Project Proposal n#1

Semi-Structured Text Document Search using Vector Space Model

*Project Report*

M. Aziz and R. Bou Zerdan

**Abstract**—Seach Engines are implemented as part of numerous websites and web applications to facilitate research purposes for the end users. With the advancement of technology and social media platforms, data forms have become increasingly varied varying from structured to semistructured and unstructured such as media, videos, images, and free text among others. To target this problem, a search engine was developed using numerous features such as the vector representation, term weighing, similarity evaluation, indexing to ensure accurate search results.

**Index Terms**—Data, Document Indexing, Indexing Methods, Natural Language Processing, Searching, Semantics

Adapted from the IEEE Computer Society template

—————————— ◆ ——————————

**Contents**

[1 Introduction 1](#_Toc150732264)

[2 Project Background 2](#_Toc150732265)

[3 Design 2](#_Toc150732266)

[4 Software design And Technologies Used 3](#_Toc150732267)

[5 Experimental evaluation 3](#_Toc150732268)

[6 Equations 3](#_Toc150732269)

[6 Helpful Hints 3](#_Toc150732270)

[7 Conclusion 4](#_Toc150732271)

[**References** 5](#_Toc150732272)

# 1 Introduction

T

He evolution of technology has called for new discoveries and methodologies to deal with the vast changes and advancements. For every new evolution, new hypotheses are needed and hence experimental validation to ensure that the results acquired conform with the needs of the advancement itself. When it comes to dealing with new technologies or dealing with constant changes, developing solutions through software, and testing it are crucial but there is no correct validation for it is an open-ended methodology for tackling any technical problem. Thus, the use of data, has become an important part in project software and technological developments where different types of data are used to test the questioned hypotheses and ensure that the results conform with the needs of current data and could be expanded to similar data resources. But similarity has been defined differently between different sources for they can be similar in terms of structure, semantics, content, among others. Hence, given the various data sources and given the variations that exist, semi structured and unstructured data have become more common with the proliferation of free text, media, and others. This has created a challenge to software developers to tackle the discrepant forms that exist between sources and generalizing one software over these sources. The variations of data sources and structure requires preprocessing to ensure that the resources are in conformity with each other and with itself among all the data.

This paper discusses the process needed to search semi structured and unstructured data based on the input text files. Hence, conversion between text files to csv files is needed requiring preprocessing of the data to conform with the search engine to be built based on the structure of the text files. Afterwards, pre-searching steps are taken into consideration involving tokenization, stop words removal, and indexing to be able to calculate similarity using the vector space model. As an advancement of a normal search engine that deals with structured data, semantic search is applied as a subcategory of natural language processing resulting in more accurate results based on meaning instead of solely relying on structure. Natural language processing is based on artificial intelligence and its purpose is to understand the written text as understood by human beings. This helps us in the process of dealing with unstructured data based on the words meaning to return the most accurate results.

The implemented search engine requires a user interface to be able to query the results based on keywords as other search engines usually function targeting nontechnical users and making searching easier by just typing words that should be relevant to the expected results. Hence, a local interface is implemented for the testing purposes and validation of results.

# 2 Project Background

## 2.1 Context and Problem Analysis

This project was developed for the purpose of applying acquired knowledge from the Intelligent Data Processing Course in terms of concepts needed for data searching such as TF\_IDF, indexing, tokenization, and vectorization. These concepts were used and further developed to deal with semi structured and unstructured data using Natural Language Processing which is a subcategory of artificial intelligence to deal with this unstructured format of data by referring to a more accurate approach and be able to treat the different tokens by their meaning rather than only structure. Semi structured and unstructured data is found in numerous formats but for the purposes of this project, text files were used to create the search engine and test its functionality.

A user interface is developed to let the user search for any keyword related to the topic included in the inputted text files. This returns the most related results from the text files and outputs them to the user.

## 2.2 Data Needed

For the purposes of this project, we used 4 input text files which were acquired from articles on the web, references to these articles are listed below. We named these as input\_1, input\_2, input\_3, and input\_4 [2], [3], [4], [5] which were preprocessed and combined before being inputted into the next step. In the next step, this data is used to perform the search model that was designed considering all the steps needed. Afterwards, the extracted data will be inputted into the user interface through API calls to get the needed data whenever a search is done.

# 3 Design

To design our search engine, we tackled 3 main steps which include data preparation and preprocessing, data searching function, and the user interface. The mentioned parts will be discussed thoroughly below.

## 3.1 Data Preparation and Preprocessing

Our data is extracted from text files which have minimal structure based on titles and subtitles. Hence, to be able to use this data, as an input to the search engine built, it should be transformed into another format which should be either Json, xml, or csv format where the semi structured data will be available for the next phase. We decided to parse our data into csv format and to do so we read the data from the files and delimited the text into different sections based on a the end of every section and start of a new one. In our case, we chose “.” as our delimiter, since the chosen data is only divided based on unnumbered titles and subtitles so in case the data was structured in a different way, the delimitation would have been performed somehow differently in order to work for the variations of inputted text files. The texts were extracted using technologies such as pandas and python and were divided into rows containing every section of the text. Then this data was divided into article\_title, titles (containing article\_title and subtitles in the article), date, and content sections. Afterwards, this data was preprocessed, by eliminating null values, replacing empty areas in the title section with the corresponding title, replacing empty areas in the article\_title section with the article. As well tracing every paragraph section with its title was performed to make sure that whenever the results are displayed, its title would be known as well. Then the preprocessing is applied to all the input text files in the directory and finally combined into one csv file to facilitate the process of applying the next steps.

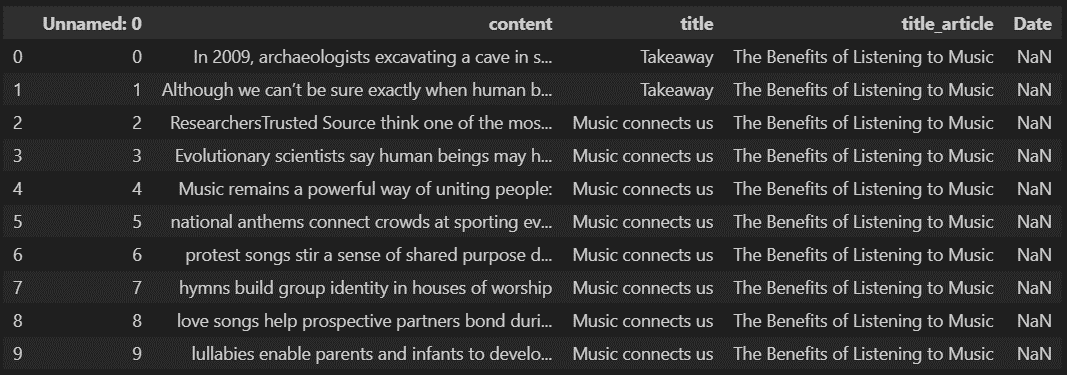


Figure : Data Preprocessing Output

## 3.2 Data Searching Function

The output data from the previous part is then inputted into this code section which tackles acquiring the correct results when searching. This data was tokenized word by word and then stop word removal was applied where an existent list of stop removal was included and other stop words were also added to it. Then the remaining list of words was destemmed to their roots ensuring that the similarity calculation would be performed correctly. [1] The vector space model was applied to the words and their similarity was calculated. Since we are dealing with semi structured data, we used Semantic Search as the approach for returning accurate results based on meaning as inferred by human beings. Semantic search is based on a form of Natural Language Processing which uses artificial intelligence to infer the meaning of audio sounds and human readable text in a human being way. Thus, this method will focus on acquiring the meaning of the search rather than basing the search on a predefined structure of the provided document. While combining the vector space model and the semantic search more relevant results will be generated. Some of these functionalities are imported from elastic search which embeds most of the needed functionalities such as indexing, scoring, and tokenizing.

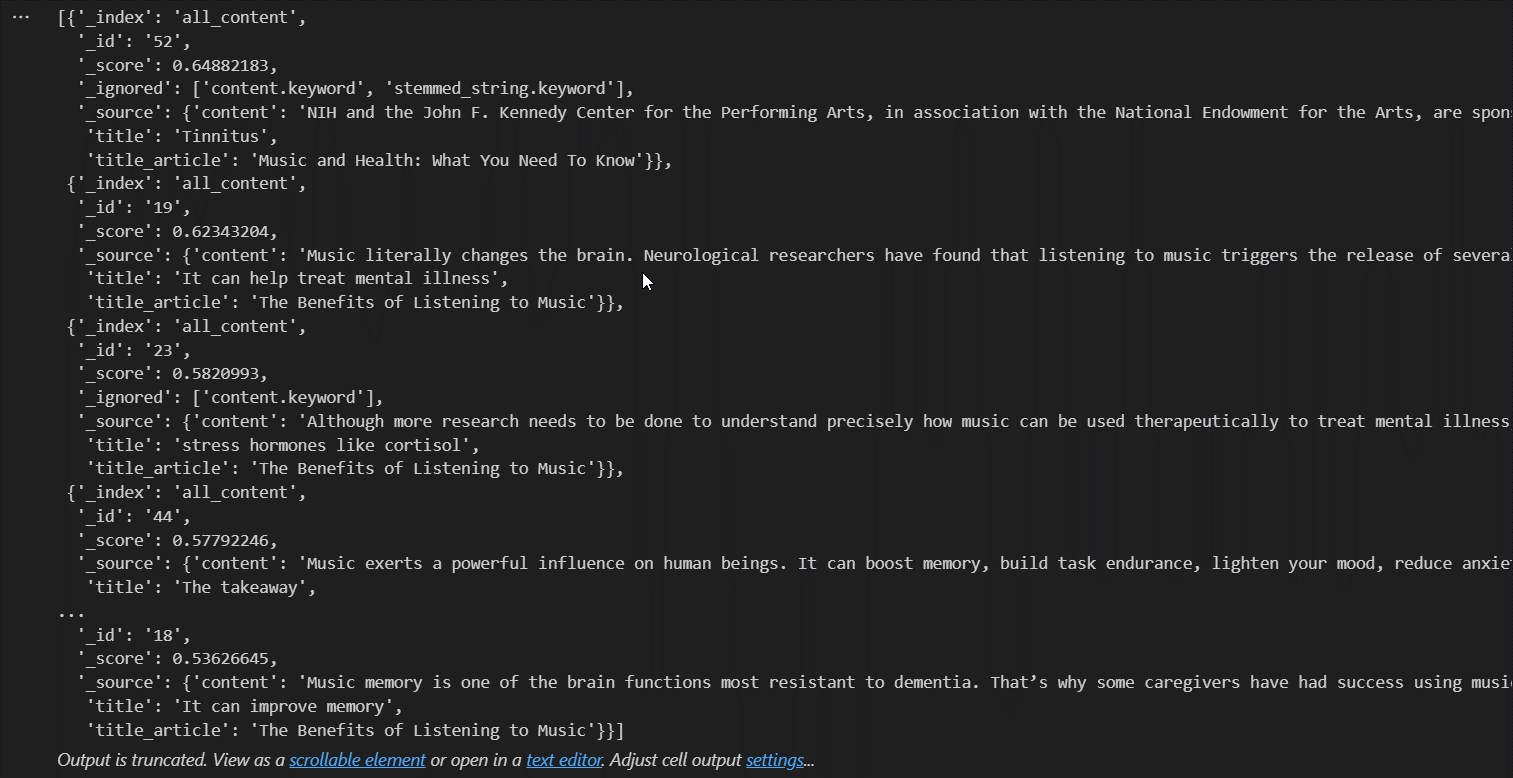


Figure : Vector Space Model Search Results

## 3.3 User Interface

A simple user interface is developed to display the results of the search. The user will be able to enter a sentence, phrase, or keyword which will be processed and return the most relevant results based on the technology used above. The user interface consists of one page hosted locally encompassing a search bar and an area to return the search results.

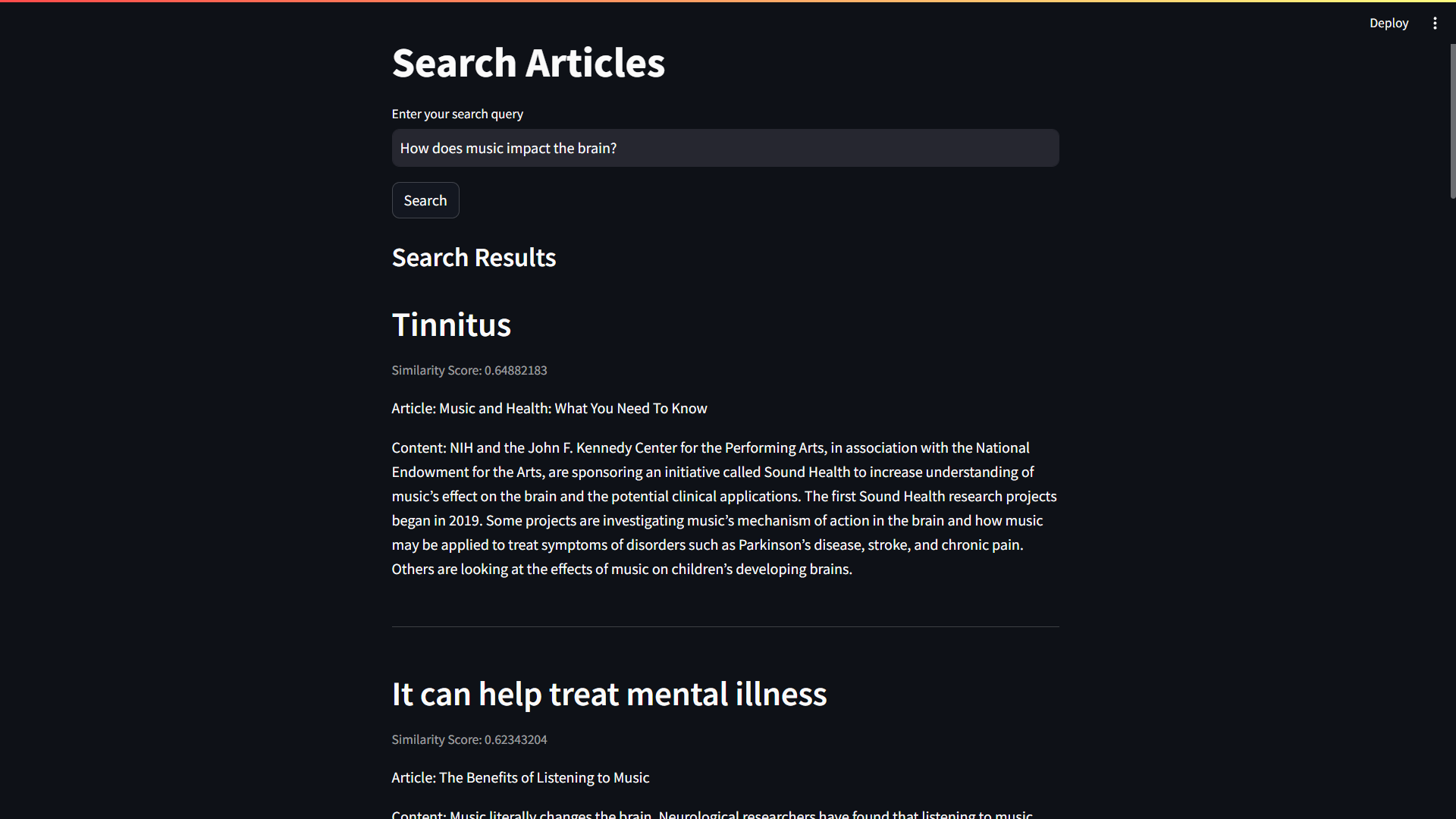


Figure : Screenshot of the Web User Interface

# 4 Software design And Technologies Used

The following tools were used in this project:

For Data Preparation and Preprocessing:

1. Python
2. Pandas
3. NLTK Library

For Data Searching Function:

1. Python
2. Elastic Search
3. Sentence Transformer

For User Interface:

1. Python
2. Stream Lit

Python is used as the main coding language due to its wide use in the field of data since it encompasses a lot of useful libraries that facilitate extracting, preprocessing, analyzing, and searching the data. For the data preprocessing numerous functions are used from the NLTK library which is a natural language processing library based on artificial intelligence. This library was implemented with the help of pandas ensuring that the actions are applied correctly on the needed column through dropping null values, filling out empty values, filtering needed data among others. The NLTK library provides built in functions that remove the initialized stop words after tokenizing the data into individual entities. We must make sure that the data is destemmed into its root values to facilitate the search on the software ensuring more accurate results. The elastic search is then used to ensure that the different entities are indexed correctly facilitating the retrieval process whenever a keyword is searched for. As well, to guarantee relevant results in addition to the similarity (concurrence) calculation, semantic search is used to return results which are related and have similar meaning rather than solely relying on the same word. Finally, to display the results to the user and make sure that he or she can search the data provided in the tested articles, a user interface was developed where the user is asked to enter a keyword or phrase and all relevant results will be returned based on the scoring caused by the vectorization model. This interface is bult using Stream Lit which is a Python library that allows building interfaces for users to experiment with data based projects.

# 5 Experimental Evaluations

In this section, the search engine functionality is evaluated, keywords are entered in the search bar which connects with the search engine and displays the resulting data. In our case, we used articles related to health and music; hence for example, when we search for the keyword “memory”, we get the below results.

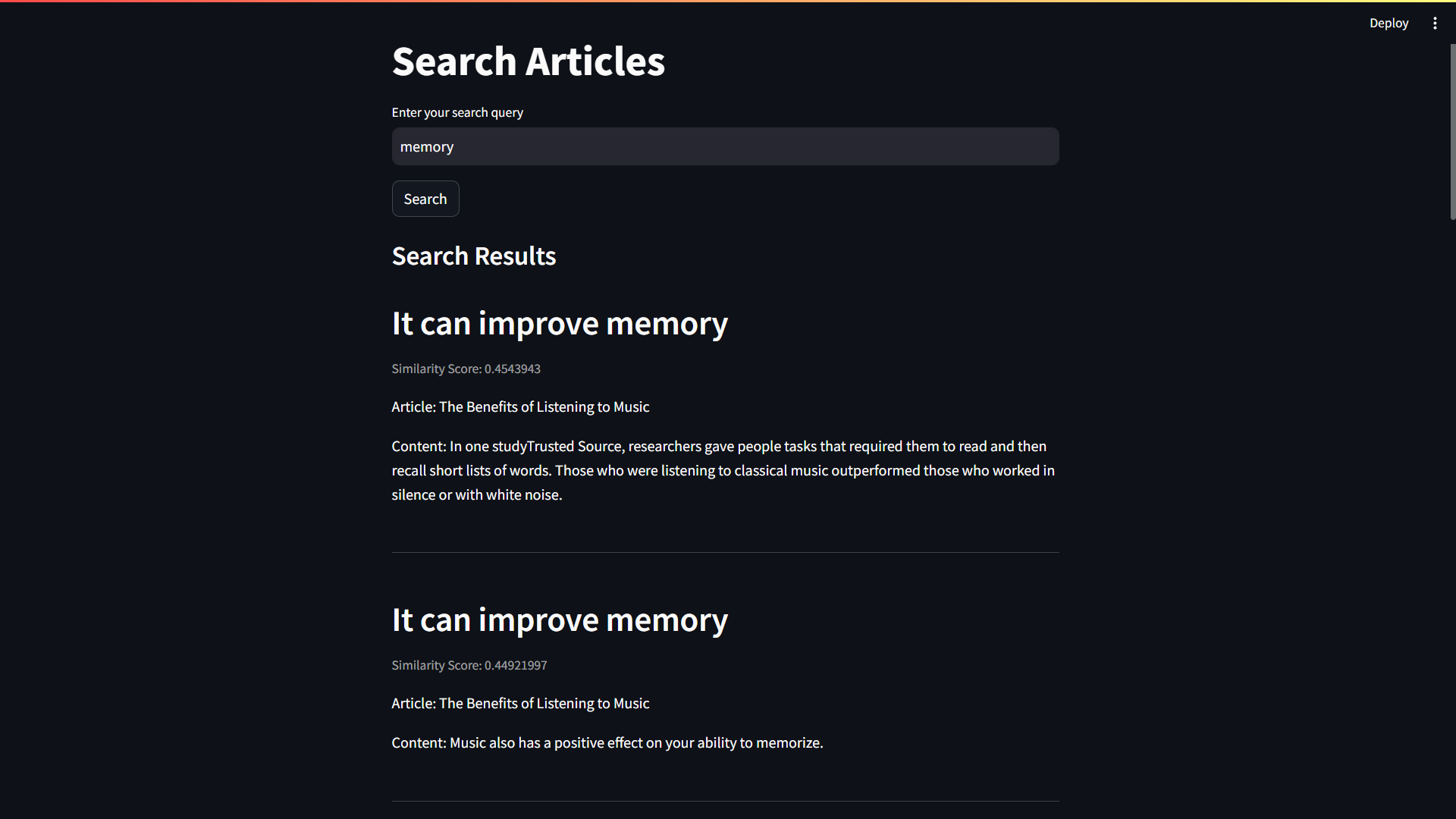


Figure : Example of a Search Query with results

Similarly, as previewed in Figure 3, the user can also input sentences as their query, and relevant results will be returned, based on the appropriate keywords in the sentence.

When the keyword is entered, it is sent to the search engine where it is evaluated and used as a reference for the search the similarity comparison is performed on the indexed documents which were preprocessed and organized according to search engine’s needs. The most relevant results based on the scoring will be returned in descending order on the user’s screen using API calls.

# Equations

* **Term Frequency (TF)** refers to the number of times a particular term appears in a document.
* **Inverse Document Frequency (IDF)** measures the importance of a term in the entire corpus. Higher weights are assigned to terms that are rare in the corpus, whereas lower weights are assigned to terms that are more common. , where N is the total number of documents and n is the number of documents containing the term.
* **BM25** incorporates **document length normalization** to counteract the document size bias by dividing the term frequency by the document’s length and applying a normalization factor.

Where |D| represents the length of document D, and avgdl is the average document length in the corpus. Parameters k1 and b are tunable constants that control the impact of term frequency saturation and document length normalization, respectively.

# Conclusion

In conclusion, the Semi-Structured Text Document Search using Vector Space Model project targeted the issue of semi structured data to ensure that known methodologies such as similarity, tokenization, and indexing among others are used along other advanced technologies to ensure accurate results. Semantic Search was the technology in our case but can be replaced with other technologies which might be artificial intelligence based as well. The proposed solution is thoroughly discussed throughout the report with applicable evidence and can be expanded as needed based on the targeted data and expected results. This project provided us with the opportunity to expand our knowledge in search engine building by building upon acquired knowledge based on search functionalities. This project revolved around Python mainly, but other languages can be used as well to attain similar results.

**References**

[1] “3   processing raw text,” ch03.rst2, https://www.nltk.org/book/ch03.html (accessed Nov. 12, 2023).

[2] “Keep your brain young with music,” Johns Hopkins Medicine, https://www.hopkinsmedicine.org/health/wellness-and-prevention/keep-your-brain-young-with-music (accessed Nov. 12, 2023).

[3] “Music and health: What you need to know,” National Center for Complementary and Integrative Health, https://www.nccih.nih.gov/health/music-and-health-what-you-need-to-know (accessed Nov. 12, 2023).

[4] “Music and health,” Harvard Health, https://www.health.harvard.edu/newsletter\_article/music-and-health (accessed Nov. 12, 2023).

[5] P. Everton Gomede, “Understanding the BM25 ranking algorithm,” Medium, https://medium.com/@evertongomede/understanding-the-bm25-ranking-algorithm-19f6d45c6ce (accessed Nov. 13, 2023).

[6] R. J. Stanborough, “Benefits of music on body, mind, relationships & more,” Healthline, https://www.healthline.com/health/benefits-of-music (accessed Nov. 12, 2023).