Semantic Search problem:

- ## **Introduction:**

  The problem of semantic search in articles using NLP is to accurately match user queries to relevant articles based on their meaning, rather than just keywords or metadata. Keyword-based search can be limiting, as it may not account for variations in terminology or contextual meaning, leading to poor results. Semantic search addresses this issue by leveraging NLP techniques to understand the meaning and intent behind user queries, and to match them to articles that contain relevant information, even if the wording is different. However, semantic search is a complex problem that requires advanced techniques such as word embeddings, topic modeling, and named entity recognition to extract the most relevant information from articles and map it to user queries. One challenge is to ensure that the search results are accurate and comprehensive, while also being efficient enough to handle large volumes of data. Another challenge is to continuously improve the search algorithm by incorporating user feedback and updating the NLP models to better understand the context and nuances of natural language.

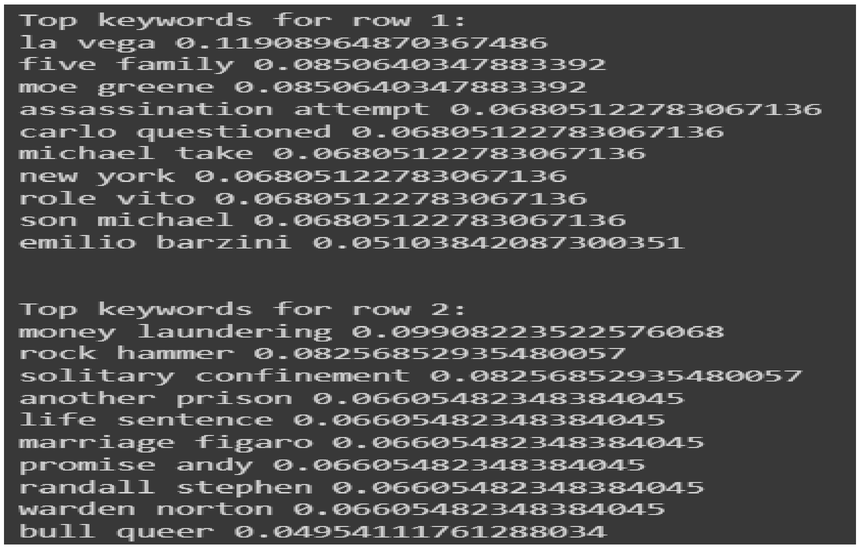
- ## **Data description:**

  The data contains two datasets:

  - The first one is about movies which consist of 5 features and 100 rows.
  - The second one is about medium articles which consist of 8 features and 337 rows.

- ## **Baseline experiments:**

  The goal of the baseline experiments is to establish a baseline performance for the semantic search algorithm. The first phase is data preprocessing that takes the input and converting it to lowercase, removing punctuation, removing numbers, tokenizing the text, removing stop words, lemmatizing the words, and joining the tokens back into a string. In the second phase, taking the preprocessed document and uses the TF-IDF Vectorizer to transform the document into a TF-IDF matrix. After this converting the TF-IDF matrix into a dictionary of keywords and returns the top hot keywords in each row sorted by their TF-IDF score. This is as shown in the fig.1 below.

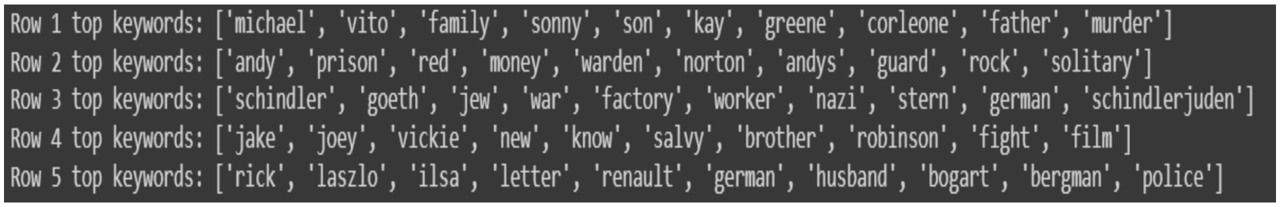


```
Top keywords for row 1:
la vega 0.11908964870367486
five family 0.0850640347883392
moe greene 0.0850640347883392
assassination attempt 0.06805122783067136
carlo questioned 0.06805122783067136
michael take 0.06805122783067136
new york 0.06805122783067136
role vito 0.06805122783067136
son michael 0.06805122783067136
emilio barzini 0.05103842087300351


Top keywords for row 2:
money laundering 0.09908223522576068
rock hammer 0.08256852935480057
solitary confinement 0.08256852935480057
another prison 0.06605482348384045
life sentence 0.06605482348384045
marriage figaro 0.06605482348384045
promise andy 0.06605482348384045
randall stephen 0.06605482348384045
warden norton 0.06605482348384045
bull queer 0.04954111761288034
```

Fig.1. The top hot keywords in each row sorted by their TF-IDF score.

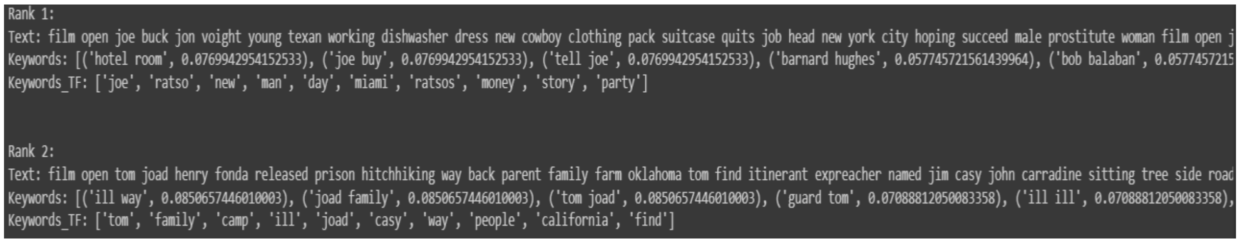
Another method for the second phase for extracting the hot keywords is using CountVectorizer object with an n-gram to extract the top 10 most frequent keywords in each text using the trained CountVectorizer. **Fig.2**



```
Row 1 top keywords: ['michael', 'vito', 'family', 'sonny', 'son', 'kay', 'greene', 'corleone', 'father', 'murder']
Row 2 top keywords: ['andy', 'prison', 'red', 'money', 'warden', 'norton', 'andys', 'guard', 'rock', 'solitary']
Row 3 top keywords: ['schindler', 'goeth', 'jew', 'war', 'factory', 'worker', 'nazi', 'stern', 'german', 'schindlerjuden']
Row 4 top keywords: ['jake', 'joey', 'vickie', 'new', 'know', 'salvy', 'brother', 'robinson', 'fight', 'film']
Row 5 top keywords: ['rick', 'laszlo', 'ilsa', 'letter', 'renault', 'german', 'husband', 'bogart', 'bergman', 'police']
```

Fig.2. Extract the top 10 most frequent keywords in each text using the trained CountVectorizer

In the last phase, the goal of semantic search on a set of documents based on a query to get the most 5 documents close to the query through using pre-trained word embedding model from the spaCy library, then compute the cosine similarity between set of documents and the query which leads to get the closest meaning for the query on text compared to the set of documents.



```
Rank 1:
Text: film open joe buck jon voight young texan working dishwasher dress new cowboy clothing pack suitcase quits job head new york city hoping succeed male prostitute woman film open j
Keywords: [('hotel room', 0.0769942954152533), ('joe buy', 0.0769942954152533), ('tell joe', 0.0769942954152533), ('barnard hughes', 0.057745721561439964), ('bob balaban', 0.0577457215
Keywords_TF: ['joe', 'ratso', 'new', 'man', 'day', 'miami', 'ratsos', 'money', 'story', 'party']


Rank 2:
Text: film open tom joad henry fonda released prison hitchhiking way back parent family farm oklahoma tom find itinerant expreacher named jim casy john carradine sitting tree side road
Keywords: [('ill way', 0.0850657446010003), ('joad family', 0.0850657446010003), ('tom joad', 0.0850657446010003), ('guard tom', 0.07088812050083358), ('ill ill', 0.07088812050083358),
Keywords_TF: ['tom', 'family', 'camp', 'ill', 'joad', 'casy', 'way', 'people', 'california', 'find']
```

Fig.3. The most 2 documents close to the input text with the hot keywords

- **Other experiments:**

Applying semantic search with another dataset using different pipeline:

At first, after reading the data we need to visualize it by using the word cloud before cleaning the data to know the domain of the articles, the following figure shows the most frequency words in the data.

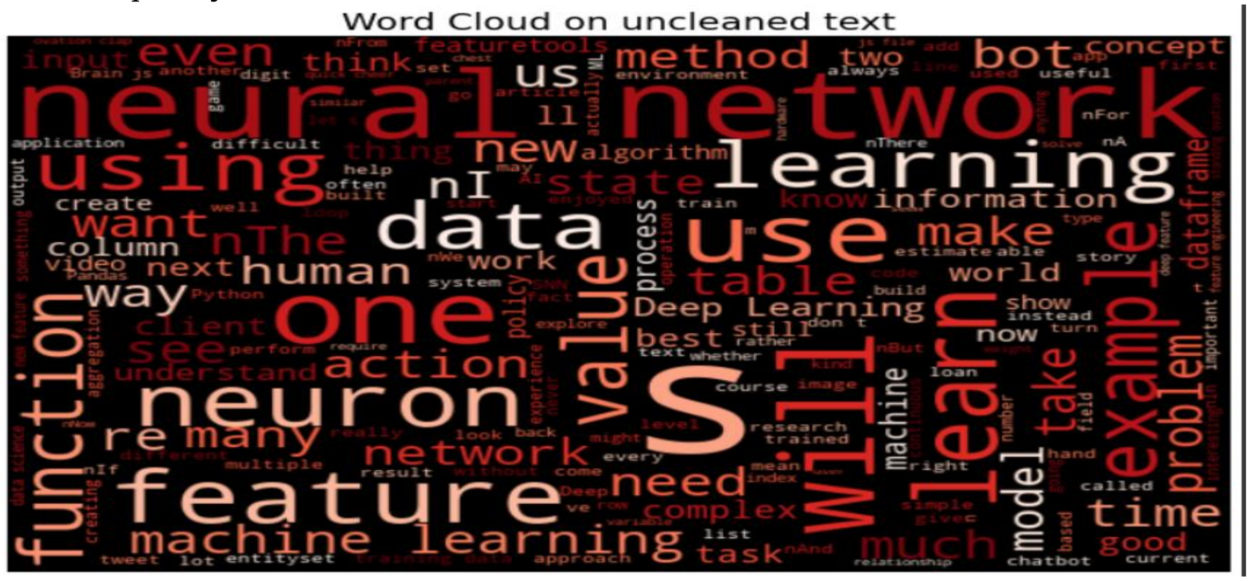


Fig.4. word cloud on the uncleaned data.

After this it is time for applying simple data preprocessing to save the meaning of the scientific domain because the lemmatization will not keep them in their meaning like "Machine Learning", "Deep Learning".

Now, Visualize the data again, after Cleaning the data to provide a quick overview of the most important terms in the text dataset and to help identify themes or patterns.

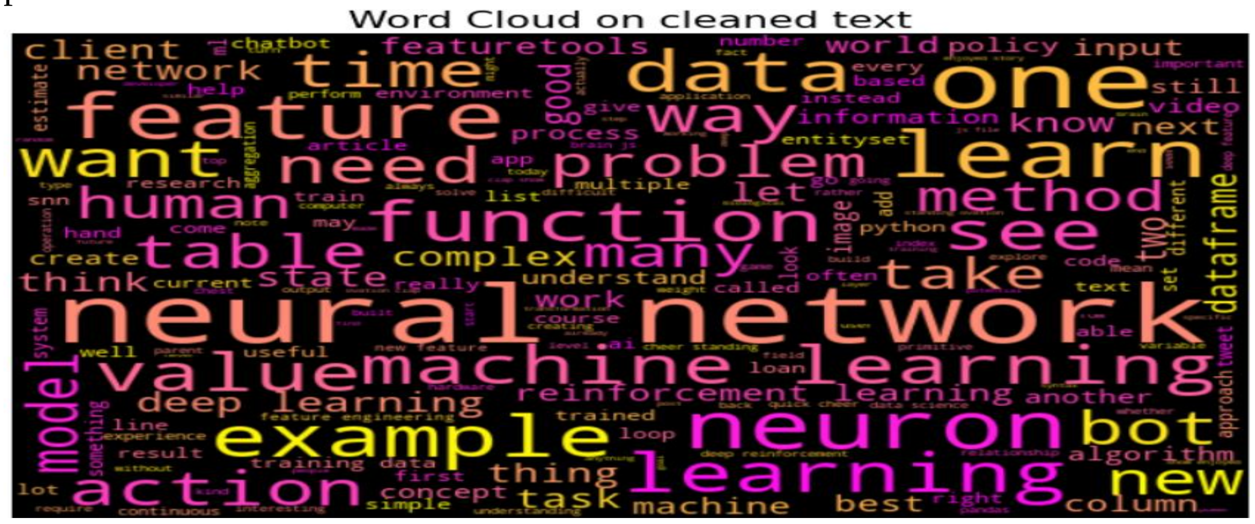


Fig.5. Word cloud on the cleaned data.

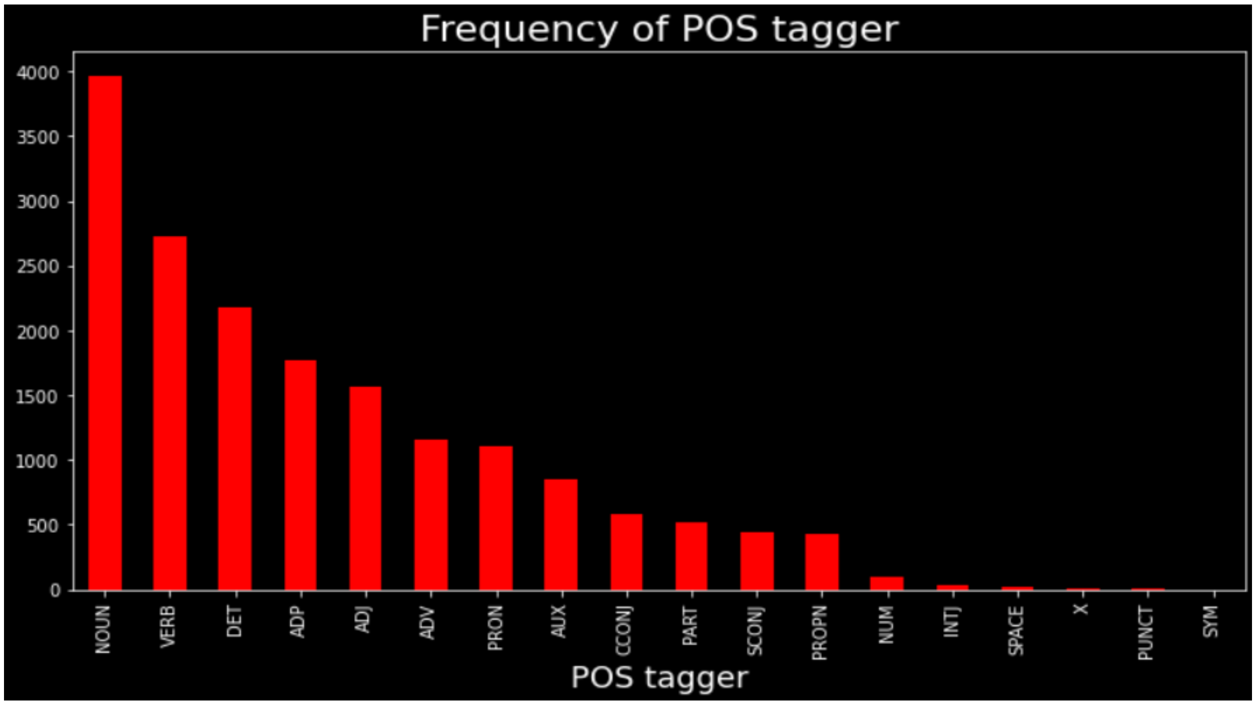


Fig.6*.Frequency of POS tagger in the cleaned data.*

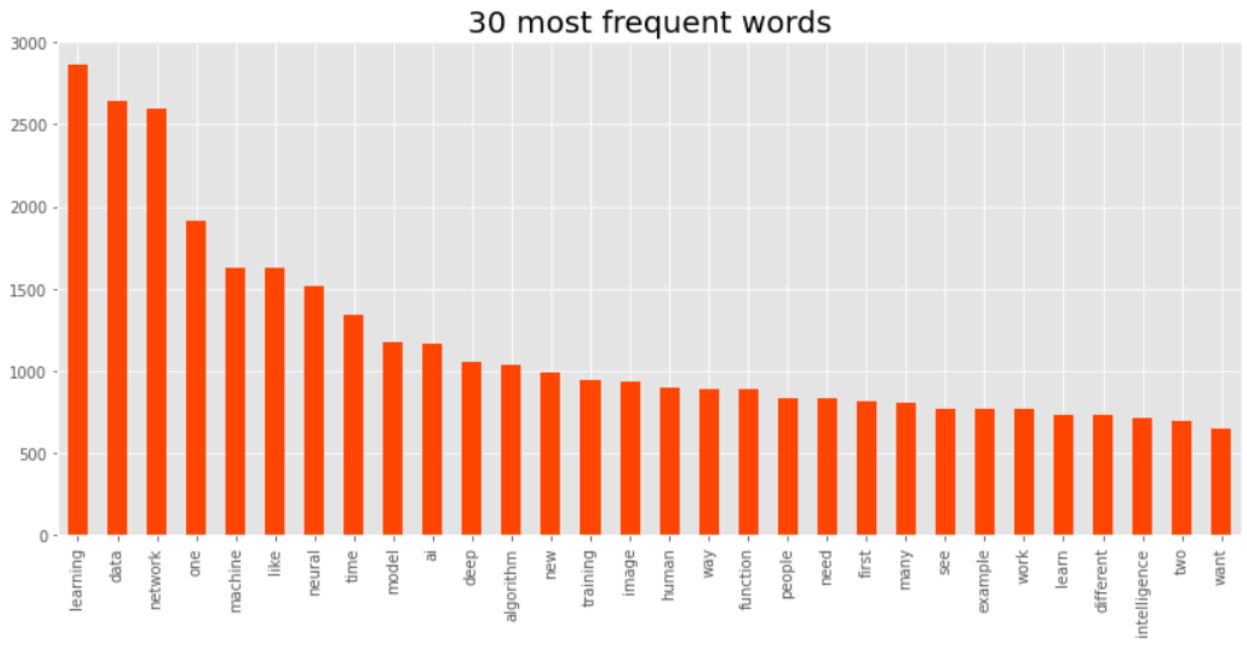


Fig.7. Top 30 most frequent words in the cleaned data.

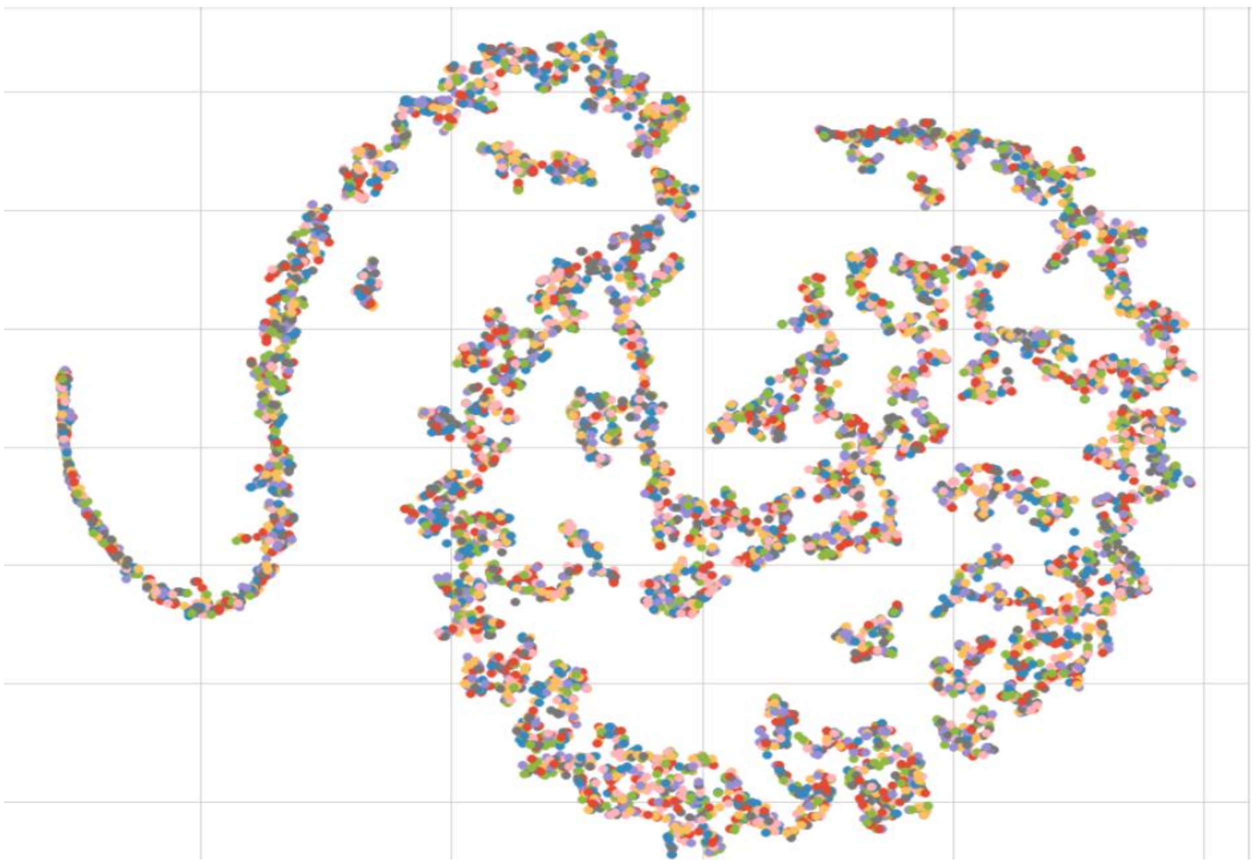


Fig.8. Visualizing high-dimensional data in a low-dimensional space Using t-SNE

Now data is ready to perform word-embedding model and calculate the cosine similarity to words and check the model performance:

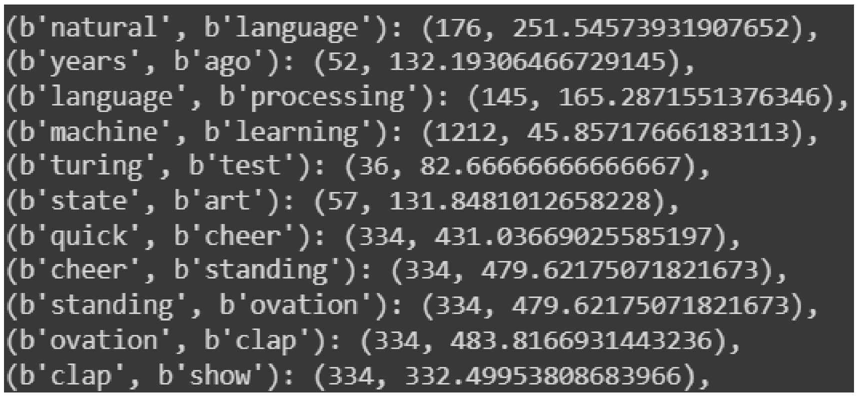
- Glove pretrained model:
  Applying it into two known words “machine” and “Learning” by getting its word-embedding and calculate the cosine similarity. The cosine similarity is very low, 33.11%.

```
0.33105597
```

  That means that the two words are not similar because the Glove pre-trained word-embedding model based on the Wikipedia data is not good enough. And retraining the model on the data used in our case did not help, so that it is better to apply another word-embedding model.

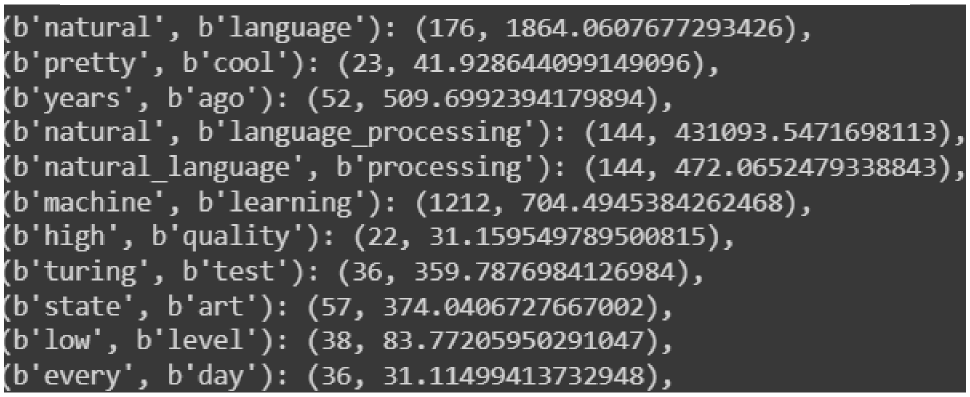
- Fasttext pretrained model:

  Using fasttext pretrained model led to improve the performance after retraining it on the data used in our case. We are going to use bi-gram and tri-gram phrase by identifying common trigram phrases to improve various natural language processing.



```
(b'natural', b'language'): (176, 251.54573931907652),
(b'years', b'ago'): (52, 132.19306466729145),
(b'language', b'processing'): (145, 165.2871551376346),
(b'machine', b'learning'): (1212, 45.85717666183113),
(b'turing', b'test'): (36, 82.66666666666667),
(b'state', b'art'): (57, 131.8481012658228),
(b'quick', b'cheer'): (334, 431.03669025585197),
(b'cheer', b'standing'): (334, 479.62175071821673),
(b'standing', b'ovation'): (334, 479.62175071821673),
(b'ovation', b'clap'): (334, 483.8166931443236),
(b'clap', b'show'): (334, 332.49953808683966),
```

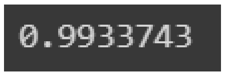
Fig.9. bi-gram



```
(b'natural', b'language'): (176, 1864.0607677293426),
(b'pretty', b'cool'): (23, 41.928644099149096),
(b'years', b'ago'): (52, 509.6992394179894),
(b'natural', b'language_processing'): (144, 431093.5471698113),
(b'natural_language', b'processing'): (144, 472.0652479338843),
(b'machine', b'learning'): (1212, 704.4945384262468),
(b'high', b'quality'): (22, 31.159549789500815),
(b'turing', b'test'): (36, 359.7876984126984),
(b'state', b'art'): (57, 374.0406727667002),
(b'low', b'level'): (38, 83.77205950291047),
(b'every', b'day'): (36, 31.11499413732948),
```

Fig.10.tri-gram

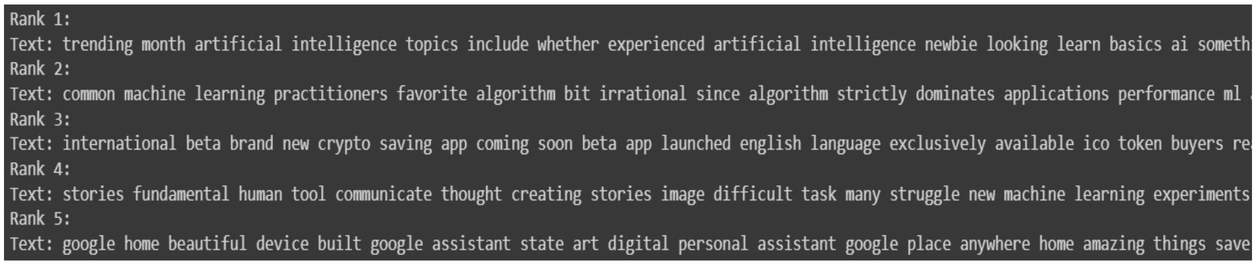
  Applying it into the same two words “machine” and “Learning” by getting its word-embedding and calculate the cosine similarity. The cosine similarity is very good, 99.34%.



```
0.9933743
```

After making sure that fasttext performs well as word embedding for the dataset, now it is time for performing the Semantic Search with the following query:

query= "I love machine learning and natural language" the result of the search shows in the following figure (fig 11), it shows the most 5 text close to the query:



```
Rank 1:
Text: trending month artificial intelligence topics include whether experienced artificial intelligence newbie looking learn basics ai someth
Rank 2:
Text: common machine learning practitioners favorite algorithm bit irrational since algorithm strictly dominates applications performance ml
Rank 3:
Text: international beta brand new crypto saving app coming soon beta app launched english language exclusively available ico token buyers re
Rank 4:
Text: stories fundamental human tool communicate thought creating stories image difficult task many struggle new machine learning experiments
Rank 5:
Text: google home beautiful device built google assistant state art digital personal assistant google place anywhere home amazing things save
```

Fig.11. The result of the search (the most 5 text close to the query)

1) What was the biggest challenge you faced when carrying out this project?

The biggest problem with hot keyword extraction is that it is often context-independent, which means that it can produce inaccurate or irrelevant results. However, this approach does not take into the specific domain in which the text is being analyzed there are many techniques to extract it depending on the domain of data.

The major challenges of semantic search are Ambiguity, Synonymy, Polysemy, Contextual variations and user intent .it can be solved using word embedding model so it must select the best word embedding model to make sure it returns the best similar text to query.

2) What do you think you have learned from the project?

I learned many techniques for hot keyword extraction the first technique considers the most frequent word is the hotkeyword and other depend on term uniqueness in document considered it as keyword. And in preprocessing data not all data adapt with same cleaning and lemmatization process because prepossessing depend on the domain of data . and in the word embedding there are many models you should try them and select best on that fit with the data based on their similarity between words

• Conclusion:

Extracting the hot keywords from a corpus of text is a valuable process that can help to identify the most significant and relevant topics discussed in a particular text. This process can be useful for a variety of applications, including search engine optimization (SEO), content creation, and market research. Overall, the process of extracting hot keywords is a powerful tool that can help businesses and organizations better understand their audience and create more effective content strategies. Semantic search is a powerful technique for improving search results by

understanding the meaning and intent behind search queries enabling computers to understand the nuances of human language and identify the most relevant content for a given query. As NLP technology continues to advance, we can expect semantic search to become even more sophisticated and effective in the years to come, revolutionizing the way we search for information online.
by trying to solve this problem with two ways with different two datasets (movies and medium articles dataset), the first way using movies dataset is dividing the problem into 3 phases First phase is applying preprocessing on data , second phase is extracting hot keywords using TFIDF and TF .and for the last phase getting the semantic Search by applying Spacy pertained model for word embedding but the way here to calculate the accuracy depend on the human feedback.
in the second way and with using the second dataset by applying two other pretrained model (Glove and fasttext) in known domain of data after simple preprocessing step and calculating the similarity between words was too good by using fasttext
Determining which hot keyword extraction and semantic search Based on the domain of data.

- **References:**

- https://towardsdatascience.com/keyword-extraction-process-in-python-with-natural-language-processing-nlp-d769a9069d5c
- https://www.kaggle.com/code/ajitrajput/semantic-search-engine-using-nlp/notebook