Final Project

MS in Data Science

ITC6008A1 - SEARCH ENGINES AND WEB MINING – FALL TERM 2022

Professor: Lazaros Polymenakos

Students:

**Pavlos Gaitanis**

**Anastasios Karmas**

**Ilias Siafakas**

Graphical user interface

Description automatically generated

American College of Greece

December 2022

Contents

[Introduction 3](#_Toc122012245)

[Run instructions 3](#_Toc122012246)

[Goal 4](#_Toc122012247)

[Dataset 4](#_Toc122012248)

[Process 4](#_Toc122012249)

[Preprocessing 5](#_Toc122012250)

[Search Engines 6](#_Toc122012251)

[TF-IDF 6](#_Toc122012252)

[Word2Vec 7](#_Toc122012253)

[TF-IDF with Sentiment Analysis 8](#_Toc122012254)

[Review Findings 9](#_Toc122012255)

[Future Work/Limitations 9](#_Toc122012256)

[Conclusions 10](#_Toc122012257)

# Introduction

This project is part of the evaluation for the ITC 6008 SEARCH ENGINES AND WEB MINING. In the project the theoretical knowledge and practical steps gained throughout the class were applied regarding web scraping and building an information retrieval system.

## Run instructions

Download the project folder and make the ‘Final Project’ the working directory. If you want to use the already retrieved dataset make sure the posts.csv file is also downloaded and exists on the same directory. If new data need to be retrieved, line 252 in the searchEngine.py file needs to be commented, and line 250,251 need to be made available so the API will work.

Make sure to add the client\_id, Client\_secret and user\_agent fields when initializing the reddit API on line 229. For more instructions on how to generate these values please follow the link below:

<https://towardsdatascience.com/how-to-use-the-reddit-api-in-python-5e05ddfd1e5c>

Required Libraries:

* PRAW
* NLTK
* Pandas
* SKLearn

Required Downloads:

nltk.download('vader\_lexicon')

# Goal

The scope of the project is to explore applications of web crawling in combination with Natural Language Processing to create a document retrieval system for the world of Formula 1. Each document retrieval system is built in the form of a search engine, which retrieves relative posts from Formula 1’s subreddit. In other words, the user creates a query formed in natural language as input, and the system produces relative documents as an output. The project achieves this by comparing the results of three different methodologies of document retrieval based on similarity, one using TF-IDF vectorizer, one Word2Vec, and the third TF-IDF with a sentiment analysis module.

# Dataset

The dataset used in this project.

The dataset was scrapped from Reddit using the Python Reddit API Wrapper (PRAW). The subreddit chosen, was the Formula 1 Subreddit as it was specified in the project description.

The dataset comprised of posts on that subreddit and their titles in a textual format, excluding image and videos.

PRAW returns the specified number of posts in an object with each object containing a subsection of the post. These were then accessed and made into a dataframe. A unique challenge this dataset presents is that there can be emojis and other non-standard characters that should be filtered before the models are fitted.

Some common properties of a PRAW post object include:

* id: A unique identifier for the post.
* title: The title of the post.
* body: The body of the post.
* author: The username of the person who submitted the post.
* created\_utc: The time at which the post was submitted, in Unix time format.
* subreddit: The subreddit in which the post was submitted.
* score: The number of upvotes the post has received.
* num\_comments: The number of comments the post has received.
* permalink: The URL of the post.

# Process

The 1000 most recent posts from the Formula 1 subreddit were extracted using the reddit open API and the PRAW library. After preprocessing the posts to remove stop words, tokenize and clear the text of any unnecessary characters, only the high quality posts were kept and they formed the working corpus. After having created the corpus, the user is asked to perform a query to retrieve relative posts. The corpus and the query are passed to each of the three search engines, where they are vectorized, and in the case of the third search engine, the sentiment is also extracted. Each search engine initializes a vectorizer, which is then fitted on the documents. The documents and the query are then transformed into vectors, and the dot product between the query vector and each document are calculated. Each document is ranked based on the relevance to the query, and the top three ranking scores are displayed back to the user.

# Preprocessing

Preprocessing is a crucial step in NLP as it helps improve the performance and accuracy of downstream NLP algorithms by cleaning and normalizing the input data. By carefully selecting and applying the proper preprocessing techniques, NLP practitioners can ensure that their algorithms can accurately and effectively process natural language data.

As mentioned above, the dataset as scrapped from Reddit was not in a form that would be able to be used with any of the methods chosen to tackle this problem. Therefore, some pre-processing steps had to be applied to the dataset to make it easier to work with and more appropriate for the algorithms. These steps will be further outlined in the next paragraphs.

The first step taken was removing the punctuation and breaking up the sentences into words in order to allow the processing at a later stage to take place at the word level. In the next step the most common stopwords of the English language were removed from the tokenized text of each post leaving behind only the important words. To list a few, ‘the’, 'and’ and ‘a’ were some of the removed stopwords. Finally, the now tokenized and stop word free text was ready to be fed into the document vectorizer to prepare for the next step.

Below is a snippet of the original text along with the tokenized result.

A picture containing calendar

Description automatically generated

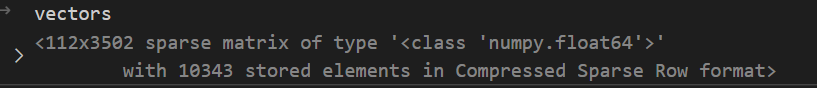
# Search Engines

## TF-IDF

TF-IDF, short for term frequency-inverse document frequency, is a numerical statistic used to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The goal of using TF-IDF is to convert the raw text into a vector of numbers that can be used as input to a machine learning algorithm.

In Python, the TfidfVectorizer class from the scikit-learn library can be used to perform TF-IDF vectorization. This class takes a list of strings, where each string is a document in the corpus. It then computes the TF-IDF values for each term in each document and returns a matrix of TF-IDF values, where each row represents a document and each column represents a term.

In our case, the TF-IDF search engine is initialized using the TfidfVectorizer class from the sklearn library. The dataframe that holds all of the preprocessed posts is further processed, to extract a single list that holds all the documents. The vectorizer object is fitted based on the document list provided, and then the documents are transformed into vectors. A 112x3502 matrix is created that holds the vectors for each document.



The query is also vectorized using the same vector space. The Tf-idf score of a word is calculated by multiplying its term frequency (the number of times it appears in the document) by its inverse document frequency (the number of documents in which it appears). The code then uses the dot product between the query vector and each document vector to compute the similarity between each post and the query, and returns a list of posts that have a similar sentiment and a high Tf-idf score.

The list of individual scores is appended as a new column in the original dataframe, and the list is sorted based on the documents that have the highest score. The top 3 highest scoring documents are then retrieved and displayed to the user along with their similarity score.

Results:

Text

Description automatically generated

We can see that using the query to retrieve posts around Vettel’s bad qualifying performance, returns three posts, one is directly related to our query and Vettel’s performance, while the other two discuss general formula 1 posts that affect performance stats during the race weekend.

## Word2Vec

A second attempt at a search engine was made using the Word2Vec approach. Word2vec is a technique for learning continuous-valued vector representations of words, based on the context in which they appear in a large corpus of text. The word2vec algorithm uses a shallow neural network with a single hidden layer to learn the vector representations of words, known as "word embeddings". The algorithm uses a combination of two learning methods: continuous bag-of-words (CBOW) and skip-gram. In CBOW, the algorithm predicts the current word based on the context of the surrounding words, while in skip-gram, the algorithm predicts the context words based on the current word. By training the neural network on a large corpus of text, the algorithm learns the relationships between words and their meanings and can produce high-quality word embeddings.

Because of the limited scope and source of our documents, such large corpus did not exist so the training was done on the limited posts that were retrieved. In addition to this, due to technical limitations, a pure Word2Vec library was not used, instead, the TFIDF vectorizer from the previous step was repurposed to simulate the word2vec process.

Text

Description automatically generated

The rest of the process followed was similar to the TF-IDF model, but the results obtained were quite different.

Text

Description automatically generated

For example, you can see that the top result is still the post about the analysis of Vettel’s qualifying, but on second place we are getting some unrelated -to Vettel’s qualifying pace- post that made its way to the top of the list.

By further analyzing the post we are seeing that it uses some of the keywords that also existed in the search, but because of the short length of the post, it receives a higher score than other, larger posts.

## TF-IDF with Sentiment Analysis

Finally, an attempt was made to provide accurate results to the user’s query by also taking into account the sentiment of their request and comparing it to the sentiment of each post.

This search engine uses the SentimentIntensityAnalyzer class from the NLTK library to compute the sentiment score for each word. This score is a numerical value between -1 and 1 that indicates the degree of positivity or negativity of the word. Next, the code computes the average sentiment score for the query and each post. This is done by summing the sentiment scores of all the words in the query or post, and dividing by the total number of words. The code then uses the cosine similarity metric to find posts that have a similar sentiment to the query. This is a measure of the similarity between two vectors, in this case the sentiment vectors of the query and each post. Both of the metrics are used to calculate the total score of each post before returning them to the user. The total score is weighted more towards the result of the cosine similarity, since the goal is to retrieve similar posts to the user’s query, but then is modified to boost posts that have similar sentiment.

This is more apparent if we use a more controversial search, for example Ferrari’s performance in the 2022 season, and asking the search engine to retrieve positive versus negative results for their performance.

When asking if Ferrari had a good 2022 season, this search engine retrieves opinion posts that outline ferrari’s performance given the circumstances and the competition, claiming that they did better than expected. The search engine also retrieves posts about predictions made at the start of the season where Ferrari was one of the favorites for winning.

Text

Description automatically generated

On the other hand, if we search for Ferrari’s worst season so far, the search engine returns posts that talk about how Ferrari would benefit from a handicapped championship given their ending situation in 2022, and a post about the worst drivers in the grid for next season, where there is also talk about the future of Ferrari’s drivers.

Text

Description automatically generated

It is also worth noting that both searches returned a single post which was the same, where it talked about the fastest pitstops of the season and how Ferrari had some good attempts, but overall remained average.

# Review Findings

Overall, we can say that the test was successful as the search engines returned relatively relevant posts based on the user’s queries. If we compare our search engines, we will notice that TF-IDF with sentiment analysis had the best range of results, while it had the lower total scoring for each document. While the confidence was low, it still managed to identify the most relevant document correctly and is in line with the rest of the search engine and with the goal of this project.

Text

Description automatically generated

# Future Work/Limitations

For future work, a more standardized way of retrieving information can be achieved. Reddit is a hard website to be crawled since it has many ways of accessing information. The API and documentation provided are quite old and do not offer a way to filter and search for specific topics to be retrieved. For example there is not an easy way to currently retrieve posts regarding last year’s season. Because of the novelty of the application, no large dataset exists so the search engines and their models can be trained.

Finally it is interesting to see how other language models and algorithms can be applied to document retrieval, like BERT variations and other LSTMs.

# Conclusions

In conclusion, this project has shown that by using different NLP models we can accurately retrieve relevant documents while also predicting the sentiment of a text. This can be used to find the polarity of a post and retrieve relevant information. This can be further applied to achieve the original goal of the project and display the most useful posts based on the question the user has. In a more practical view, through the project we create a basis to be able to provide accurate information to fans interested in Formula 1 topics.