

Master of Science (MS) in Data Science  
Module: ITC6008A1 - SEARCH ENGINES AND WEB MINING

Google App Reviews: Information Retreival

Instructor: Dr. Lazaros Polymenakos

Students’ names:

Alkiviadis Kariotis 241735

Celsa Almeida 267628

Georgios Aliferis 272183

Konstantinos Margaritis 122451

Term: Fall Term 2023  
Type: Group Term Project

Submission Date: Sunday 17th of December

Words:

Table of Contents

[Introduction 3](#_Toc153739914)

[Main Approach 4](#_Toc153739915)

[Libraries 4](#_Toc153739916)

[Scraping 6](#_Toc153739917)

[App Searching 6](#_Toc153739918)

[Testing Single App review 7](#_Toc153739919)

[Data Preprocessing and Cleaning 8](#_Toc153739920)

[Indexing and Retrieval 9](#_Toc153739921)

[TF-IDF Vectorization 9](#_Toc153739922)

[Query Expansion and Boolean Search 10](#_Toc153739923)

[Sentiment Analysis 10](#_Toc153739924)

[Recommendation System 12](#_Toc153739925)

[User feedback analysis 12](#_Toc153739926)

[LDA and topic extraction 13](#_Toc153739927)

[Machine Learning 14](#_Toc153739928)

[Word Embeddings and Similarity 15](#_Toc153739929)

[Final Query processing and IRS 16](#_Toc153739930)

[Parallel approach with BERT (Bidirectional Encoder Representations from Transformers) 17](#_Toc153739931)

[Conclusion 19](#_Toc153739932)

[References 20](#_Toc153739933)

# ­­

# Introduction

Our team project focuses on taking advantage of various data analysis, visualization, and natural language processing techniques to process the user app reviews we have scraped from the Google Play store. Our analysis is performed on the Social Media apps commonly used Greece, based on the variables that we have set through our code. The results may be changed in a dynamic way, since from just executing our system, the latest and relevant reviews of the app category that a user wants to search for will be provided along with the ability to change the category. We proceeded with various NLP tasks to create an information retrieval system, indicatively involving text preprocessing, sentiment analysis, word embeddings etc. Finally, we deep-dive into topics of machine learning and embeddings to enhance our understanding of the user reviews and to improve our recommendation system. In parallel, pretrained models from the Hugging Face platform were utilized to summarize and query our dataset. Lastly, an attempt was made to evaluate sentiment in the customer reviews with pretrained models.

# Main Approach

## Libraries

We imported the necessary libraries for the project which included those concerning data analysis, visualization and NLP tasks :

* json library, which allowed us to handle and deal with json data formats.
* pandas library, which gives us the ability to manipulate data easily and perform a fast and effective analysis on them.
* tqdm library, which is focused on creating progress bars, in order to inspect our progress as we parse through loops and iterations throughout our entire code snippet, giving as a visual indication of the progress of the executing code.
* seaborn library, which in combination with matplotlib is used for purposes of visualization. This specific library provides high level of interface in order to create quite attractive and informative graphs concerning mostly statistical graphics.
* From the pygments library we used highlight, which is a function that is used in order to achieve syntax highlighting.
* Again, from the pygments library we imported the JsonLexer which was used to specify the specific lexer of json code format.
* Also, from the pygments library, we used the TerminalFormatter, which is a function that is used to format the code in order to be projected in the terminal as output.
* From the google play scraper we imported four different functions, the sort, the reviews, the app and the search. That functions were used in order to achieving the scraping of the information of the apps from the Google Play Store, including all their details and also their reviews.
* From matplotlib we used the inline and InlineBackend.figure\_format='retina commands, which are commonly refered to as magic commands. They are used towards configuration of the inline plotting in Matplotlib and adjust the format resolution in retina, in order to achieve a higher resolution.
* From the nltk.corpus we imported the module wordnet, which is the way to access not only lexical resources but also lexical databased for English words.
* From the sklearn.metrics.pairwise we imported the cosine\_similarity gunction which is used to calculate the cosine similarity between the vectors that we constructed through obtaining the app reviews and is suitable for text similarity tasks.
* re library, which is used for pattern matching and also for text manipulation by the use of regular expressions.
* From the sklearn we imported the a predefined set of common English stop words called ENGLISH\_STOP\_WORDS, which will come handy for text preprocessing.
* Then again from the sklearn we imported the SnowballStemmer which is a function used for word stemming, and more specifically to reduce the words to their base form.
* We imported the TfidfVectorizer, used to convert text data into TF-IDF feature vectors.
* Textblob in order to simplify some NLP related tasks like sentiment analysis.
* LatentDirichletAllocation from sklearn that we took advantage when we proceeded with LDA modeling.
* Counter function from collections library, which count the occurrences of elements.
* The train\_test\_split from sklearn to be able to split automatically our data into that three categories and directly use them in our machine learning models.
* MultinomialNB from sklearn, which provides us a way to perform a text classification through the use of Multinomial Naïve Bayes.
* LaberEncoder again from sklearn, in order to be able to convert categorical data into arithmetic or numerical formats.
* Classification\_report from sklearn again, as to generate a detailed report regarding our classification performance.
* Kmeans from sklearn, to be able to appply a K-means machine learning algorithm as to cluster our data.
* Word2Vec from gensim.models, which is a way to learn embeddings from text data.

## Scraping

### App Searching

A screenshot of a black and white screen

Description automatically generatedWe conducted a search for apps on the Google Play Store, specific to the Greek marketplace. The search query used is dynamically changeable and with the delivered code snippet we proceeded with the search of social media apps. The code is modified dynamically to bring back reviews from the Google Play Store that are categorized in different application sectors. The number of total hits that we are able to acquire are 30, due to Google’s maximum limit that is enforce for this kind of search (Google Play Scraper, 2023). Within this limit we retrieved over 26 social media apps while ensuring that our retrieved data are relevant and focused . Through the appropriate library, a search is performed that involves specifying the country and the language that we need to focus on (in our case: English app reviews in Greece). After obtaining the results, we extracted key information. We created a list consisting of dictionaries, with each one of them containing details about the apps scraped, such as the appID, the title of the application, the rating score that the users have given to it, the genre or category that it belongs and finally a URL leading to the app icon or logo. So as to have more flexibility we created a data frame with the use of the pandas library, in order to have our data in a more structured and tabular format, making it easier to perform further analysis, visualization or any other further analysis that we conducted. Here is a preview of the aforementioned data frame that we obtained:

By iterating through the respective dataframe we generated an HTML representation of the icons or logos of the apps which we have acquired data from, inside the Jupyter environment:

A screenshot of a cell phone

Description automatically generated

### Testing Single App review

We proceeded with fetching user reviews for a single app that we have successfully identified. By using the appID, we accessed it in the data frame. So, this is the format that we have managed to obtain the reviews in:

A computer screen with white text

Description automatically generated

After doing the aforementioned necessary safety checks we looped our requests through the Google Play scraper, in order to get the reviews for multiple apps. Here we placed a filter, which fetched 200 reviews if their score was 3, or if any other case applied, it brought back 100. That was done in order to create a more balanced dataset, because of the tendency that apps review ratings have towards five stars. So, as to avoid other more complex machine learning ways to fix the imbalance we decided to filter this out during our scraping phase. Also, another filter applied here (which can be modified depending of the needs of the investigation each time) is that the retrieval of reviews is done based firstly on the most relevant and then based on me newer ones. The total reviews that we managed to bring back were 19200 and then we proceeded at saving them into a csv file, because it took time to collect all those app reviews in combination with the filtering conditions that we have set up in the loop of it.

## Data Preprocessing and Cleaning

Following the scraping process and saving the app reviews in the respected csv file, we proceeded with cleaning the data. After inspecting the data searching for anomalies or any weird values or formats, we proceeded with removing the duplicate rows based on the ‘content’ column, in order not to lose any important data. Then we proceeded with dropping some columns that we considered to be irrelevant in relation to our project scope, which for example were the ID of the reviewer, the content of the reply of the app in the review of the user, the user image etc.

We then proceeded with the processing of the text itself, in order to reduce the complexity for our future analysis and NLP tasks applications. All the steps were brought together in a function called clean\_tokenize, where we converted the text into lowercase, removed any characters that were not alphabetic and finally excluded all the common English stop words that are found within the aforementioned ‘content’ column. Furthermore, we applied stemming in order to reduce contained words into their base root form.

## Indexing and Retrieval

In order to be able to retrieve information effectively and efficiently based on a natural language text query from a user, we first built indexes as a base for the retrieval techniques that we planned. More specifically we focused on the inverted and positional indexes. Firstly, we implemented the inverted index function, in order to create a mapping for the content ( words or terms) w.r.t. where they are located in the dataset (Pibiri & Venturini, 2020). We iteratively process the ‘content’ in each row of for each user entry, looking at each word/token and updating the inverted index while adding the identifier (some identifiers may be associated with the specific word). In this way, we have an easy tool to locate or map all different instances of words across the entire dataset.

We then proceeded with the positional index which is a more advanced version of the inverted index. The main difference being that while the inverted index maps terms in relation to the documents that they appear in, the positional index moves a step further t by tracking the position that each term has in each of the documents that it exists (He & Suel, 2012). We applied the positional index method through a dedicated function, and stored the results in a nested dictionary structure, where we used as first key the word itself, as second key we placed the identifier of the document and in ‘values’ we stored all the positions that the word appears in the respected nested second key document. This way we were able to easily access every word/token by just mapping through the dictionary, similar to using a decision tree.

## TF-IDF Vectorization

In this part we employed the TF-IDF (Term Frequency Inverse Document Frequency) vectorization, which is used in order to rank users, in our case, based on their review content, based on their relevance (Huang, Yin, & Hou, 2011). We used this class to convert our review data column into numerical vectors, in that each vector will be representing the importance of each one of them in each review and at the same time being relative to the entire corpus of the reviews. This matrix then would be found very useful when it comes to future actions like text classification, applications of other machine learning techniques, statistical analyses, sentiment analysis techniques or just identifying important terms across the reviews.

## Query Expansion and Boolean Search

In this section of our python code snippet, we started to build a query expansion technique, used towards information retrieval in order to improve the recall of the relevant documents through augmenting the original query that the user has provided, with additional terms that are based on similar context or meaning, and in more simple terms it will locate synonyms, by offering an example using NLTK’s WordNet (Farkiya, Saini, Sinha, & Desai, 2015). So, through the ‘wordnet’ we are able to get the access at a lexical database, which contains meanings of words, their synonyms and also the relationship between them. We take a query and through our function called expand\_query, we search for synonyms in the aforementioned module, and aggregate them into a set, returning back a string containing synonymous terms.

We then proceeded with boolean search, which provides us with the ability to retrieve relevant results based on the user query, in terms of the inverted index based on the boolean operators, referring to and/or/not in order to retrieve the desired documents based on specified conditions (Aliyu, 2017). So, we created a function named boolean\_search, which receives the query and an inverted text as two inputs, and through the use of the logical operators, it creates and returns the IDs of a set of documents which are highly associated and also based on the inverted index that has been provided.

## Sentiment Analysis

Sentiment analysis, which is also known as opinion mining, is a process of processing natural language text data, by trying to categorize the emotional tone and polarity that it may be expressed in a piece of natural language text, separating into three main categories: the positive, neutral, and negative (Medhat, Hassan, & Korashy, 2014).

This part is allocated towards the determination of the overall sentiment polarity sentiments of our user reviews, where we constructed a function which performs an analysis based on the library of Text Blob, in which the polarity of the text is analyzed based on a condition and if it is greater than zero it is assigned a positive sentiment. If it has a value less than zero then it takes the negative sentiment, and if it is exactly zero it obtains the neutral value. All the fetched values from the respected function are then stored into a new column in our data frame.

Then we proceeded with the aspect-based sentiment analysis, which is a more advanced form of the ‘traditional’ sentiment analysis that is able to assess the sentiment polarity of a specific aspect, features, or even categories that are mentioned in the text that is being used on (Do, Prasad, Maag, & Alsadoon, 2019). So, in order to assess specific aspects of the reviews we proceeded with defining specific aspects, such as ‘performance’, ‘user interface’, ‘customer support’ etc., in order to evaluate whether anyone of the predefined aspect it is mentioned in the specific review that is being examined. From that analysis we stored the results in a new column, named ‘aspect\_sentiments’, in the reviews\_df, which contains dictionaries of mapping let’s say aspects to their respective sentiment labels.

Furthermore, we proceeded with the identification of frequent aspects, through our texts collection of reviews, providing at the same time a list of them. In text analysis, aspect identification can be defined as the process of recognizing and retrieving or extracting the important aspects, topics or the keywords that are more frequently mentioned in the data set that you are comparing them to (Feng, Cai, & Ma, 2019). So, we build a function to satisfy this purpose, which identifies the most common noun within our users’ reviews, beginning at first tokenizing and tagging all the words in the text, then it filters out the nouns and proceed with calculating the frequency distribution of them, returning finally a list of editable number of the top variable that we set it to fetch back. Here we also made a visualization of the distribution of the sentiments, that we were able to identify based on our methods:

A graph with blue bars

Description automatically generated with medium confidence

In order to dive deeper into our analysis, for the part of the aspect sentiment analysis, who created also a plot for the top 10 aspects, regarding their frequency, showing how the three categories of sentiment are distributed through the top 10 topics identified. Here is the plot:

A graph with different colored bars

Description automatically generated with medium confidence

## Recommendation System

A recommendation system is a type of filtering information system, which purpose is not only to predict but also suggest related items or content to the user that prompts it (Robertson & Hancock-Beaulieu, 1992). Here we proceeded with two different approaches in order to suggest apps to users, one method used was primary relied on similarity score, and the second one is a combination of similarity and positive sentiments. So, the first method based on cosine similarity, receives a user query as input and then calculates the similarity between the input provided and the term frequency inverse document frequency vector representation of the user reviews from our main data frame. Starts by firstly transforming the natural language text input query into a vector, then makes the comparison, and then returning the top five apps with the highest cosine similarities scores. Then the second approach, follows the same process but it also adds on top an extra consideration layer, the sentiment, so again it transforms the query into a vector, then calculates how relevant is the score with each review score, selects the top ten that are close and filters them to retain only the one’s referring to positive sentiment reviews, and then returns the final results based on similarity and sentiment.

## User feedback analysis

Here we proceeded with trying to analyze our users reviews or feedback, in order to extract insights or patterns from the user generated content that we have managed to scrape, and try to understand common issues, concerns or trends that our users may be experiencing. So, we started by creating a function in order to identify the most common issues. We cleaned the review column that contained all the text and merged it into a single text corpus then this corpus was tokenized into individual words and calculated the frequencies of them by using the counter class. As a result, from the function, we get back a list with the number of the most common issues that we have said, alongside their respective frequencies. Then we dove deeper to not only identify the common issues but also try to explore how sentiments are associated with these issues that we are investigating. So, for each issue that we have identified we examined their keywords in comparison with the sentiment distribution within those reviews, saving the result into a dictionary in which each issue is associated with the respective sentiment counts providing us the ability to get a more comprehensive view of how the sentiments are distributing for each one of the issues that we have investigated.

## LDA and topic extraction

In the context of machine learning and natural language processing a probabilistic model is used called Latent Dirichlet Allocation or LDA, which aims towards discovering in a collection of documents the latent topics or themes (Jelodar et al., 2019). So, here we imported and specified the aforementioned model, fitting it into the TF-IDF matrix which has been created in the earlier steps and is representative for the entire corpus of the reviews of the users collected. The code then proceeds with identifying for each associated topic the most significant words, and then through the LDA model we are able to specify the number of the top words must that we need to extract from every topic that we are dealing with. So, we reiterate through each topic, get back the words with the higher probabilities, store them inside a dictionary and create at the same time a relationship between the stored values and the respective topic. Then, we did the same process but now combination with the associated sentiments of the reviews of our users, which was applied after we identified the most dominant topic for each review through the higher probability, and then segmenting them based on the dominant topic and the respective sentiment label. Through the whole aforementioned process, we were able to extract some insights to get a clearer idea on the core themes regarding user feedback and the connection of each one with the sentiments and the topics.

Then we proceeded with trying to evaluate the quality of the recommendations of the apps, based on what we've performed above. Through assessing the sentiment distribution based on the user query we try to assess the effectiveness and the quality of the resulting recommendation by the use of various criteria and metrics. So, based on not only the vectors of similarity of the query but also on the analysis of the positive sentiment, we proceeded with calculating the average sentiment distribution of all the reviews that we have in our data set, and here is the printing of the respected python code snippet:

A black background with white numbers

Description automatically generated

## Machine Learning

Furthermore, we proceeded with the clustering machine learning technique. The scope of this is to group together similar data points and assign them to the so-called clusters based on common similarities or features found in the dataset (McGregor, Hall, Lorier, & Brunskill, 2004). So, we created five clusters through the use of the K-Means model and applied the fit\_predict method on the TD-IDF matrix, in order to assign each review to its respective cluster. Here is the plot:

A graph showing a cluster of clusters

Description automatically generated with medium confidence

Moreover, for deeper investigating purposes we tried a classification task, whereby using machine learning we try to assign data points, specific labels based on their characteristics (Kotsiantis, Zaharakis, & Pintelas, 2007). So, we set as X the TD-IDF matrix and as Y the encoded labeled sentiments by using the LabelEncoder. Then we split into train and testing using 80% for training purposes, and the leftovers for testing, using a Multinomial Naïve Bayes classifier. Here are the results that we obtained from the aforementioned process, regarding our models’ accuracy:

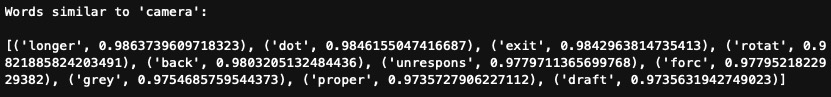
A screen shot of a computer

Description automatically generated

## Word Embeddings and Similarity

Here we proceeded with word embeddings, which are a type or representation of words, that allow words having similar meaning to have similar representations, achieved by mapping each individual word into a real value vector inside a predefined vector space, where the vector values are learned in the same perspective that neural network does (Kusner, Sun, Kolkin, & Weinberger, 2015). So, we used Word2Vec technique, which uses a pretrained two layer neural network in order to produce a vector space of typically some several hundred dimensions, by assigning each one of the unique words of the entire corpus to a specific corresponding vector from the space, positioning their vectors in such a way so that they share common context regarding the corpus and they are located close to another inside the space (Church, 2017). So, we applied the technique on the reviews with vectors of size 100, meaning that each of the words given would be a 100-dimensional vector, with window size set to 5 and min count set to 1 so that even words will be included only once in the model. Here are some results of the application of the Word Embeddings:





We tried also the Jaccard index, which is a measure for calculating the similarity between two different sets. This technique is defined as the division of the size of the intersection with the union size of the sets (Fletcher & Islam, 2018).



## Final Query processing and IRS

In this step of our analysis, we developed a tailored Intelligent Query Processing and Information Retrieval System tailored for the app reviews that we have extracted from our scraping process. We started by initially using both the inverted index and a positional index from the review’s dataset, that we have performed in some of our first steps. Then we proceeded with utilizing the TfidfVectorizer. In order to transform the contents of our reviews into a TF-IDF matrix, but also to give emphasis on the importance that words have in the entire collections of the reviews, by trying to balance their frequency against the entire corpus. To move on we tried to enhance our query effectiveness through designing a simplified query let’s say expansion function, whose purpose would be to add extra synonyms to the original query, managing this way to broad the scope of our search based on the query that the user has granted. Adding on, we also developed a Boolean search approach, as to retrieve reviews that will contain any of the terms of the prompted query, following firstly this process which if it fails to find any matched through this expansion we then turn back and search for the original user query. After those steps, the documents are ranked based on how relevant they are with the query, which is calculated through the computation of the cosine similarity between the vector of the inserted query and the documents one’s in the TF-IDF matrix. On top of that, we have built a function or a mechanism if you like in order to handle the cases where no matching of documents is being done, ensuring that way that our Python code snippet will structure and response and return something through many different situations and circumstances. For demonstration purposes we're executed a query “great camera quality” which showcases how relevant reviews are retrieved based on not only their content but also on their sentiment, quits in other words validate the effectiveness and the efficiency of our system, working let's say in a much smaller scale like an informational retrieval system. Here are the results of the aforementioned query:

A screenshot of a computer program

Description automatically generated

# Parallel approach with BERT (Bidirectional Encoder Representations from Transformers)

In parallel to our approach above, we also checked for utlizing pretrained models on the Hugging Face platform.

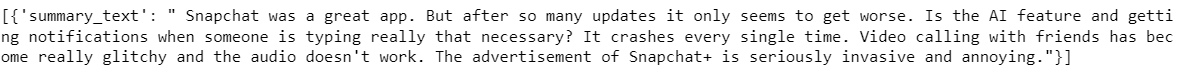
**Zero-shot classification**

*Summarization*

Usually, a user after noting the overall app rating, may scroll through app reviews to get a more detailed of customer experiences. Simple summarization may possibly substantially curtail this long scan of reviews.

We attempted simple summarization with the BART (Bidirectional Auto-Regressive Transformers) model viz **facebook/bart-large-cnn**

This was utilised with a subset of reviews corresponding to the Snapchat app as an example. The output is as below:



We note that though we scraped reviews in a balanced manner to reduce bias so as to avoid either too positive or negative opinions, that the summary ifrom Bart is overall negative in sentiment.

*Question & Answering*

A user may also like to check a particular aspect within the reviews or to query them directly.

Considering the large size of the context to be inputted to answer the user’s single query , we found two appropriately trained models on Hugging Face and posed two queries:

1. “camera quality ” - *that we used in our tested IR system earlier*.

b) “Does snapchat have issues for sharing photos with friends?”

**roberta-base-squad2** : This Question-Answering model is set up by Deepset (which is also the responsible entity for the open source NLP network: Haystack) and hosted on Hugging Face. It is fine-tuned using the SQuAD2.0 dataset i.e. to address ‘unanswerable’ questions

“camera quality ”



“Does snapchat have issues for sharing photos with friends?”



**distilbert-base-cased-distilled-squad:** This model is a fine-tuned version of DistilBERT model trained by distilling BERT base (40% less parameters than bert-base-uncased). Fine tuned on the SQuAD v1.1 dataset

“camera quality ”



“Does snapchat have issues for sharing photos with friends?”



*Sentiment Analysis*

Since many platforms and individuals share trained models for public use, one way to understand or predict sentiment in a new set of data, while conserving time and resourcee

is to use zero-shot classification and infer sentiment from pre-trained models.The Hugging Face platform provides 3 such basic pretrained models for sentiment analyis viz.

* Twitter-roberta-base-sentiment a roBERTa model trained on ~58M tweets
* Bert-base-multilingual-uncased-sentiment fine-tuned for sentiment analysis on product reviews in 6 languages (English, Dutch, German, French, Spanish, Italian)
* Distilbert-base-uncased-emotion fine-tuned for emotion detection in textual input: sadness, joy, love, anger, fear, surprise.

In our case since our dataset refers to app reviews, the **Bert-base-multilingual-uncased-sentiment** is most appropriate.

We perform zero-shot inference on a randomly sample subset set of 50 reviews.

The predictions of the ratings in our dataset from 1 to 5 are hit or miss.

Poor Precision and recall metrics are noted:

Precision: 0.26329365079365075

Recall: 0.26

Hugging Face also provides the **Bert-base-multilingual-uncased-sentiment model fine-tuned on amazon customer reviews** (provide by user:Yi Luan)

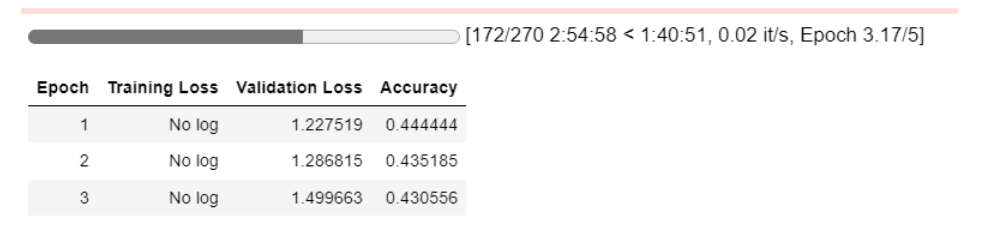
Similarly with 50 randomly selected reviews, we determine precision and recall. Precision appears slighty improved by this model:

Precision: 0.4294475524475525

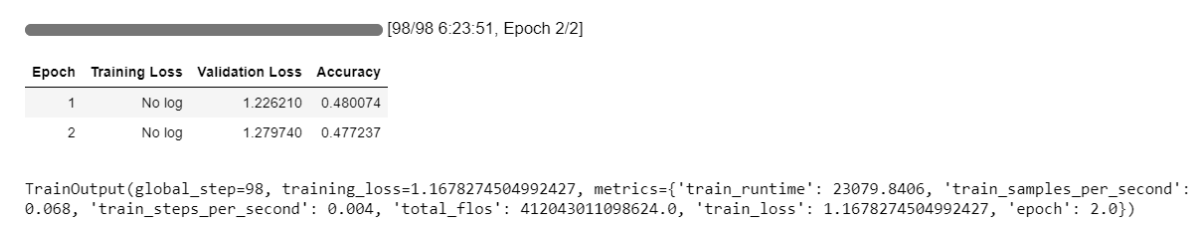
Recall: 0.28

It appears we can fine tune it further on our dataset. However, first attempts via the trainer tool also from Hugging Face demonstrate poor training accuracy on both the subset of Snapchat reviews and the whole dataset. This dataset may be not suitable for this type of model or may need further specific adjustments for the training, test , validation split which may be explored in the future.

Poor Training accuracy on Snapchat Subdataset:

¶

Poor Training accuracy on Whole dataset:



# Conclusion

Through the entire set of steps performed throughout our project, and the final combination of them, we were able to create a strong blend of natural language process and machine learning techniques. This analysis could play a vital role in the roots of an IR system that in its final form will aim at making targeted improvements and personalized recommendation systems. The importance of understanding customers feedback in an automatic way can lead to the raise of the industry, by increasing the downloads of the apps, keeping the owners of the applications updated in a constant daily manner about problems and bugs that they may come up, causing them to act proactively when it comes to deal with a problem but also towards trend identification, having the ability to ring a bell whether this notification would be positive, negative or even neutral trend. In terms of future work, we may also further explore utilization of pre-retrained models in more detail.

# References

Aliyu, M. B. (2017). Efficiency of Boolean search strings for information retrieval. *American Journal of Engineering Research, 6*(11), 216-222.

Church, K. W. (2017). Word2Vec. *Natural Language Engineering, 23*(1), 155-162.

Do, H. H., Prasad, P. W., Maag, A., & Alsadoon, A. (2019). Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Systems with Applications, 118*, 272-299.

Farkiya, A., Saini, P., Sinha, S., & Desai, S. (2015). Natural language processing using NLTK and WordNet. *International Journal of Computer Science and Information Technology, 6*(6), 5465-5469.

Feng, J., Cai, S., & Ma, X. (2019). Enhanced sentiment labeling and implicit aspect identification by integration of deep convolutional neural network and sequential algorithm*. Cluster Computing, 22*, 5839-5857.

Fletcher, S., & Islam, M. Z. (2018). Comparing sets of patterns with the Jaccard index. *Australasian Journal of Information Systems, 22*.

Google Play Scraper. (2023). [Python library]. Retrieved from https://pypi.org/project/google-play-scraper/: <https://pypi.org/project/google-play-scraper/>

He, J., & Suel, T. (2012, August). Optimizing positional index structures for versioned document collections. *In Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 245-254).

Huang, C. H., Yin, J., & Hou, F. (2011). A text similarity measurement combining word semantic information with TF-IDF method. *Jisuanji Xuebao (Chinese Journal of Computers), 34*(5), 856-864.

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2019). Latent Dirichlet Allocation (LDA) and topic modeling: Models, applications, a survey*. Multimedia Tools and Applications, 78*, 15169-15211.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques*. Emerging Artificial Intelligence Applications in Computer Engineering, 160*(1), 3-24.

Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015, June). From word embeddings to document distances. *International Conference on Machine Learning* (pp. 957-966). PMLR.

McGregor, A., Hall, M., Lorier, P., & Brunskill, J. (2004). Flow clustering using machine learning techniques. *In Passive and Active Network Measurement: 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004. Proceedings 5* (pp. 205-214). Springer Berlin Heidelberg.

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal, 5*(4), 1093-1113.

Pibiri, G. E., & Venturini, R. (2020). Techniques for inverted index compression. *ACM Computing Surveys (CSUR), 53*(6), 1-36.

Robertson, S. E., & Hancock-Beaulieu, M. M. (1992). On the evaluation of IR systems*. Information Processing & Management, 28*(4), 457-466.