TEXT ANALYTICS
ASSIGNMENT 1 REPORT
ERP : 25371

Objective

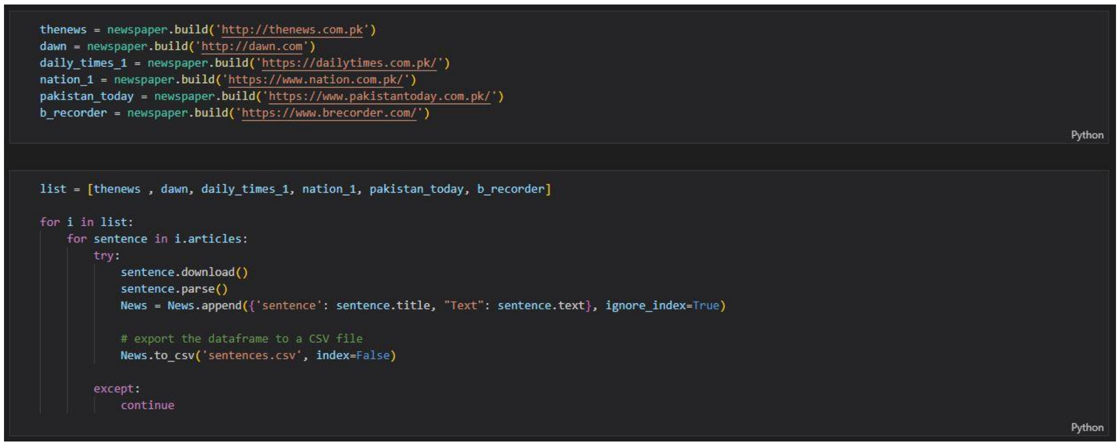
The objective of this assignment is to train various text analytics models, such as Bag of Words, Word2Vec, Glove and Singular Value Decomposition (SVD), and develop an application that will take input from the user in the form of a sentence and return similar sentences related to it based on similarity. Our aim is to implement these models and evaluate their performances to provide a comprehensive understanding of their capabilities and limitations in handling natural language data. The project aims to demonstrate how the selection of the appropriate NLP model can play a vital role in enhancing the quality of results.

Methodology

We are using five different Natural language models which are Bag of words, word2vec (Pretrained), Glove (Pretrained), customized word2vec, and SVD. We are also trying different combinations of these model by performing hyperparameter tuning to get the best possible results. To check out the performance of these models and quality of the embeddings created we are performing clustering and calculating the silhouette score of the clusters to measure the performance of clustering. We are mainly using 3 types of clustering in this assignment which are K-Mean, Mini Batch K mean and Agglomerative clustering. PCA is also implemented so that we can visualize our clusters on the scatter plot.

Data Collection

Data which we are using in this project are headlines and summary text of different Pakistani newspapers which are collected using NewsApi of python. We have collected data for several days and the final size of data which I have got is (3328,2). Which means we have 3328 number of different headlines and there are two columns in the data. One is Sentences which are headlines and other one is Text. Which contains summary text of the headlines.



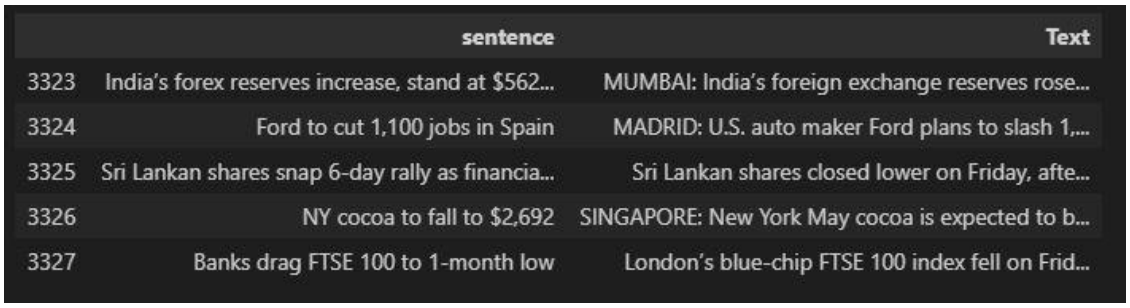
```python
thenews = newspaper.build('http://thenews.com.pk')
dawn = newspaper.build('http://dawn.com')
daily_times_1 = newspaper.build('https://dailytimes.com.pk/')
nation_1 = newspaper.build('https://www.nation.com.pk/')
pakistan_today = newspaper.build('https://www.pakistantoday.com.pk/')
b_recorder = newspaper.build('https://www.brecorder.com/')
```

```python
list = [thenews , dawn, daily_times_1, nation_1, pakistan_today, b_recorder]

for i in list:
    for sentence in i.articles:
        try:
            sentence.download()
            sentence.parse()
            News = News.append({'sentence': sentence.title, "Text": sentence.text}, ignore_index=True)

            # export the dataframe to a CSV file
            News.to_csv('sentences.csv', index=False)

        except:
            continue
```



	sentence	Text
3323	India’s forex reserves increase, stand at $562...	MUMBAI: India’s foreign exchange reserves rose...
3324	Ford to cut 1,100 jobs in Spain	MADRID: U.S. auto maker Ford plans to slash 1,...
3325	Sri Lankan shares snap 6-day rally as financia...	Sri Lankan shares closed lower on Friday, afte...
3326	NY cocoa to fall to $2,692	SINGAPORE: New York May cocoa is expected to b...
3327	Banks drag FTSE 100 to 1-month low	London’s blue-chip FTSE 100 index fell on Frid...

Natural Language Models

We are using 5 different models to create embeddings of the news headlines and checking out the results of these approaches by checking out the similarity and clustering. The models which we are using are:

1- Bag of Words.
2- Word2Vec (Pre – Trained)
3- Glove (Pre-Trained)
4- Customized Word2Vec
5- LSA/SVD.

Bag of Words (BOW)

The first model we are using is Bag of word. It is a technique in natural language processing (NLP) that represents a piece of text as a collection (or "bag") of its individual words, disregarding grammar and word order but keeping track of the frequency of each word. In this project, we are using BOW with different combinations like first we are using stemming in preprocessing and forming TF-IDF Vector, then we are using stemming and creating binary TF-IDF vector. After that we are using lemmatization technique and forming TF-IDF and binary TF-IDF vectors.

With each combination we are also implementing clustering and checking out the result of the model by measuring silhouette score. So, following are the result of bag of word model and its combinations.

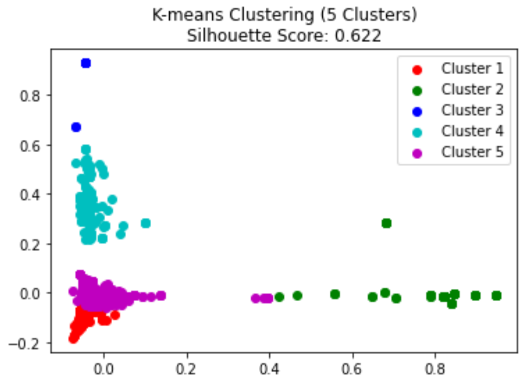


Image 1: K-Mean Clustering with Stemming (Binary)

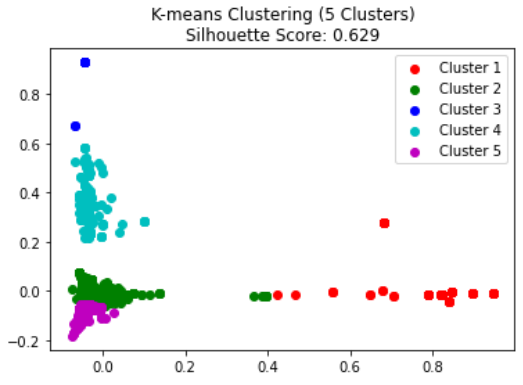


Image 2: K-Mean with Stemming (TFIDF).

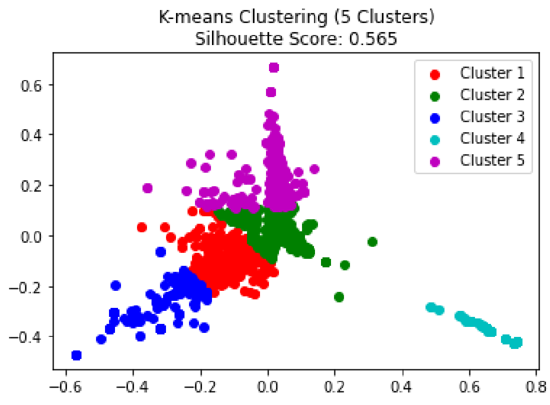


Image 3: K-Mean Clustering with Lemmatization. `

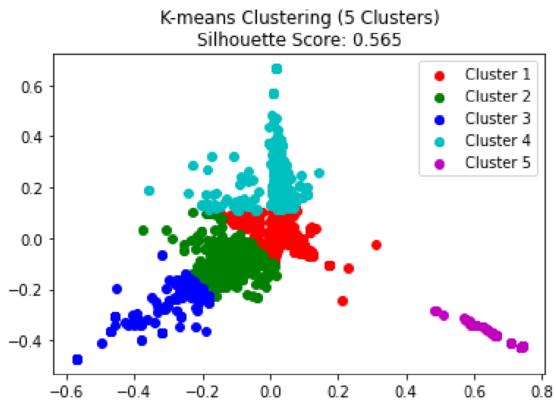


Image 4: K-Mean with Lemmatization (Binary TFIDF).

So, as we can see from the results of clustering, BOW is best working with stemming so now we are testing its clustering by implementing different number of clusters and trying out different clustering techniques like K-Mean, Mini batch K-mean and agglomerative clustering.

Hyper Parameter Tuning on BOW

We are performing some hyperparameter tuning on BOW model to tune up the model and increase its performance by finding best parameters. We have used unigrams, bigrams, and trigrams and creating tf-idf vectorizer with each one. We are also using max_df and min_df parameter.

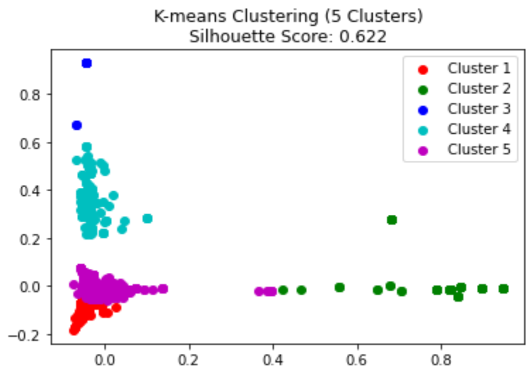


Image 5: K-Mean with Bi Gram

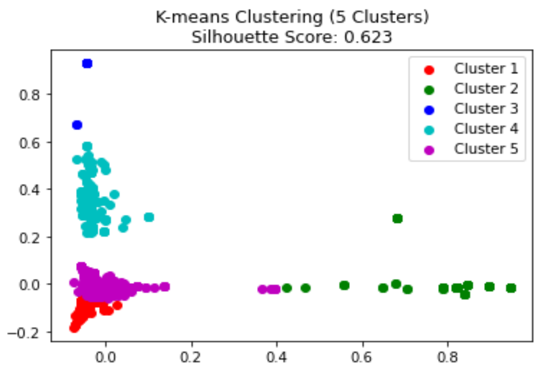


Image 6 : K-Mean with Tri Gram

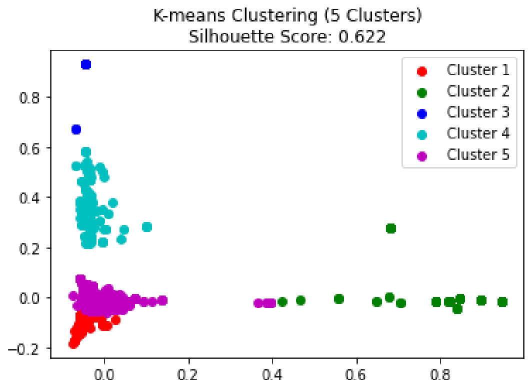


Image 7: K-Mean with Max_DF = 2

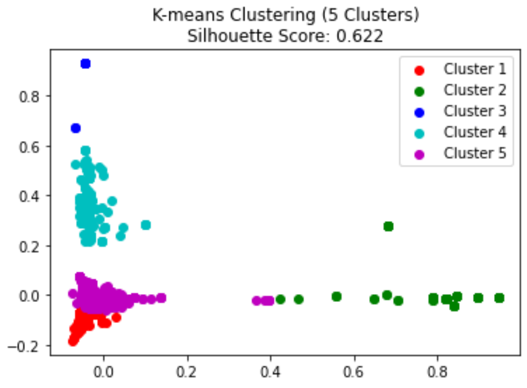


Image 6 : K-Mean with Max_DF = 3

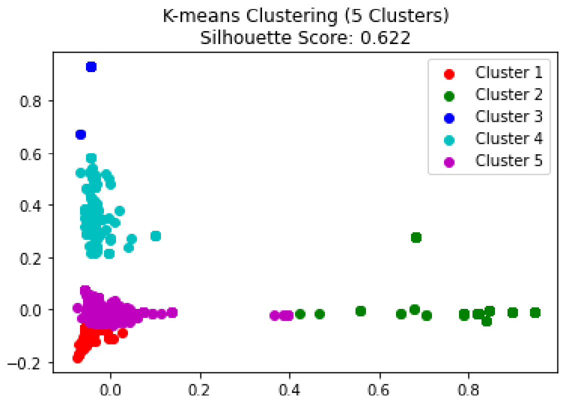


Image 7: K-Mean with Min_DF = 2

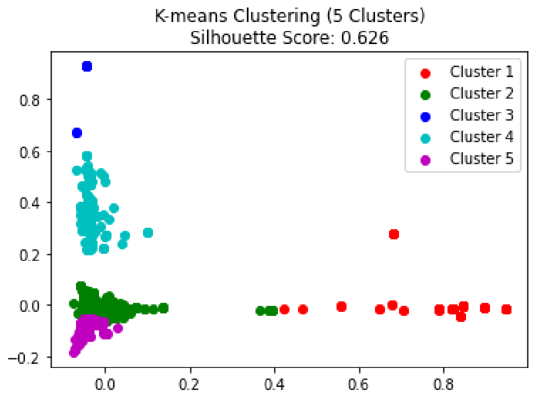


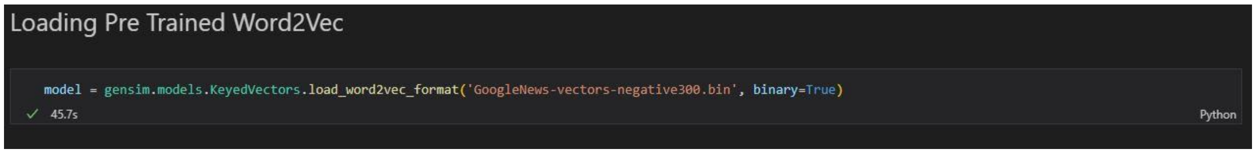
Image 6 : K-Mean with Min_DF = 3

As we can see, changing parameter is not much affecting the clustering results. We are getting almost same result after changing bigram and trigram or max_df and min_df.

Word2Vec (Pre – Trained Model)

After bag of words, we are implementing word 2 pre-trained model. We are using model of google news. The Google News Word2Vec model is a pre-trained word embedding model that was trained on a large corpus of news articles. This model contains embeddings for 3 million words and phrases, and the embeddings are 300-dimensional vectors.

The Google News Word2Vec model is available for download from the Google Code Archive. Once downloaded, the model can be loaded into Python using the Genism library.

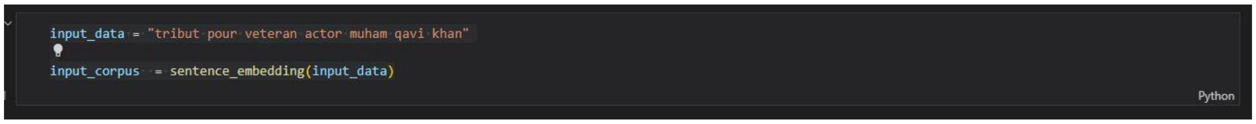


Loading Pre Trained Word2Vec

```python
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)
```

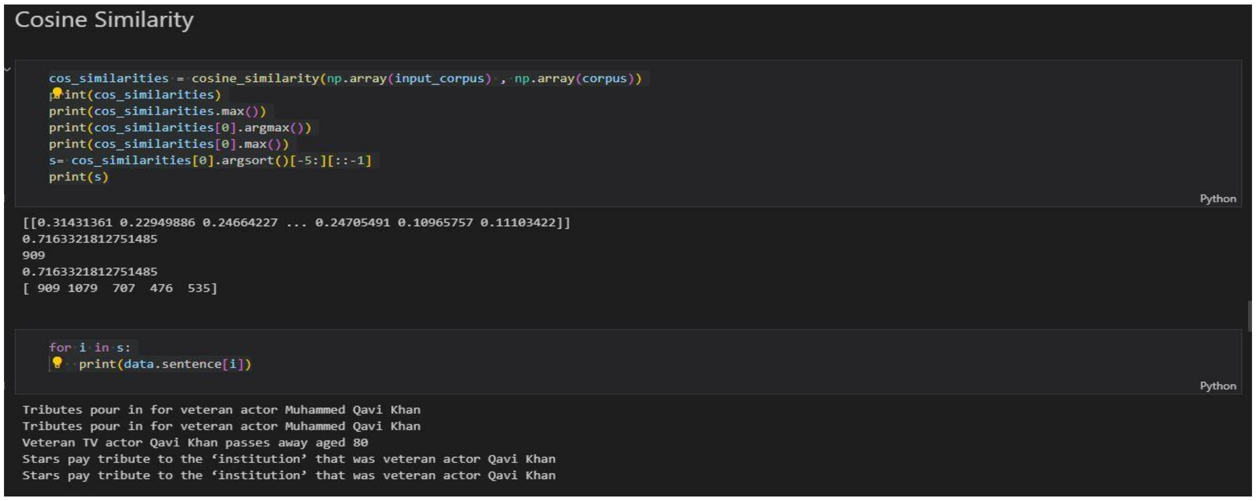
✓ 45.7s

After loading this model, we are implementing this model on our dataset and creating sentence embedding using this. Once sentence embeddings are created we are calculating cosine similarities by giving a text input and it will return most similar sentences.



```python
input_data = "tribut pour veteran actor muham qavi khan"

input_corpus  = sentence_embedding(input_data)
```



Cosine Similarity

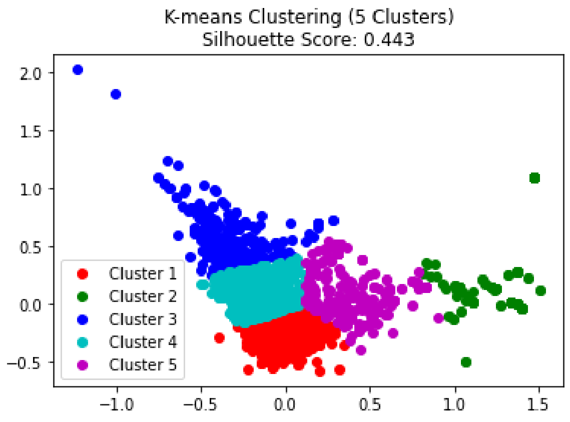
```python
cos_similarities = cosine_similarity(np.array(input_corpus) , np.array(corpus))
print(cos_similarities)
print(cos_similarities.max())
print(cos_similarities[0].argmax())
print(cos_similarities[0].max())
s= cos_similarities[0].argsort()[-5:][::-1]
print(s)
```

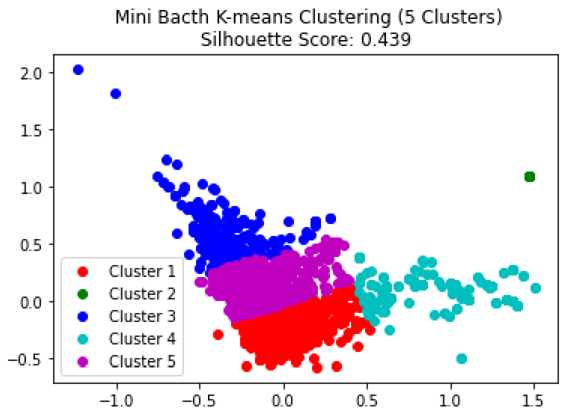
```
[[0.31431361 0.22949886 0.24664227 ... 0.24705491 0.10965757 0.11103422]]
0.7163321812751485
909
0.7163321812751485
[ 909 1079  707  476  535]
```

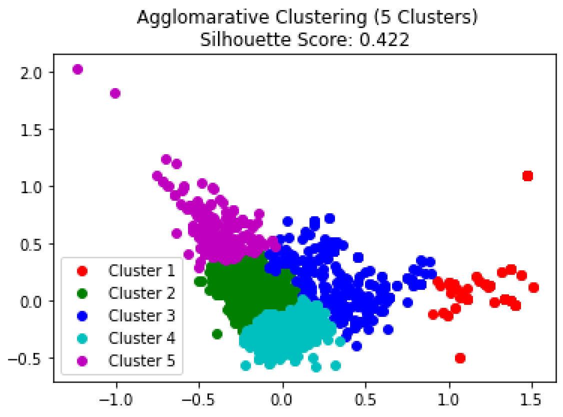
```python
for i in s:
    print(data.sentence[i])
```

```
Tributes pour in for veteran actor Muhammed Qavi Khan
Tributes pour in for veteran actor Muhammed Qavi Khan
Veteran TV actor Qavi Khan passes away aged 80
Stars pay tribute to the ‘institution’ that was veteran actor Qavi Khan
Stars pay tribute to the ‘institution’ that was veteran actor Qavi Khan
```

We are also implementing clustering to check out how well our results got clustered together. The results of the clustering are following. We are using techniques of clustering out of which K-Mean clustering is performing better on 5 clusters.





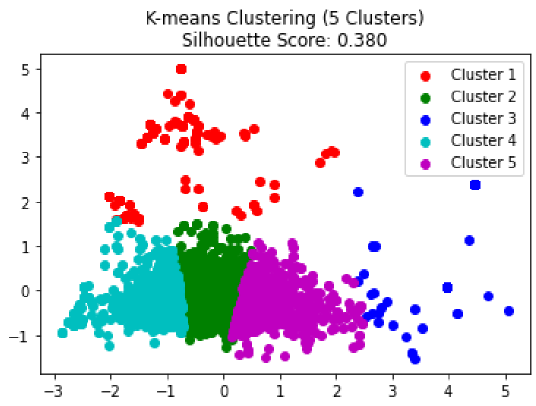


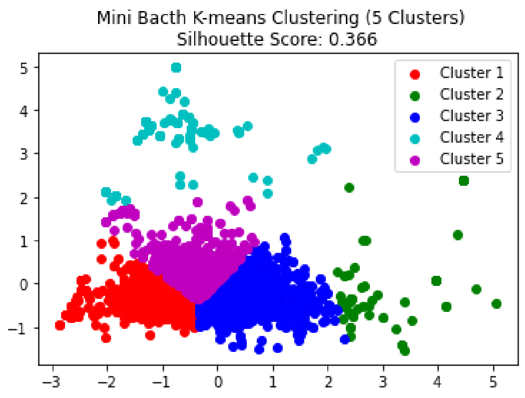
Glove (Pre-Trained Model)

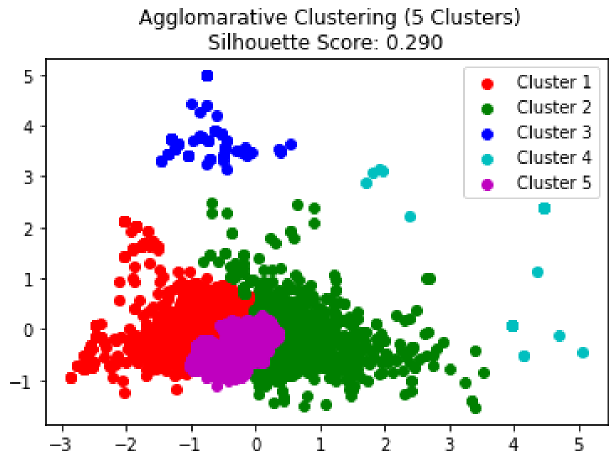
GloVe (Global Vectors for Word Representation) is another popular algorithm for learning word embeddings, developed by researchers at Stanford University. The pre-trained GloVe models are pre-built models that have already been trained on large corpora of text, and they can be used to generate word embeddings for new text data.

The pre-trained GloVe models are available for download from the Stanford NLP Group website. There are several different pre-trained models available, ranging in size but we are using 300 vector size model with file size of 4.6 GB.

We are loading a glove file and creating sentence embedding using the glove model. Once the sentence embeddings are created, we are computing cosine similarity to the performance of the embeddings created using GloVe model. We are also implementing clustering to check out how well our results got clustered together. The results of the clustering are following. We are using techniques of clustering out of which K-Mean clustering is performing better on 5 clusters.

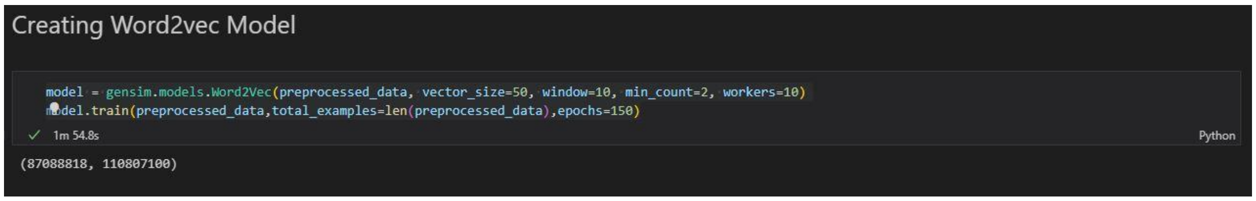






Customized Word2Vec Model (CBOW)

Customized model of word2vec is the one which we are creating and training by setting hyperparameters with our choice. Its also same model as pre trained word2vec but in this we are setting parameters and training the model. Model can be created using genism model library in python.



Creating Word2vec Model

```python
model = gensim.models.Word2Vec(preprocessed_data, vector_size=50, window=10, min_count=2, workers=10)
model.train(preprocessed_data,total_examples=len(preprocessed_data),epochs=150)
```

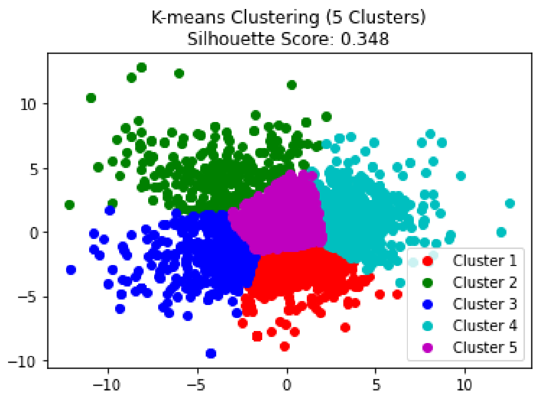
✓ 1m 54.8s Python

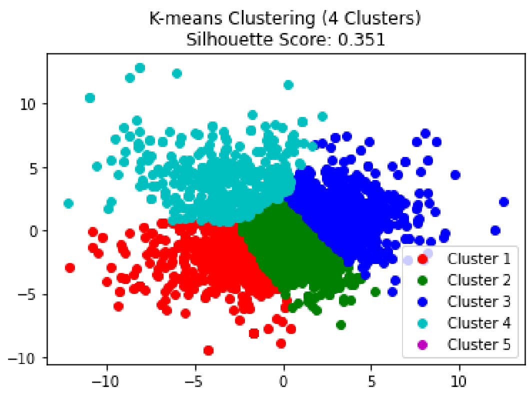
```
(87088818, 110807100)
```

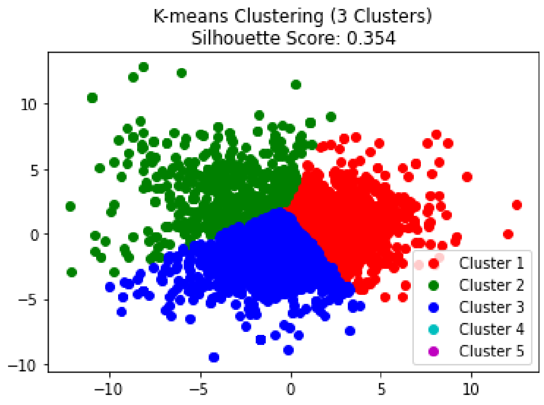
We are creating a word2vec model and using that we are creating sentence embedding using that model. Once the sentence embeddings are created, we are computing cosine similarity to the performance of the embeddings created using GloVe model. We are also implementing clustering to check out how well our results got clustered together. At fist we are using CBOW model and after that we are using Skip gram model to check the performance.

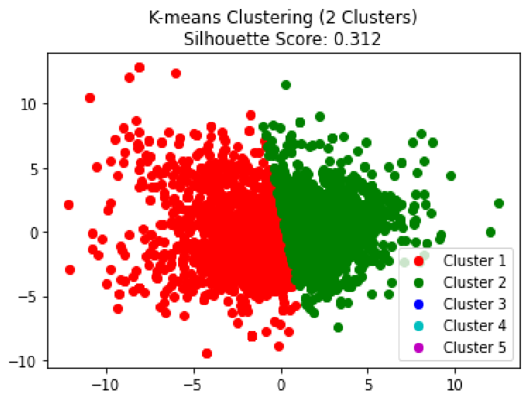
Clustering in Customized Word2Vec (CBOW Model)

1- K – Mean Clustering



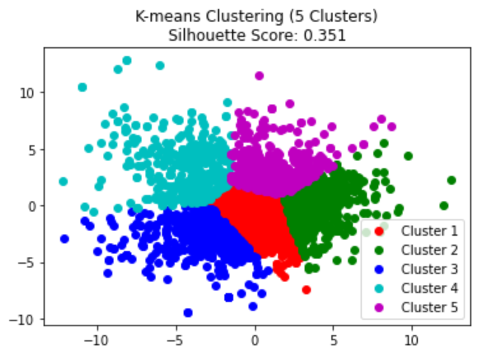


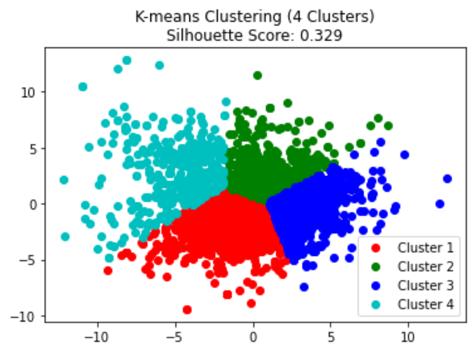


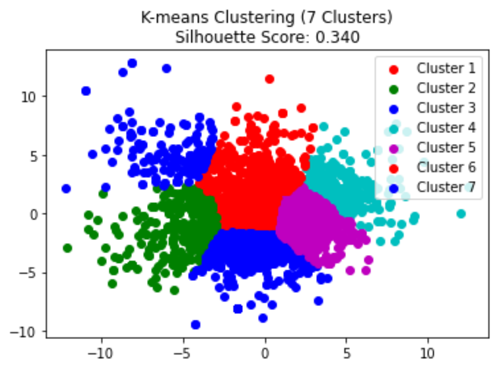


2- Mini Batch K-Mean Clustering

In Mini Batch K Mean Clustering we are getting our best results on 5 number of clusters.





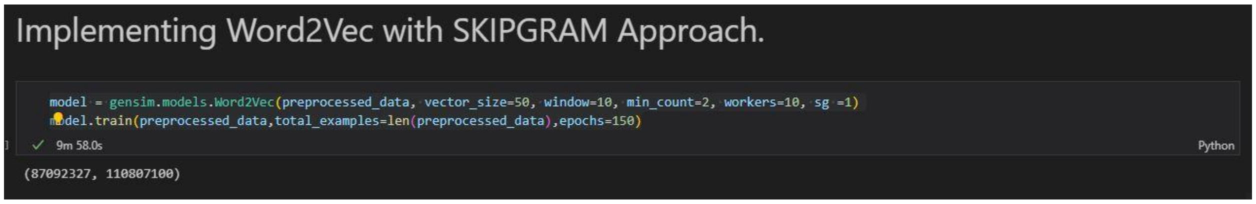


Now we are performing some Hyper Parameter tuning and changing our model from CBOW to SKIPGRAM. We are using different approaches of tuning hyperparameters like changing vector size, number of workers etc.

Customized Word2Vec (SKIPGRAM)

The skip-gram approach is a method in Word2Vec where the model predicts the context words based on the target word. The goal of the skip-gram model is to learn the probability distribution of each context word given a target word, i.e., to maximize the probability of the context words given the target word. In this approach, the input to the model is a target word, and the output is a set of context words that are likely to appear in the same context as the target word.

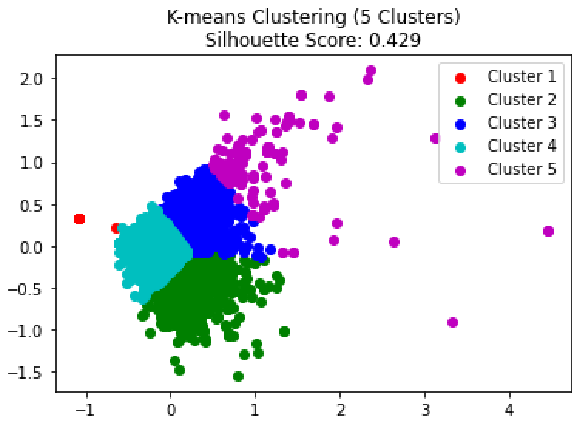
Skipgram model can be implemented by adding parameter SG =1 which means skipgram enabled model.


Implementing Word2Vec with SKIPGRAM Approach.

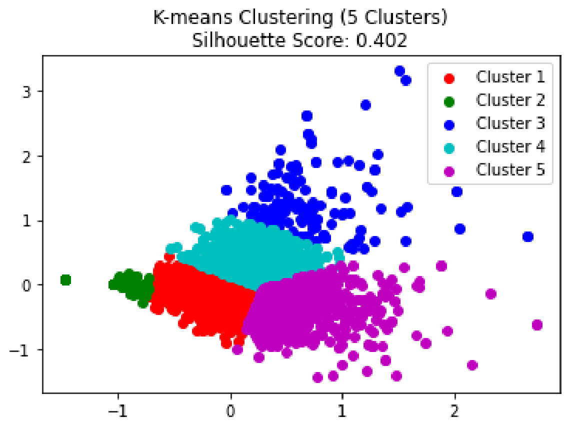
```
model = gensim.models.Word2Vec(preprocessed_data, vector_size=50, window=10, min_count=2, workers=10, sg =1)
model.train(preprocessed_data,total_examples=len(preprocessed_data),epochs=150)
```

```
(87092327, 110807100)
```

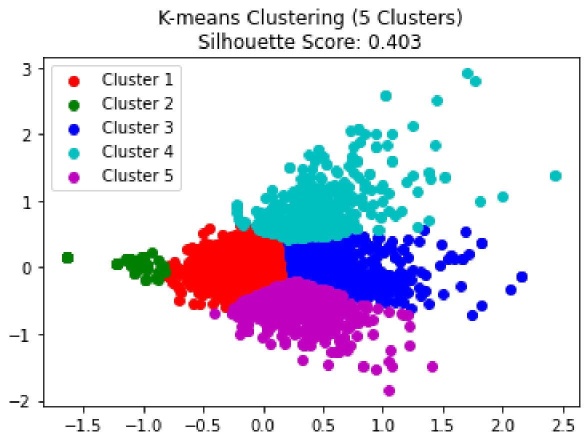
We are calculating cosine similarity using skipgram model and implementing clustering on the model. The clustering we are doing is K-mean clustering to compare performance of it with CBOW model. So, following is the result of K mean clustering.



After that we are changing vector size of the model and now increasing it from 50 to 300 and checking out the difference in result using clustering. As we can see score decrease upon increasing vector size.



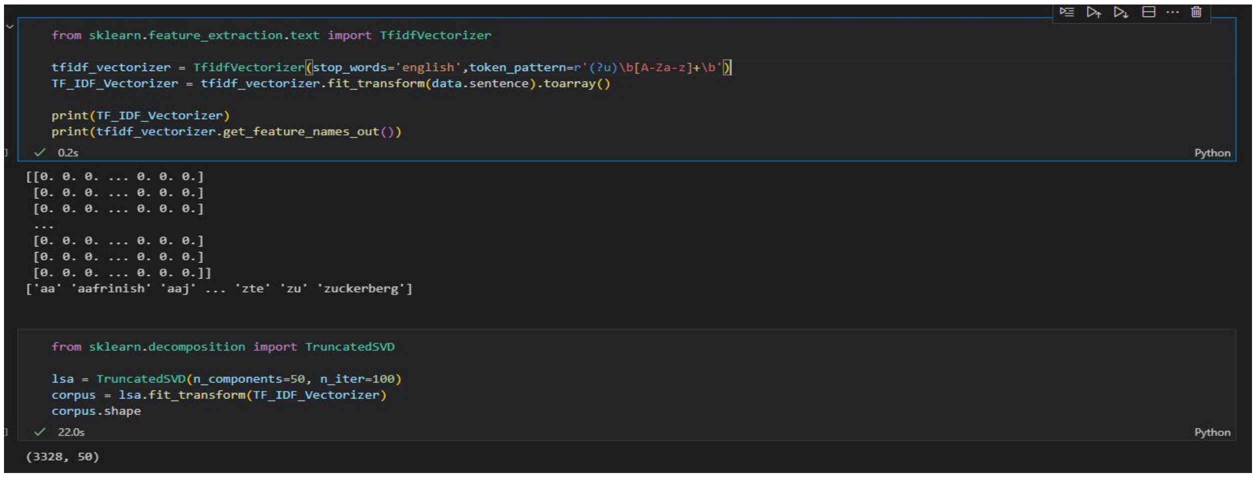
As we can see, the score of the clustering decrease. So now we are changing the window size of the model from 10 to 5 and checking out the impact on clustering but from the result we can see clustering is not much affected by the window size.



Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a technique in natural language processing (NLP) used to identify patterns in text data. It is a mathematical approach that involves creating a matrix of word occurrences in a document corpus and using matrix factorization techniques to reduce the dimensionality of the matrix. This results in a representation of the text data in a lower-dimensional space, where semantically similar words and documents are grouped together.

For LSA model, we are first loading our dataset and creating TF-IDF vectorizer using that data. Once the TF-IDF vector is created then we are applying truncated SVD on it to reduce its dimensionality.



```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(stop_words='english',token_pattern=r'(?u)\b[A-Za-z]+\b')
TF_IDF_Vectorizer = tfidf_vectorizer.fit_transform(data.sentence).toarray()

print(TF_IDF_Vectorizer)
print(tfidf_vectorizer.get_feature_names_out())
```

✓ 0.2s Python

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
['aa' 'aafrinish' 'aaj' ... 'zte' 'zu' 'zuckerberg']
```

```
from sklearn.decomposition import TruncatedSVD

lsa = TruncatedSVD(n_components=50, n_iter=100)
corpus = lsa.fit_transform(TF_IDF_Vectorizer)
corpus.shape
```

✓ 22.0s Python

```
(3328, 50)
```

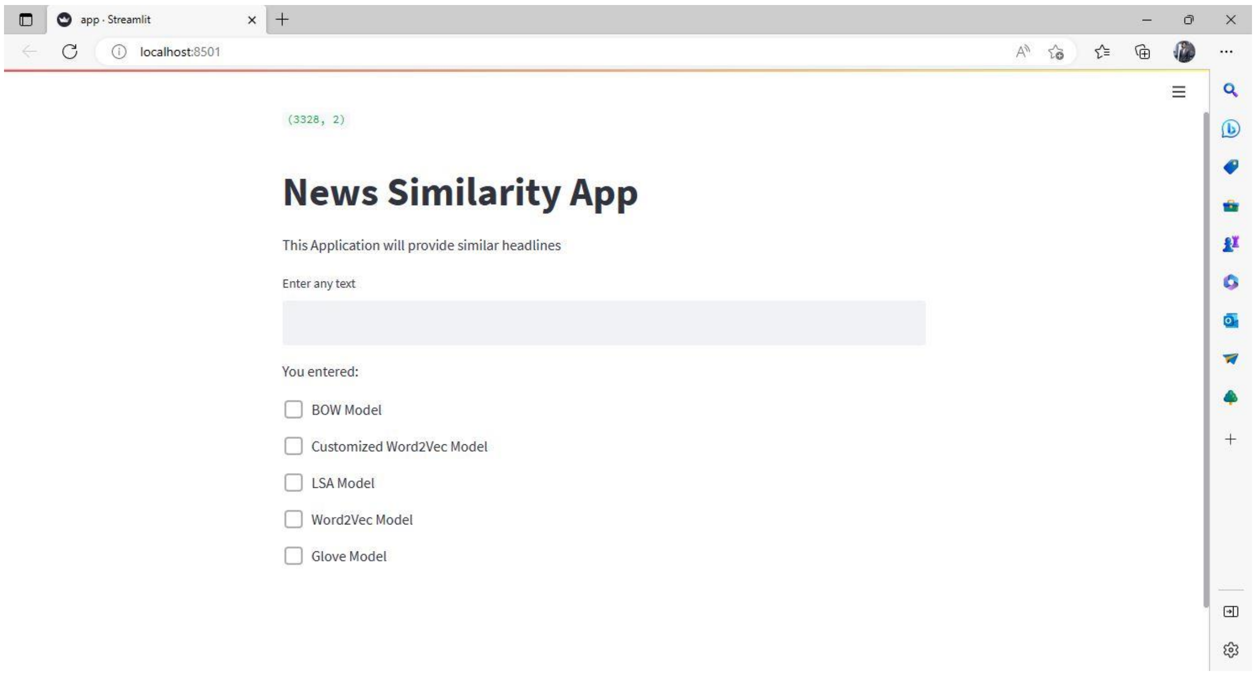
Once the LSA model is created we are then implementing this model on our dataset to create sentence embeddings and calculating the cosine similarity between the data corpus and the embeddings of the input sentence.

Clustering Type	3 Clusters	4 Clusters	5 Clusters
K Mean	0.529	0.169	0.184
Mini Batch K Mean	0.192	0.164	0.151
Agglomerative	0.541	0.547	0.536

From the above results of clustering, we can see that performance of agglomerative clustering is better as compared to the other two types in LSA/SVD matrix.

STREAMLIT INTERFACE

After selecting our best models, we our deploying our sentence embeddings and models on the web app created on stream lit API. The app will take raw text as an input and will return headlines which are similar or related to that text by calculating its cosine similarity. We can check similarity of text using all five models which are deployed on the app. I am attaching some screenshot of the interface of the app and some of its searches. The code for the webapp is available in app.py file.



```
(3328, 2)
```

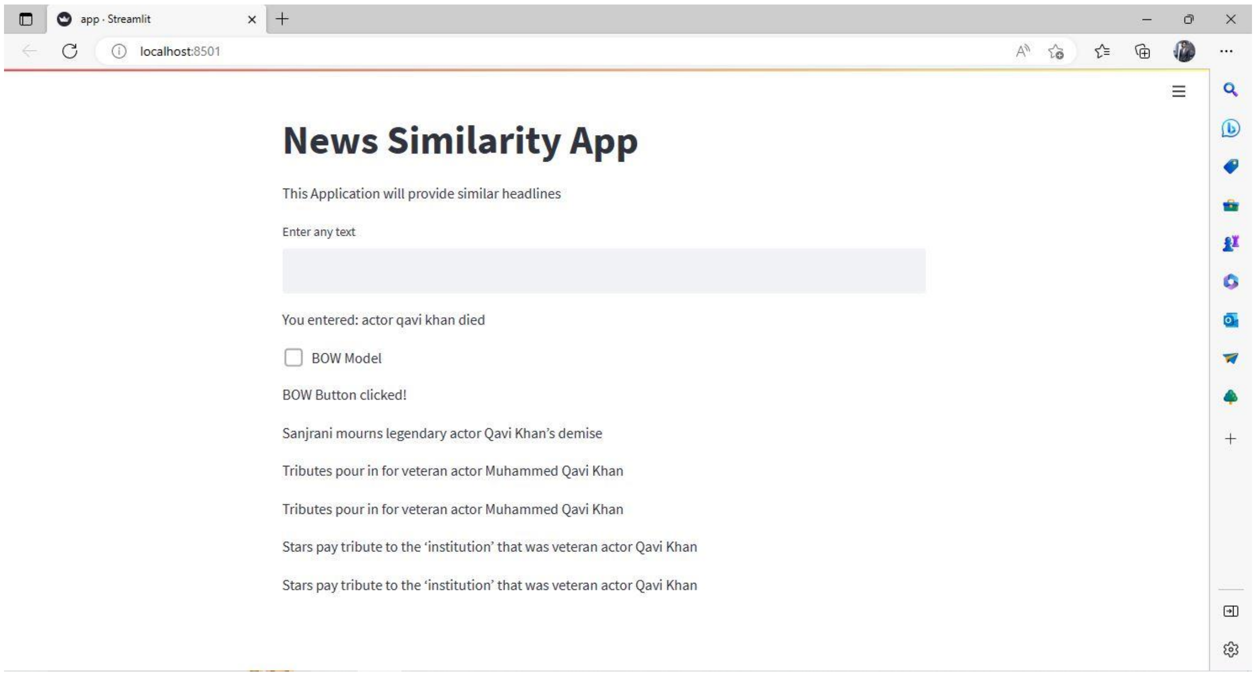
News Similarity App

This Application will provide similar headlines

Enter any text

You entered:

- [ ] BOW Model
- [ ] Customized Word2Vec Model
- [ ] LSA Model
- [ ] Word2Vec Model
- [ ] Glove Model



News Similarity App

This Application will provide similar headlines

Enter any text

You entered: actor qavi khan died

☐ BOW Model

BOW Button clicked!

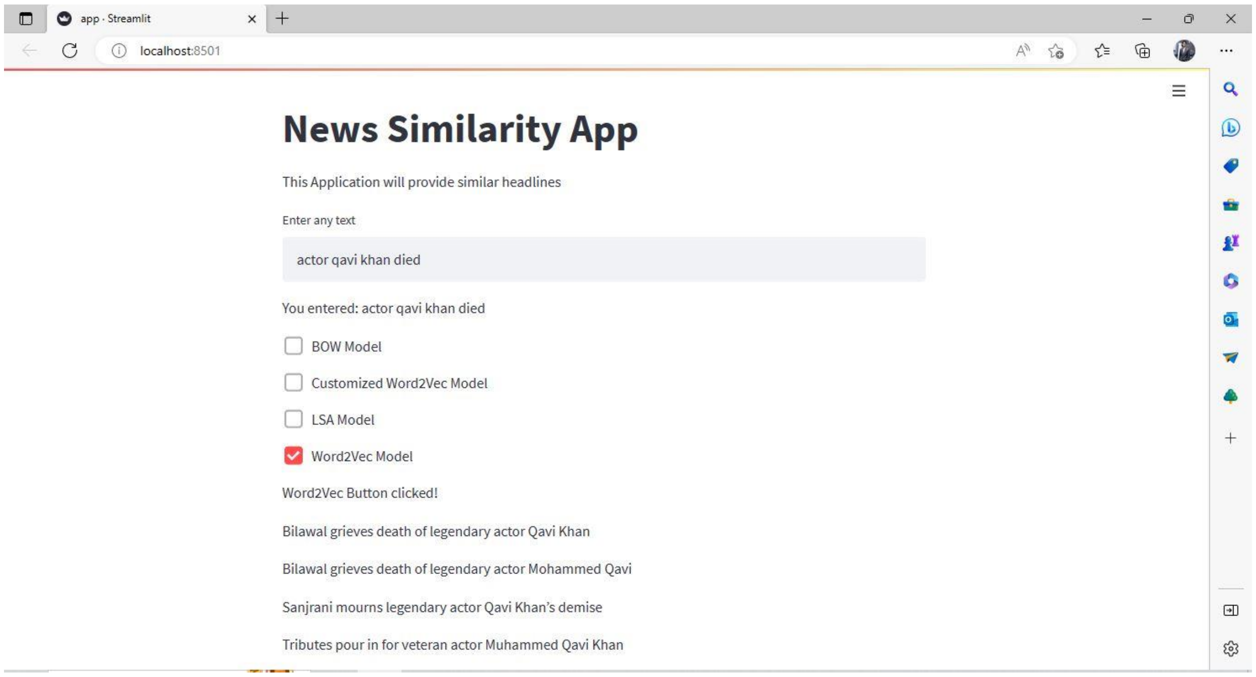
Sanjrani mourns legendary actor Qavi Khan's demise

Tributes pour in for veteran actor Muhammed Qavi Khan

Tributes pour in for veteran actor Muhammed Qavi Khan

Stars pay tribute to the 'institution' that was veteran actor Qavi Khan

Stars pay tribute to the 'institution' that was veteran actor Qavi Khan



News Similarity App

This Application will provide similar headlines

Enter any text

actor qavi khan died

You entered: actor qavi khan died

☐ BOW Model

☐ Customized Word2Vec Model

☐ LSA Model

☑ Word2Vec Model

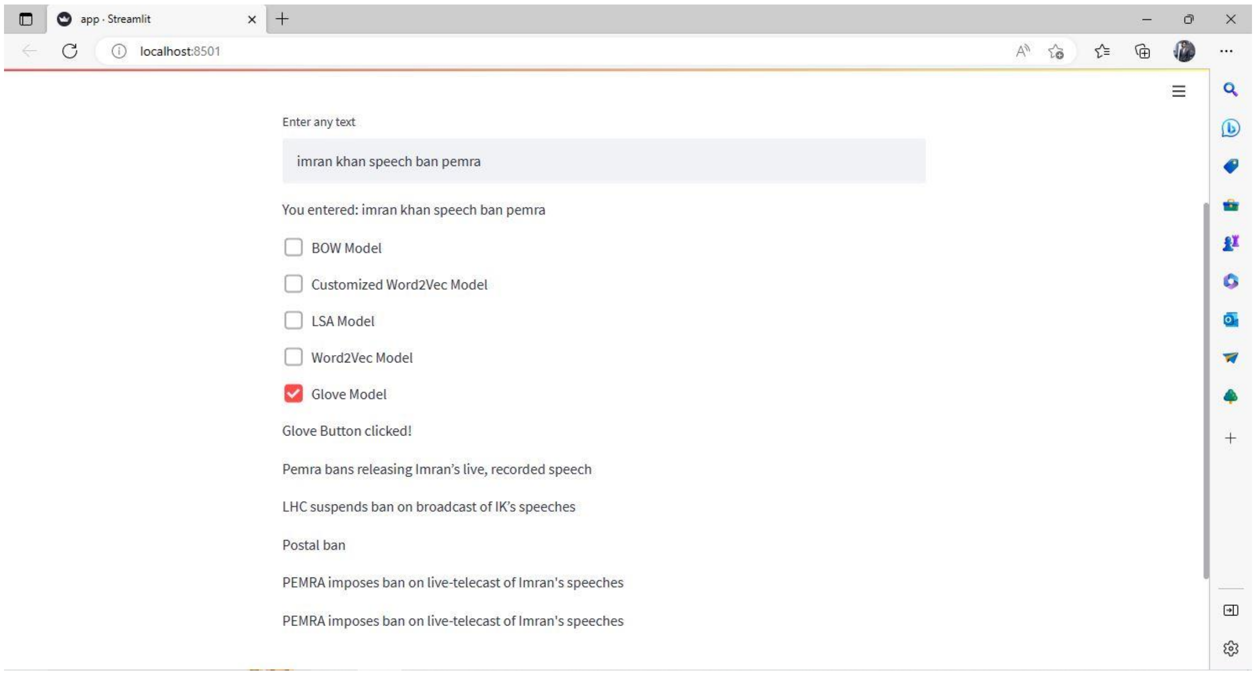
Word2Vec Button clicked!

Bilawal grieves death of legendary actor Qavi Khan

Bilawal grieves death of legendary actor Mohammed Qavi

Sanjrani mourns legendary actor Qavi Khan's demise

Tributes pour in for veteran actor Muhammed Qavi Khan



Enter any text

imran khan speech ban pemra

You entered: imran khan speech ban pemra

☐ BOW Model

☐ Customized Word2Vec Model

☐ LSA Model

☐ Word2Vec Model

☑ Glove Model

Glove Button clicked!

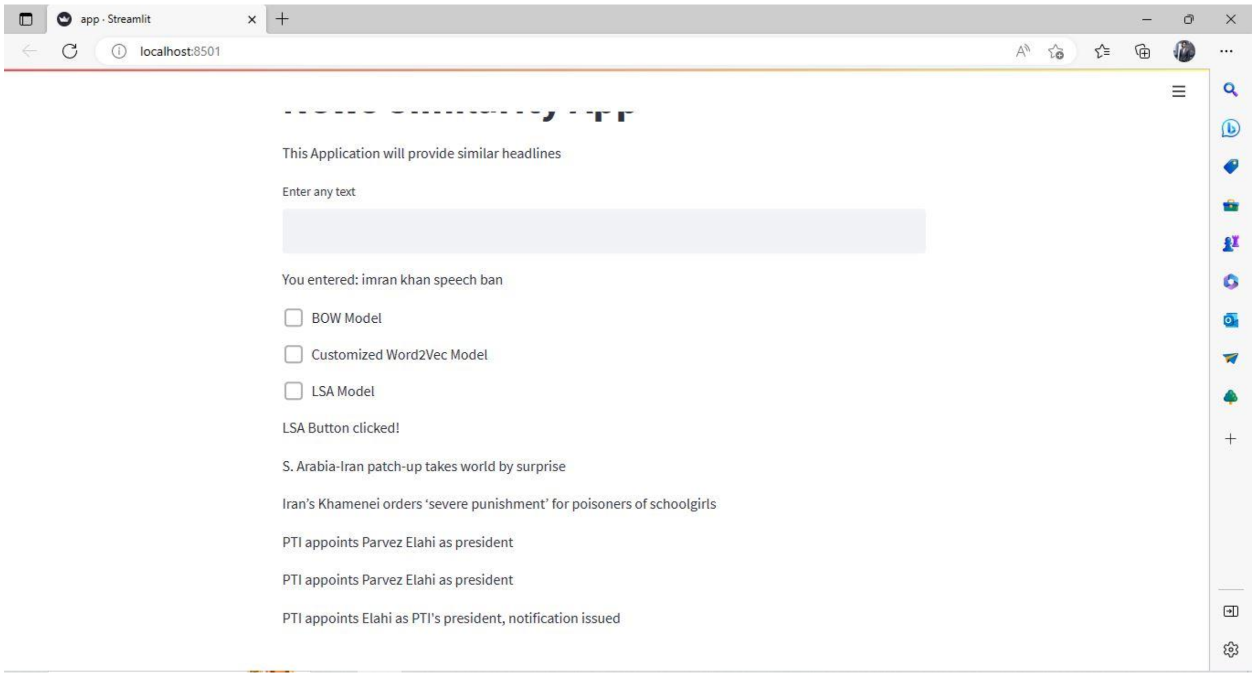
Pemra bans releasing Imran's live, recorded speech

LHC suspends ban on broadcast of IK's speeches

Postal ban

PEMRA imposes ban on live-telecast of Imran's speeches

PEMRA imposes ban on live-telecast of Imran's speeches



This Application will provide similar headlines

Enter any text

You entered: imran khan speech ban

☐ BOW Model

☐ Customized Word2Vec Model

☐ LSA Model

LSA Button clicked!

S. Arabia-Iran patch-up takes world by surprise

Iran's Khamenei orders 'severe punishment' for poisoners of schoolgirls

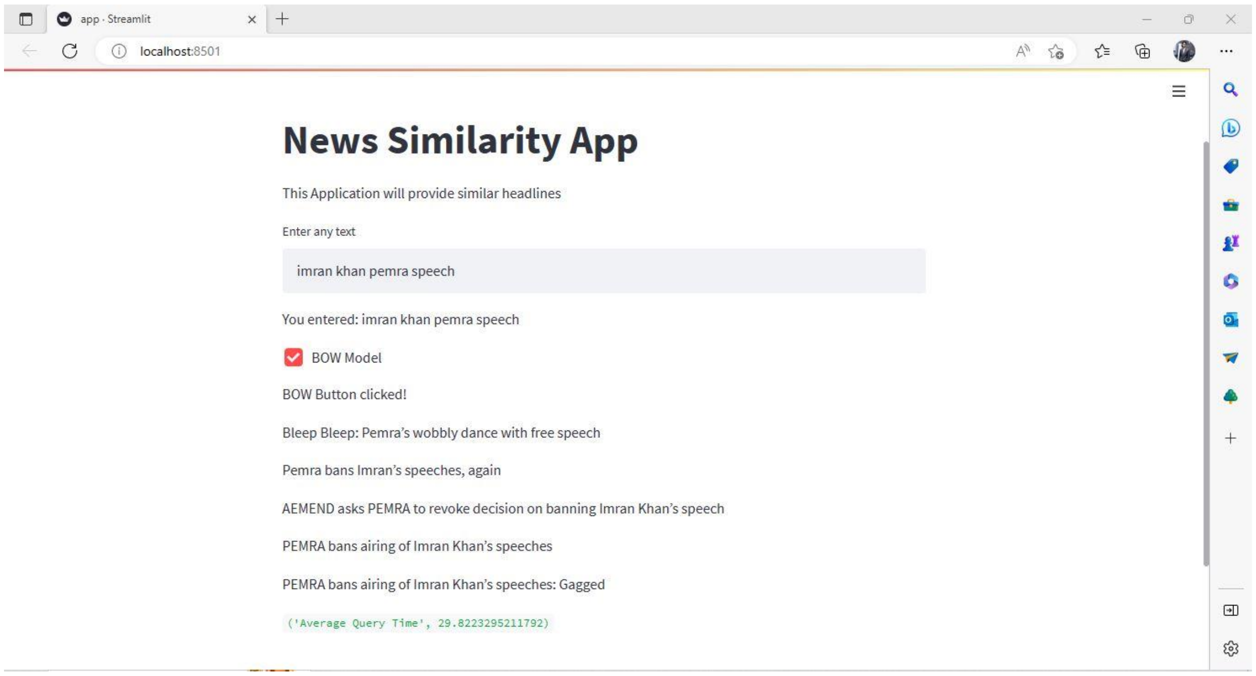
PTI appoints Parvez Elahi as president

PTI appoints Parvez Elahi as president

PTI appoints Elahi as PTI's president, notification issued

TIME PER MODEL

We are also calculating the time each model taking to run from its loading to its execution. For the specific reason we are using time library of python to calculate time.



News Similarity App

This Application will provide similar headlines

Enter any text

imran khan pemra speech

You entered: imran khan pemra speech

✓ BOW Model

BOW Button clicked!

Bleep Bleep: Pemra's wobbly dance with free speech

Pemra bans Imran's speeches, again

AEMEND asks PEMRA to revoke decision on banning Imran Khan's speech

PEMRA bans airing of Imran Khan's speeches

PEMRA bans airing of Imran Khan's speeches: Gagged

```
('Average Query Time', 29.8223295211792)
```

RESULTS

So, after trying different combinations and performing hyperparameter tuning of the models. We have analyzed the following points:

1- Bag of word model is best performing with the stemming and TF-IDF. Using stemming or Binary TF – IDF decrease performance of the model. So stemming is working well with BOW.

2- Using bigram and trigram with TF-IDF is not much affecting the result as we have observed that using unigram are giving us best clustering score. Performance of TF-IDF with bigram and trigram decrease a bit.

3- We are getting better result using default max_df and min_df parameter. Increase the number is not increasing the performance.

4- K-Mean clustering is working well and giving best score as compared to mini batch and agglomerative clustering for both word2vec and Glove.

5- SKIPGRAM model is performing better than CBOW model in customized word2vec model.

6- Changing vector size and window size is not much affecting the performance of SKIPGRAM model.

7- TFIDF model is performing better than LSA model. Clustering score with TF-IDF BOW is better than LSA/SVD.