Multimodal Search Engine

Aabid Hoda
Department of Mathematics, IIT Delhi
Entry no. 2019MT10068

Anirudh Sharma
Department of Mathematics, IIT Delhi
Entry no. 2019MT10075

Ayan Jain
Department of Mathematics, IIT Delhi
Entry no. 2019MT10678

Deepak
Department of Mathematics, IIT Delhi
Entry no. 2019MT10685

Jayesh
Department of Electrical Engineering, IIT Delhi
Entry no. 2021EET2720

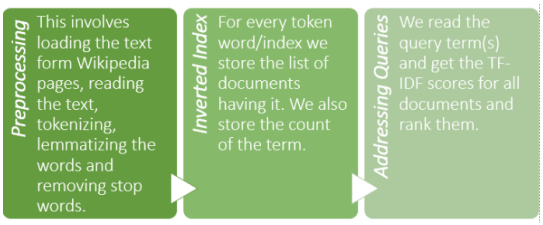
***Abstract*—In this assignment we aim to build a multimodal search engine. We start by implementing a text search engine and then an image search engine. Finally, we will combine both with some modifications that will take Text, Image, Audio as well as Video as an input and return the list of relevant documents and images from the corpus.**

PART A

A. PROBLEM STATEMENT

To take N phrases/words, download the text from the wikipedia pages for those phrases. Implement a search engine which takes as input word and the algorithm the user wants to use (0 for TF-IDF algorithm) and (1 for the Word2Vec algorithm). The articles must be ranked from 1-N, where the rank represents closeness of the article from the searched term. Along with the ranks, URLs of the respective articles must also be given. Rank of the articles for a search term using both TF-IDF and Word2Vec. Closest search terms to the input word using the selected algorithm.

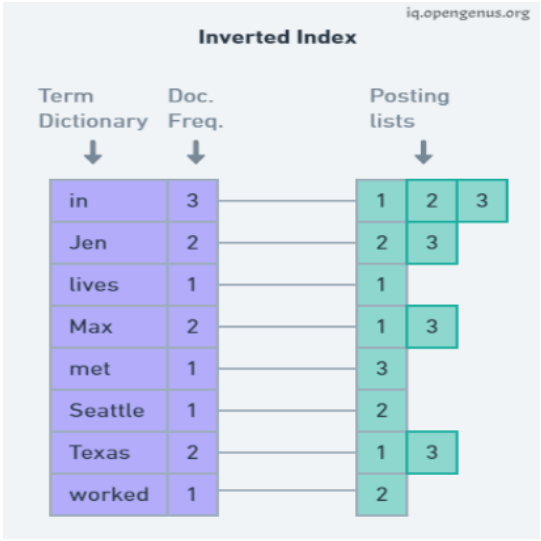
B. STRUCTURE CHART FOR TF-IDF ALGORITHM



Flow chart for TF-IDF Algorithm

C. IMPLEMENTATION OF TF-IDF ALGORITHM

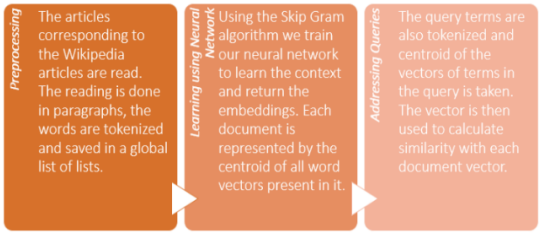
- A list of (N = 370) phrases is taken. We download the text of the corresponding wikipedia pages one by one.
- The text is then tokenized by breaking/splitting to text on a number of delimiters like : ” ”, ”.” , ”;” , ”:” ,””, ”,” , ”[”, ”]”, ”}”, ”{”, ”)”, ”(”. The words which are present in the english stopwards list by NLTK library are then removed. Each word is also stemmed using the Porter Stemmer library from NLTK.
- For each document we store it’s doc-id / frequency of the term in this document for each term that is present in it after preprocessing.
- We also store the frequency of each term in the inverted indes.
- Wherever a query term come, it is also preprocessed in a similar way as the wikipedia text.
- For each query word we give a TF*IDF score to each document.
- The documents are then ranked and presented to the user based upon this score.
- **Note-1** : TF-IDF is not a leaning (ML) algorithm, hence, there is no way to calculate the “similarity” between two terms present in the inverted index storage.
- **Note-2**: As TF-IDF is incapable of calculating the “similarity”, the documents that can get scores are only those containing at-least one query term. In our case, we are given 1 query word, hence, we rank only those documents in which this term is present.



Representational image for inverted index

Identify applicable funding agency here. If none, delete this.

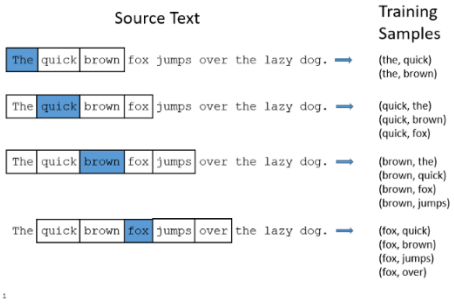
D. STRUCTURE CHART FOR WORD2VEC ALGORITHM



Flow chart for word2vec algorithm

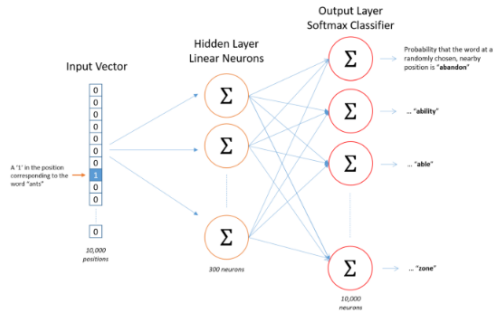
E. IMPLEMENTATION OF THE WORD2VEC ALGORITHM:

- We read the articles for each phrase, reading is done one article at a time. The words are tokenized and then stored in a "temporary list" of words. After each paragraph, we store the "temporary list" in a "Global list of lists".
- After, reading all articles, we learn the context using neural network. We use the Skip- Gram model/algorithm for this. Skip Gram works to predict the context from the target word. This way we get embeddings for the neural network which can be used to compare similarity between any two words present in the corpus.
- For each document we get a vector using the Skip Gram Model's learning. This vector is constructed by taking the mean of vectors of all terms present in the document.
- Whenever the query term(s) is given we preprocess it in exactly the same way and then find the centroid of the query term.



Representational image for Skip Gram Model

- We then calculate the cosine similarity between the centroid document vector and the centroid query vector.
- The documents get scores between 1 to -1, the documents are then ranked upon these scores.
- We use the Skip Gram model to find 10 closest terms in the corpus to the query provided by the user.



Representational image for Neural Network

F. OUTPUT FOR TF-IDF ALGORITHM:

We typed the query "India" for checking the output results by the TF-IDF algorithm. The image for the list of articles given by the algorithm is shown below. As TF-IDF is not a learning based algorithm, there is no way we can give the words close/similar to the input query.

All the articles in the image are related to the term "India". The similarity score is the value of TF*IDF for the documents/articles wrt the given query.

The time taken for the retrieval of documents for "India" was 0.008033 s.

Results for your query India are:

Rank	TF-IDF Score	Article Heading
1	0.012945	Sikh diaspora
2	0.0115492	University of Calcutta
3	0.00872274	Burmese Indians
4	0.00746269	Matrilineal society of Meghalaya
5	0.00740132	Manasollasa
6	0.00616592	Indian Institute of Management Rohtak
7	0.00515811	Sri Lankan Tamils
8	0.00463201	Indian Institute of Management Lucknow
9	0.00286478	Mangalorean Catholics
10	0.00226757	Sexual abuse by yoga gurus

Result for TF-IDF search

G. OUTPUT FOR WORD2VEC ALGORITHM:

The 10 closest term to the given query "India": Again, we typed the query "India" for checking the output results by the Word2Vec algorithm. The image for the list of articles given by the algorithm is shown below.

```
You may also like to search:
-----> ('india', 1.0)
-----> ('lanka', 0.9005614519119263)
-----> ('bengal', 0.8930199146270752)
-----> ('malaysia', 0.8899531364440918)
-----> ('pakistan', 0.8881345987319946)
-----> ('russia', 0.8828991055488586)
-----> ('caribbean', 0.8791177868843079)
-----> ('canada', 0.8783040046691895)
-----> ('europ', 0.8769310712814331)
-----> ('germani', 0.8749318718910217)
```

Closest 10 terms for query word

The time taken for the retrieval of documents for "India" was 0.049517 s.

Top 50 results for your query India are:

Rank	Similarity Score	Article Heading
1	0.804076	Sikh diaspora
2	0.802952	Sri Lankan Tamils
3	0.799511	Canadians
4	0.78784	British people
5	0.782321	Polyethnicity
6	0.776458	Cornish people
7	0.775246	History of the Jews in Puerto Rico
8	0.773701	Sri Lankan Tamil nationalism
9	0.773536	Burmese Indians
10	0.770406	Armenian Americans

Result for Word2Vec search

H. SIMILARITY BETWEEN BOTH THE OUTPUTS

Although, the searched articles in both the algorithm appears to be related to the query term “India”. There is a difference in the ranking of the related documents.

In the TOP-7 results 3 articles are common. In the top-20 there were 13 common articles. In top 25 articles, 15 were common.

I. CONCLUSION

TF-IDF is a statistical method which calculates the relevance of each document by multiplying two terms.

- **Term frequency** It measures how often a word occurs in a document.
- **Inverse Document frequency** It measures how rare a word is or how much importance the presence of the word has.

N closest words can be found only for the words which are present in the corpus. For, TF-IDF algorithm, we construct the inverse document frequency only. In order to get the n closest terms we would have to construct a matrix of size V*V where V is the size of the vocabulary. While, the Word2Vec algorithm easily gives the closest terms by using the embeddings learned from training the neural network. The results of TF-IDF only include documents containing the query term while in Word2Vec all documents can be ranked.

PART B

J. PROBLEM STATEMENT

Take N words/phrases and download the top 50 images from any search engine. Also resize them to max (1000, 1000) pixels and maintain aspect ratio. We don’t need to resize smaller images. Now we need to take input as an image from the user and a value ’k’ and then, Use SIFT (Scale-Invariant Feature Transform) and Bag of Visual Words to build the index to perform the search (using k words = 5, 10, 50, 100, 500). After this we show the images which are closest visually to the input image and also show the tag (the word or phrase which we used to download it). Rank all the images in order of the relevance. Show the statistics like time taken to retrieve, also, how many images from the search word are in top-50 results.

K. DATASET PREPARATION

We built our dataset on Phrases and took the images from the google images using the icrawler. Some of phrases we used are: ”Pluto Mickey Mouse Pet”,”Donald Trump”,”Starbucks Logo”,”Ranveer Singh Moustache”, ”Kingfisher Front”, ”game of thrones” etc After that we did the resizing of images to max (1000, 1000) pixels.

L. BAG OF VISUAL WORDS

In Bag of Visual Words technique, similar to Bag of Word analogy in Document, content of an image can be inferred from the frequency of ”visual words” (portion/feature of an image). Visual Words are Independent Features. This technique can be used to compare the images. We extract the features i.e create bag of visual words, create histogram out of them and then compare it with another histogram (of another visual words of other image). To get the Visual Words, we use SIFT (Scale-Invariant Feature Transform) technique which is based on the general idea that, Feature Descriptors are Points in a High-Dimensional Space. In short, the Algorithm of SIFT follow following steps:

- Finds the locations where it’s likely to get features using scale space peak selection.
- Accurately locates the feature key points using Keypoint Localization
- Assigning the orientation to keypoints.
- Describes the feature as a high dimensional vector.

So we can extract the features from the image using SIFT using above algorithm. Once we have extracted the features from an image, we use the K-Means Clustering. It partitions the data into k clusters. (k is based on user input based on the number of features he wish to consider). Brief about workings of K-Means: Initialization: Choose k arbitrary centroids as cluster representatives Repeat until convergence

- Assign each data point to the closest centroid
- Re-compute the centroids of the clusters based on the assigned data points

We use the K-Means to compute the dictionary of visual words. We build the histogram based on k-means thereby turning every image into the histogram. If we use too less features, we might not cover all parts of image region, and if we use more features, we might over-fit. Thus optimal number features to use and extract are required for better results.

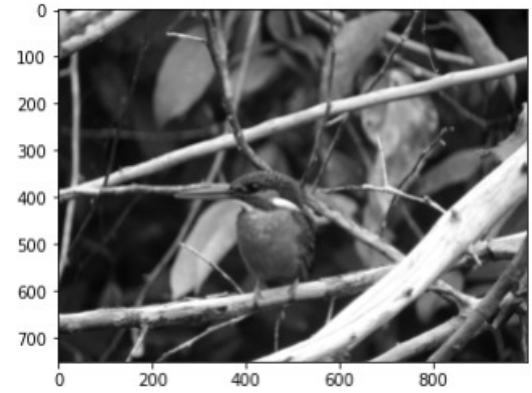
M. IMPLEMENTATION DETAILS

- Building dataset using icrawler getting top-50 images from google search engine based on phrases.
- Resize images to max (1000, 1000) pixels
- We then extract features using cv2.SIFT and detect and compute keypoints using extractor.detectAndCompute.
- Used kmeans fit on the descriptor list.
- Built histogram using k-means increasing frequency of each feature by 1.
- Used nearest neighbor to get the distance (closeness) between the histograms.

- Rank based on closest (least distance) histogram and get Top-50 results.

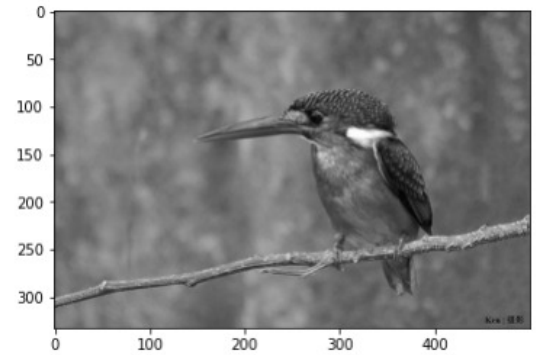
N. OUPUT

Input Image



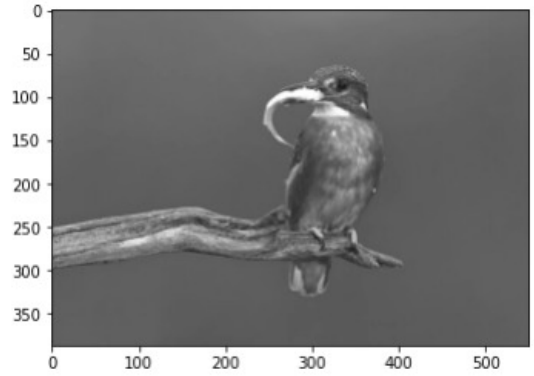
```
Image Tag : Kingfiher Front
Image Similarity Rank : 1
Similarity Score : 100.0 %
```

1st matching image



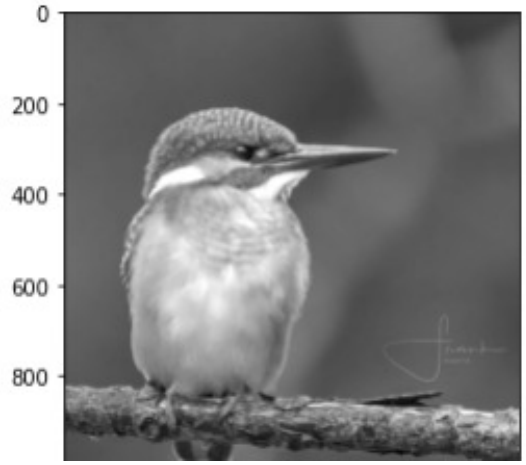
```
Image Tag : Kingfiher Front
Image Similarity Rank : 2
Similarity Score : 24.02530733520421 %
```

2nd matching image

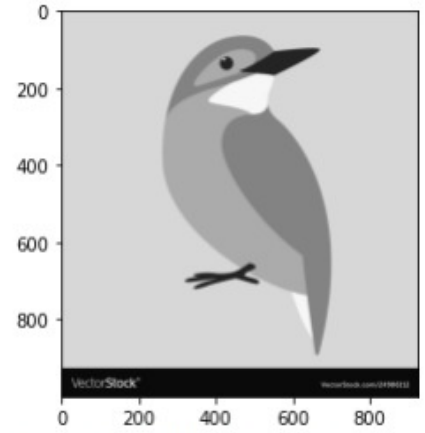


```
Image Tag : Kingfiher Front
Image Similarity Rank : 3
Similarity Score : 24.02530733520421 %
```

3rd matching image

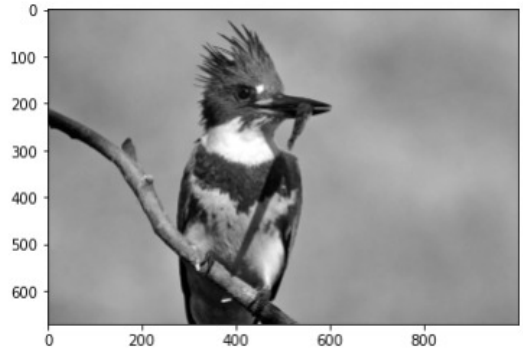


4th matching image



```
Image Tag : Kingfiher Front
Image Similarity Rank : 5
Similarity Score : 21.0896722059534 %
```

5th matching image



6^{th} matching image

O. CONCLUSION

On a dataset of 1000+ images, SIFT and Bag of Visual words techniques took 6 minutes for k = 5 and 10 and 9 minutes for k = 100 to fetch out the Top-50 results from the dataset. Accuracy of prediction and correctness is dependent on the number of features to use (which changes for every input image), thus, problems of under-fitting and overfitting can occur.

PART C

P. PROBLEM STATEMENT

We have to implement a model that is capable of searching using all four modalities i.e. text, image, audio and video.

Given any audio file , our model should search for relevant articles related to the context of the audio file.

Given a video , our model should display related images and related articles similar to input video.

Model should retrieve the most relevant and similar words with respect to the audio or video input.

For all the similar articles our model should also provide ranks and their respective URLs. Ranks depicts how close or similar it is from the input file.

For all images, related to input video, our model should provide a similarity score which depicts how closely that image is linked to the input video.

Q. IMPLEMENTATION OF SEARCH BY AUDIO ENGINE

Whenever a query of audio search comes, we implemented following :

- Extracting text from audio file. For implementing the same , we used the speech recognition library which is based on the hidden Markov model.
- We extracted the text and stored it using the above mentioned library .
- Later we used the Word2Vec model developed in part A, to search for relevant articles related to the retrieved text from the audio.
- In Word2Vec implementation, we firstly used tokenization, stemming and lemmatization to extract important words from the text.
- Now, the Word2Vec model searches for the most similar articles related to the audio text .
- Lastly, we used the learned Skip Gram model for finding and displaying the 10 most similar articles sorted according to their rank(based on similarity) with their URLs.

R. OUTPUT OF SEARCH BY AUDIO

We used an audio file for testing the speech search engine, the speech _ recognition library converted the given audio to the following text: “USA has many universities.” The same text was passed to the text _ search engine/algorithm (Word2Vec algorithm). The following result was obtained:

Rank	Similarity Score	Article Heading
1	0.920945	California State Polytechnic University, Pomona
2	0.915521	Texas State University
3	0.915291	Florida International University
4	0.915022	Nova Southeastern University
5	0.914302	University of Miami
6	0.913838	University of Houston
7	0.912866	University of North Dakota
8	0.912554	University of Surrey
9	0.911353	University of Calcutta
10	0.911033	University of Central Florida

Wikipedia pages for search by audio

S. IMPLEMENTATION OF SEARCH BY VIDEO ENGINE

Video is a combination of audio and frames of images displayed at a certain frequency. We used the same fact to develop a video search engine . Given a video, we use its audio and frames both . We looked at the audio and images separately and then merged the results to find similar articles and images related to input video.

Whenever a query of video search comes, we implemented following :

- Extract text from the video file. For implementing the same ,we used the MoviePy library.
- Using the above library , we first convert the video file to an audio file .
- Now, we use the above implementation of audio search to search for related articles.
- We display the 10 most similar articles sorted according to their rank (based on similarity) with their URLs.
- For image search , we first take snapshots of frames at a particular fixed interval. For implementing this, we use the CV2 library . We then store these frames.
- Now, we search for similar images related to these frames using the Image Search Engine developed in Part B of this assignment.
- At last, we display similar images with their similarity scores.

T. OUTPUT OF SEARCH BY VIDEO

We used a video file for testing the video search engine . Firstly we extracted the audio from the input video files using the MoviePy library. The extracted text is as follows : “editions of the current flag which was used for the 1951 the additional colors but then added as representation of Sri Lanksan Tamil

and Muslims" . Now, we use the Text Search Engine developed in Part A . The following results were obtained:

Rank	Similarity Score	Article Heading
1	0.973273	Flag of Sri Lanka
2	0.965182	Sri Lankan Tamils
3	0.957604	Islam in Sri Lanka
4	0.954301	Polyethnicity
5	0.95398	Coast Veddas
6	0.95085	Cornish people
7	0.949115	Greeks
8	0.946843	Icelanders
9	0.946791	Chinese Indonesians
10	0.946679	Canadians

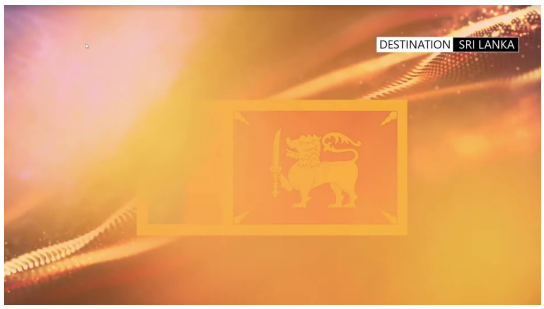
Wikipedia pages for search by video

For the image retrieval we took snapshots at equal intervals and got the following snapshots:

1st snapshot image

2nd snapshot image

3rd snapshot image



4th snapshot image

The following image matches were obtained:

1st matched image

2nd matched image

3rd matched image

4th matched image

5th matched image

6th matched image

U. CONCLUSION

From the above results we concluded that search by video is very effective and efficient . It captures images and audio both, thus making it very effective.With our limited corpus size of 1000 images and 370 articles, the accuracy of the search by video was very good. Increase in size of corpus will increase the accuracy of this search.