF Vectorizer Example 3.



Welcome to Covid Aware

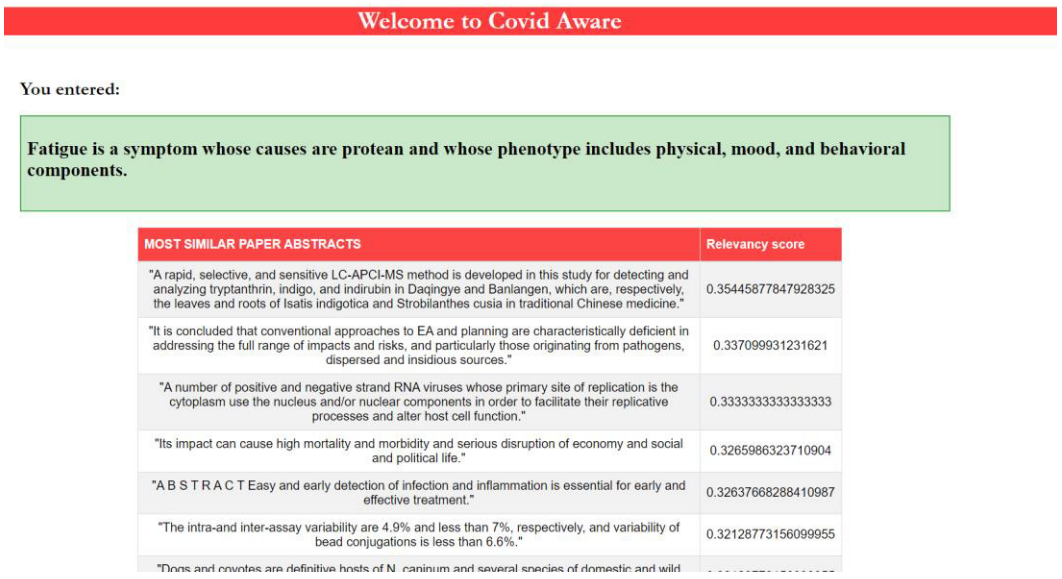
You entered:

On the basis of our previous study on antiviral agents against the severe acute respiratory syndrome (SARS) coronavirus, a series of nucleoside analogues whose 5 0 -hydroxyl groups are masked by various protective groups such as carboxylate, sulfonate, and ether were synthesized and evaluated to develop novel anti-hepatitis C virus (HCV) agents.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Coronaviruses are the etiological agents of respiratory and enteric diseases in humans and livestock, exemplified by the life-threatening severe acute respiratory syndrome (SARS) caused by SARS coronavirus (SARS-CoV)."	0.46325486188930975
"The advent of severe acute respiratory syndrome (SARS) in the 21st century and the recent outbreak of Middle-East respiratory syndrome (MERS) highlight the importance of coronaviruses (CoVs) as human pathogens, emphasizing the need for development of novel antiviral strategies to combat acute respiratory infections caused by CoVs."	0.4538580949473653
"Background: The emergence of novel respiratory viruses such as avian influenza A(H7N9) virus and the Middle East Respiratory Syndrome Coronavirus (MERS-CoV) highlights the importance of understanding determinants of transmission to healthcare workers (HCWs) and the public."	0.4500625130238483
"The pathology of 2 zoonotic human viral infections that recently emerged, severe acute respiratory syndrome (SARS) due to coronavirus (SARS-CoV) and avian influenza A subtype H5N1, is reviewed and compared based on the literature and the cases examined by the authors."	0.4462962333124183
"The study aims to examine the knowledge and the practice of the precautionary measures taken by older adults in Hong Kong against the outbreak of severe acute respiratory syndrome (SARS)."	0.4438880151427718

Figure 31. Count Vectorizer Example 3.

Finally, in rare cases like example 4 (Figure 32 and 33), it can be observed that the abstract talks about something different but neither of the Vectorizers have been able to find a relevant research paper. This could either be because of lack of similar research papers in the dataset or due to prospect for improvement in the methods.



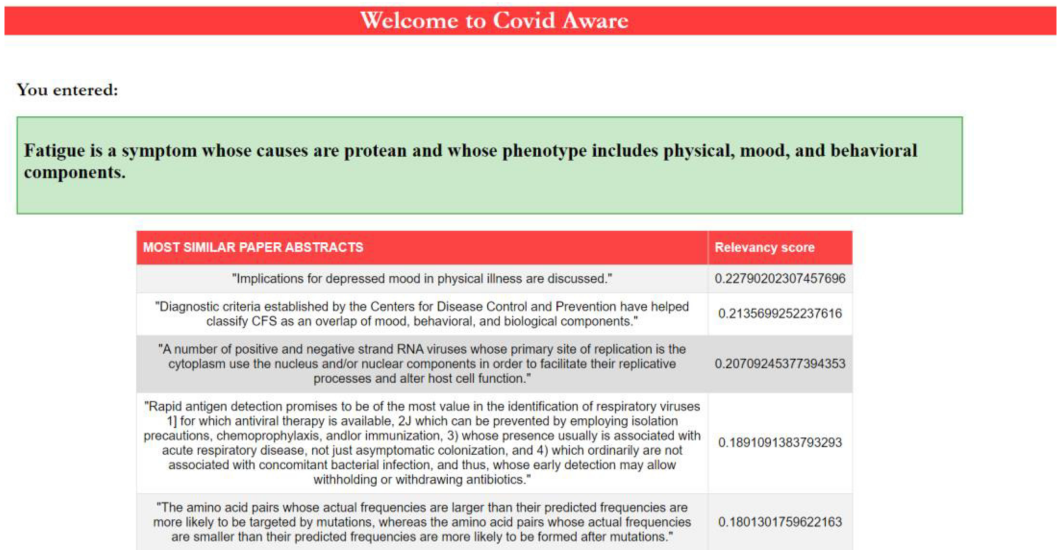
Welcome to Covid Aware

You entered:

Fatigue is a symptom whose causes are protean and whose phenotype includes physical, mood, and behavioral components.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"A rapid, selective, and sensitive LC-APCI-MS method is developed in this study for detecting and analyzing tryptanthrin, indigo, and indirubin in Daqingye and Banlangen, which are, respectively, the leaves and roots of Isatis indigotica and Strobilanthes cusia in traditional Chinese medicine."	0.35445877847928325
"It is concluded that conventional approaches to EA and planning are characteristically deficient in addressing the full range of impacts and risks, and particularly those originating from pathogens, dispersed and insidious sources."	0.337099931231621
"A number of positive and negative strand RNA viruses whose primary site of replication is the cytoplasm use the nucleus and/or nuclear components in order to facilitate their replicative processes and alter host cell function."	0.3333333333333333
"Its impact can cause high mortality and morbidity and serious disruption of economy and social and political life."	0.3265986323710904
"A B S T R A C T Easy and early detection of infection and inflammation is essential for early and effective treatment."	0.32637668288410987
"The intra-and inter-assay variability are 4.9% and less than 7%, respectively, and variability of bead conjugations is less than 6.6%."	0.32128773156099955
"Dogs and coyotes are definitive hosts of N. caninum and several species of domestic and wild	[illegible]

Figure 32. Count Vectorizer Example 4.


Welcome to Covid Aware

You entered:

Fatigue is a symptom whose causes are protean and whose phenotype includes physical, mood, and behavioral components.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Implications for depressed mood in physical illness are discussed."	0.22790202307457696
"Diagnostic criteria established by the Centers for Disease Control and Prevention have helped classify CFS as an overlap of mood, behavioral, and biological components."	0.2135699252237616
"A number of positive and negative strand RNA viruses whose primary site of replication is the cytoplasm use the nucleus and/or nuclear components in order to facilitate their replicative processes and alter host cell function."	0.20709245377394353
"Rapid antigen detection promises to be of the most value in the identification of respiratory viruses 1] for which antiviral therapy is available, 2J which can be prevented by employing isolation precautions, chemoprophylaxis, andlor immunization, 3) whose presence usually is associated with acute respiratory disease, not just asymptomatic colonization, and 4) which ordinarily are not associated with concomitant bacterial infection, and thus, whose early detection may allow withholding or withdrawing antibiotics."	0.1891091383793293
"The amino acid pairs whose actual frequencies are larger than their predicted frequencies are more likely to be targeted by mutations, whereas the amino acid pairs whose actual frequencies are smaller than their predicted frequencies are more likely to be formed after mutations."	0.1801301759622163

Figure 33. Tfidf Vectorizer Example 4.

The basic comparative analysis check to compare the relevancy for both methods can be summarised as -

	Example 1	**Example 2**	**Example 3**	**Example 4**
TFIDF Vectorizer	Relevant	Relevant	Relevant	Irrelevant
Count Vectorizer	Irrelevant	Irrelevant	Relevant	Irrelevant

Table - Test analysis for both Vectorizers.

Recommender tool performance

The tool's performance is obviously a factor of major concern. It has also been observed that the tool's performance has differed according to the conditions and dataset.

To test the tool's performance, the dataset was limited to different number of rows throughout the testing. The results of which can be seen in Table 2.

This analysis is based on a computer with following hardware –

Processor - i7-9750H 2.6Hz – 4.5Ghz, 6 cores, 12 LP.

RAM – 16GB DDR4 1330MHz

Number of rows loaded	Time Required to initialise the app	Response time to recommend papers after initialisation
1000 rows	3 seconds	<1 second
10000 rows	6 Seconds	<1 second
30000 rows	136 seconds	<1 second
45000 rows	~160 seconds (Failed sometimes)	<1 second
60000 rows	Failed. (requires higher RAM)	Not Tested

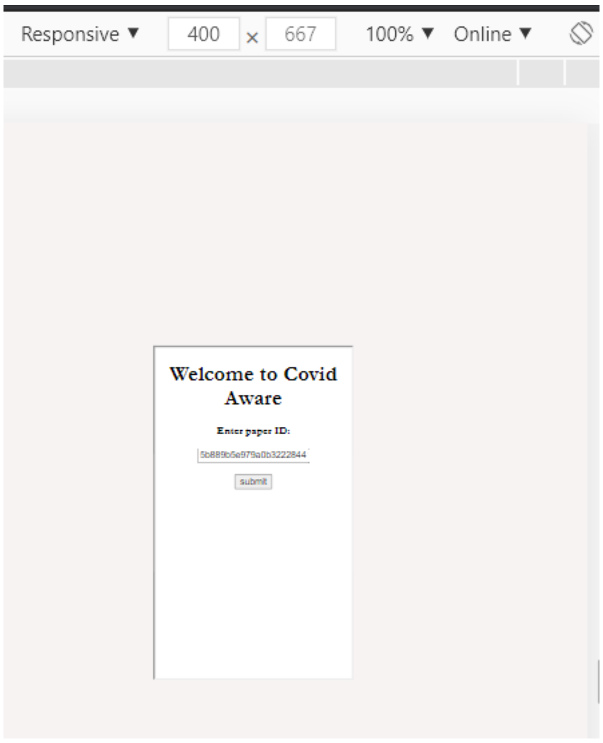
Table 2. App Performance tests.

The tool fails to load more than 45,000 rows in the dataset as it requires 25GB RAM allocation for 60,000 rows. The app does not require much time to load with less rows. Only the initial steps seem to be having an impact as the Vectorizer learns the vocabulary, then creates the count matrix and then finally the cosine matrix. Creating a huge cosine matrix requires high amount of memory, hence these challenges were encountered.

It can, however, be seen that after the cosine similarity matrices are generated, the whole application runs swiftly and requires no time to resolve the abstract, recommendations and the relevancy scores.

Website responsiveness

To check if the website is responsive and also works on different display resolutions, it was tested normally first on a computer browser (Google Chrome) and in responsive mode under developer tools to check if it would work on a mobile phone. As it can be noticed in figures 34 and 35, both the pages seem to be responsive and well-functioning. (Note: Figure 34 has been cropped to show only the important part and does not actually show the data collection form at the bottom left of the screen.)

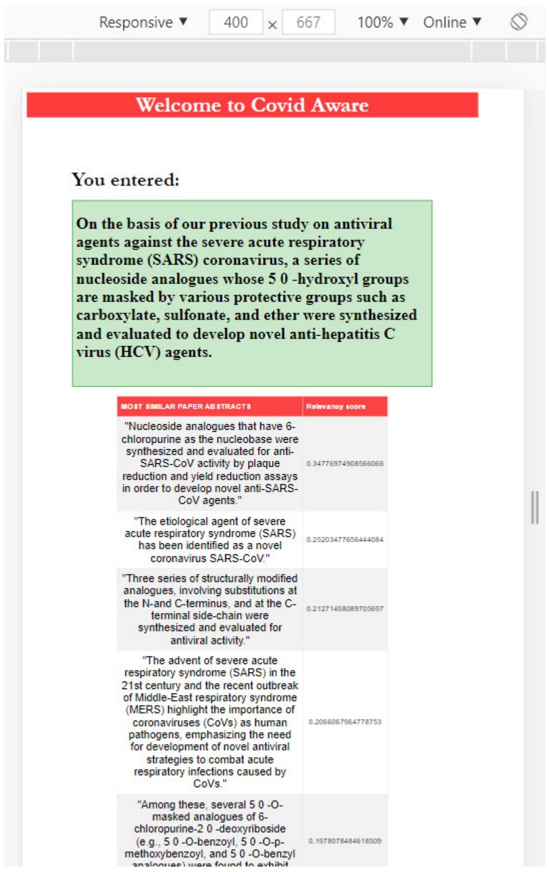


Welcome to Covid Aware

Enter paper ID:

5b889b5e979a0b3222844

Figure 34. Home page responsiveness on mobile browsers.



Welcome to Covid Aware

You entered:

On the basis of our previous study on antiviral agents against the severe acute respiratory syndrome (SARS) coronavirus, a series of nucleoside analogues whose 5 0 -hydroxyl groups are masked by various protective groups such as carboxylate, sulfonate, and ether were synthesized and evaluated to develop novel anti-hepatitis C virus (HCV) agents.

MOST SIMILAR PAPER ABSTRACTS	Relevancy score
"Nucleoside analogues that have 6-chloropurine as the nucleobase were synthesized and evaluated for anti-SARS-CoV activity by plaque reduction and yield reduction assays in order to develop novel anti-SARS-CoV agents."	0.34776974908566066
"The etiological agent of severe acute respiratory syndrome (SARS) has been identified as a novel coronavirus SARS-CoV."	0.25203477656444084
"Three series of structurally modified analogues, involving substitutions at the N-and C-terminus, and at the C-terminal side-chain were synthesized and evaluated for antiviral activity."	0.21271458089705697
"The advent of severe acute respiratory syndrome (SARS) in the 21st century and the recent outbreak of Middle-East respiratory syndrome (MERS) highlight the importance of coronaviruses (CoVs) as human pathogens, emphasizing the need for development of novel antiviral strategies to combat acute respiratory infections caused by CoVs."	0.2066067964778753
"Among these, several 5 0 -O-masked analogues of 6-chloropurine-2 0 -deoxyriboside (e.g., 5 0 -O-benzoyl, 5 0 -O-p-methoxybenzoyl, and 5 0 -O-benzyl analogues) were found to exhibit	0.1978078484618509

Figure 35. Results page responsiveness on mobile browsers.

10. Conclusion and Critical Evaluation

The implementation of the application has been successful, and the results have met the expectations to be qualified as a functional recommender system. A set of functional and non-functional requirements were set in Chapter 7, which have mostly been able to achieve.

To summarise the achievements, it can be stated that the application is able to import the dataset which is tsv format. The tool was able to process the whole dataset and vectorize it. The application is also successfully able to create the cosine similarity matrices and return expected results. However, as earlier noticed in the analysis section, the application fails to create the matrix when we load more than 45000 rows as it is not able to compute the cosine matrix for such a huge value on this computer due to limited resources. This can partially be considered as a limitation as it fails due to inadequate resources and not due to capability. However, this could also be attempted to eliminate using different methods or more research. The application also returned with the Paper Abstract when the user entered the Abstract ID.

Flask has worked flawlessly. After building the code, Flask initialises up well and under 30 seconds if the rows loaded in the dataframe are less than ~20000. The home page loads up instantly using either http://127.0.0.1:5000/ or http://127.0.0.1:5000/home. It loads our basic html page for data collection. The data (abstract ID) was successfully collected and passed to the function that returns the abstract and the function that generates the recommendations list. Within a response time of <1s, the output with the paper abstract and relevant documents with its relevancy scores was resolved.

To provide an insight particularly on the website; the website serves the functionality. It is not aesthetically very appealing neither very off-putting. The website is responsive and works well in computer browsers as well as mobiles and tablets as shown in the Analysis section. A shortcut to go back to the home page form the results page is also implemented which can be done by clicking on the top asserting, "Welcome to Covid Aware". What I personally feel is off-putting or could be improved is a problem in the home page which I noticed very late and was not able to fix in a short time frame. The background stays static when zoomed in and the text magnifies. If zoomed at 300%+, the text pops out of the box and starts looking aesthetically off-putting, so it looks good only at 80% till 125%. Being more creative, I could have also added some good logos, pictures and better colours to make it look better.

To point the **biggest flaw of this program** which I initially did not focus on and realised while testing is that **the code does not have an ValueError exception.** After mistakenly entering a wrong value, an error was encountered which made me realise that the code does not have an escape mechanism and would crash if a wrong input were provided. This is the main shortcoming of this application.

Finally, converging on the main objective of the application, does it provide accurate recommendations as expected? I believe the recommendation system serves it purpose and recommends good abstracts when TFIDF Vectorizer is used for tokenising the vectors, whereas Count Vectorizer has not shown results as good as TFIDF. There is always room for improvement but the recommendations by using TFIDF Vectorizer can be deemed satisfactory.

Some functionalities were considered optional, like having a SQL database and displaying relevancy scores on the results page. Unfortunately, SQL database was not implemented but that could have been a major improvement and definitely have a big impact on the overall outlook of the project.

11. Future Work

Future work would be to make this application more convenient and feature rich. There have been some difficulties and time constraints due to which some of the feature implementation has been left for future work. To focus on the future work, I first need to list out all the shortcomings of this application.

The features which can be improved as future work are as listed below –

- To be able to process large amount of dataset at a faster speed. At this point the current dataset used has 60000 rows, and when an attempt is made process more than 30000 rows, there is a significant difference in the initial processing speed as it creates a huge cosine matrix. This could be solved by better resources (higher CPU and RAM). Using a third-party hosting resource, based on the current implementation, the app would require 32/64 Gigabytes of RAM. With incremental data, this could be a considerable issue. New research papers would be added to the system on a regular basis and the application should be able to process without any manual intervention.
- To use a SQL Database to store the user information and the data. This will help us reduce the time spent on managing data and allow incremental data while also allowing the user to query the data in a better way.
- To allow the user to use search terms instead of being limited to abstract IDs. There would be a significant difference and would make the recommendation system useful even without knowing the Abstract IDs and can be used as an independent search tool. However, this is not a demerit as the purpose was to only design a recommender and not a search tool.
- Aesthetics – The looks of this recommender tool can be the next area of improvement. Although not essential, good aesthetics do attract more users and gradually helps gain trust.

- Make the recommender system as an internal service instead of an entire application or website. This service at the moment is a standalone website but the recommender system is better as a feature for a library system or research paper-based websites.

12. Reflection on Learning

As a learner and a beginner in the field of computing, I have learned and gained a tremendous amount of knowledge in a small amount of time. I had always been interested more in the hardware, environments and managerial side of IT and have worked in a similar role throughout my placement year and part-time thereafter. Hence, I never had an opportunity to work on something related to Data Science. My supervisor was very kind to let me work and guide me on a ML related project. Being this my first attempt working in the field, I have learned a variety of things. Since I had started from the fundamental level and my level of understand was very minimal, the whole project felt overwhelming. Whilst working under very limited time constraints for this project, I have also been working part-time for a company. This made it even more difficult to achieve it within the specified timeframe. Hence, the insights I have gained are much higher than my expectations. Coming from a different educational background, writing a dissertation has also been my first experience.

When first discussed with the project supervisor, it was decided initially that a SQL database would be used for storing the data but due to time constraints and having very limited knowledge, we decided to focus first on the main functionalities and then improve slowly in the lesser essential areas. I have also learnt to use version controlling better and efficiently, which will help me while working full-time on my current job as an administrator. I have learned some theoretical concepts in NLP like TFIDF Vectorizer, fit_transform, etc. But more importantly, I have learned how to find a solution and how to go about finding an appropriate way to solve a problem. I have faced huge amounts of problems and difficulties throughout this project but with the help from my supervisor and the internet, it was all possible. Even though I have not applied much html and CSS in this project as I was more focused on getting results and the app working, I am very happy that I could use the learning outcomes from other areas in the same project. Flask was something I had never heard about, so again this was my first experience working on a Web Framework.

Now, in addition to creating static pages for web development, I have also learned to work on the backend which gives me a complete idea of creating and hosting an entire application. The biggest challenge was to learn and handle all these different technologies, make sense of it, and put it all together to develop a working application.

Apart from the technical part, I have also learned many useful skills about researching, documenting and writing. Being an international student, I feel it is a very valuable skill as this is my first experience of writing something research based. In the initial stages, I read plenty of research papers and theoretical concepts. Soon I was lost as I could not connect the dots as all the concepts were new for me. I somehow succeeded in implementing the whole idea at a basic-cum-satisfactory level. After successful implementation, completing this write-up was the biggest challenge, as I have written this dissertation towards the end of my deadline due to unforeseen challenges during the biggest pandemic of the decade. In summary, I have developed a lot of valuable technical, researching and problem-solving skills (and realisation of the need for organisational skills) which will be advantageous for my future.

13. References

Agarwal, Nitin & Haque, Ehtesham & Liu, Huan & Parsons, Lance. (2005). Research Paper Recommender Systems: A Subspace Clustering Approach. 475-491. 10.1007/11563952_42.

Beel, Joeran & Gipp, Bela & Langer, Stefan & Breitinger, Corinna. (2015). Research-paper recommender systems: A literature survey. International Journal on Digital Libraries. 1-34. 10.1007/s00799-015-0156-0.

Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In Proceedings of the 8th international conference on Intelligent user interfaces, pages 263–266. ACM, 2003.

Flask.palletsprojects.com. API — Flask Documentation (1.1.X). [online] Available at: <https://flask.palletsprojects.com/en/1.1.x/api/#flask.Flask>

G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," in IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734-749, June 2005, doi: 10.1109/TKDE.2005.99.

Grimaldi, E., 2018. How to build a content-based movie recommender system with Natural Language Processing. [Blog] Available at: <https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243>.

Havan, E., 2019. Recommender System Application Development. [online] towardsdatascience. Available at: <https://towardsdatascience.com/recommender-system-application-development-part-1-of-4-cosine-similarity-f6dbcd768e83>].

Heidenreich, H., 2018. Natural Language Processing: Count Vectorization With Scikit-Learn. [online] Medium. Available at: <https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e>.

Lops, Pasquale & de Gemmis, Marco & Semeraro, Giovanni. (2011). Content-based Recommender Systems: State of the Art and Trends. 10.1007/978-0-387-85820-3_3.

MA, K., 2016. Content-Based Recommender System For Movie Website. Master's. KTH Royal Institute of Technology.

Pallets. 2020. Flask. [online] Available at: <https://palletsprojects.com/p/flask/> [Accessed 27 December 2020].

Pallets. 2020. Jinja. [online] Available at: <https://palletsprojects.com/p/jinja/> [Accessed 27 December 2020].

Pallets. 2020. Werkzeug. [online] Available at: <https://palletsprojects.com/p/werkzeug/> [Accessed 27 December 2020].

Paul Resnick and Hal R. Varian. Recommender systems. Commun. ACM, 40(3):56–58, 1997.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., 2011. scikit-learn. Journal of Machine Learning Research, 12, pp.2825--2830.

Philip, Simon & Shola, Peter & Abari, Ovye. (2014). Application of Content-Based Approach in Research Paper Recommendation System for a Digital Library. International Journal of Advanced Computer Science and Applications. 5. 10.14569/IJACSA.2014.051006.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J., 1994. GroupLens. Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94,.

Semanticscholar.org. n.d. Semantic Scholar | AI-Powered Research Tool. [online] Available at: <https://www.semanticscholar.org/>.

Singh, Ramni & Maurya, Sargam & Tripathi, Tanisha & Narula, Tushar & Srivastav, Gaurav. (2020). Movie Recommendation System using Cosine Similarity and KNN. 2249-8958. 10.35940/ijeat.E9666.069520.

Wang, L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R., Liu, Z., Merrill, W., Mooney, P., Murdick, D., Rishi, D., Sheehan, J., Shen, Z., Stilson, B., Wade, A., Wang, K., Wilhelm, C., Xie, B., Raymond, D., Weld, D., Etzioni, O. and Kohlmeier, S., 2020. CORD-19: The Covid-19 Open Research Dataset.

14. Appendices

The source code for the application has been uploaded along with this write-up.

The code has a specific directory structure and is recommended to use it as it is zipped without moving any files. The structure is as follows –

Main folder	**Sub-folders**
index.py	
Requirements.txt	
Abstracts.tsv	
\static\	Style.css
\templates\	Home.html returned_papers.html

It is also recommended to edit line 12 with the number of rows to be tested depending on the RAM as it might break the build if RAM<24GB

A requirements.txt file is provided which contains all the python modules required to run this code. The code has only been tested in Google Chrome and is recommended to use Google Chrome, although the webpages are responsive and have been built to work in any browsers or devices.

Inputs for testing this application can be taken from abstracts.tsv file.