NEWS ARTICLE SUMMARIZATION AND CLASSIFICATION

PROJECT REPORT

Submitted for CAL in B.Tech –Natural Language Processing(CSE4022)

By

Aakash Tiwari 16BCE1091

Dhruv Garg 16BCE1190

Slot: A2+TA2

Name of faculty: **Prof. BHARADWAJA KUMAR**

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

April, 2019

ABSTRACT

Every day, people rely on a wide variety of sources to stay informed -- from news stories to social media posts to search results. Being able to develop Machine Learning models that can automatically deliver accurate summaries of longer text can be useful for digesting such large amounts of information in a compressed form. Automatic systems based on extractive summarization techniques select the most significant sentences of one or more texts to generate a summary. The amount of information available electronically is growing. So there is increasing demand for automatic method for text summarization. Text summarization is process of reducing the size of text document while preserving its information content. Providing concise article to the new generation where people don't have the time to read about news article these summarized content can provide information as well as knowledge and keep the younger generation aware.

1. Introduction:

1.1 Objective and goal of the project

In today's world Big data is widely used for publishing news on the internet or website. To thoroughly understand an event, we have to read massive reports and keep clues in mind, which is very difficult and usually results is a one sided interpretation. To see an occasion, individuals need to seek different reports, coax out pieces of information, along these lines to make a honest judgment dependent on their learning and feelings. While, as an expansive number of news reports keeps on being opened on the Web, particularly for mainstream occasions, the assignment of seeing all parts of the occasion turns into a troublesome. In this way, we fall back on automatic summarization system, which can gather gigantic reports appropriated in the entire Web, and give a brief outline of what occurred. Keeping people aware of all the news all around the world and mostly the younger generation who lead a digital life is the main objective and the goal of the project. Creating awareness in today’s younger generation is very important because they are going to be the future for the nation.

2. Methodology: Experimental/Simulation

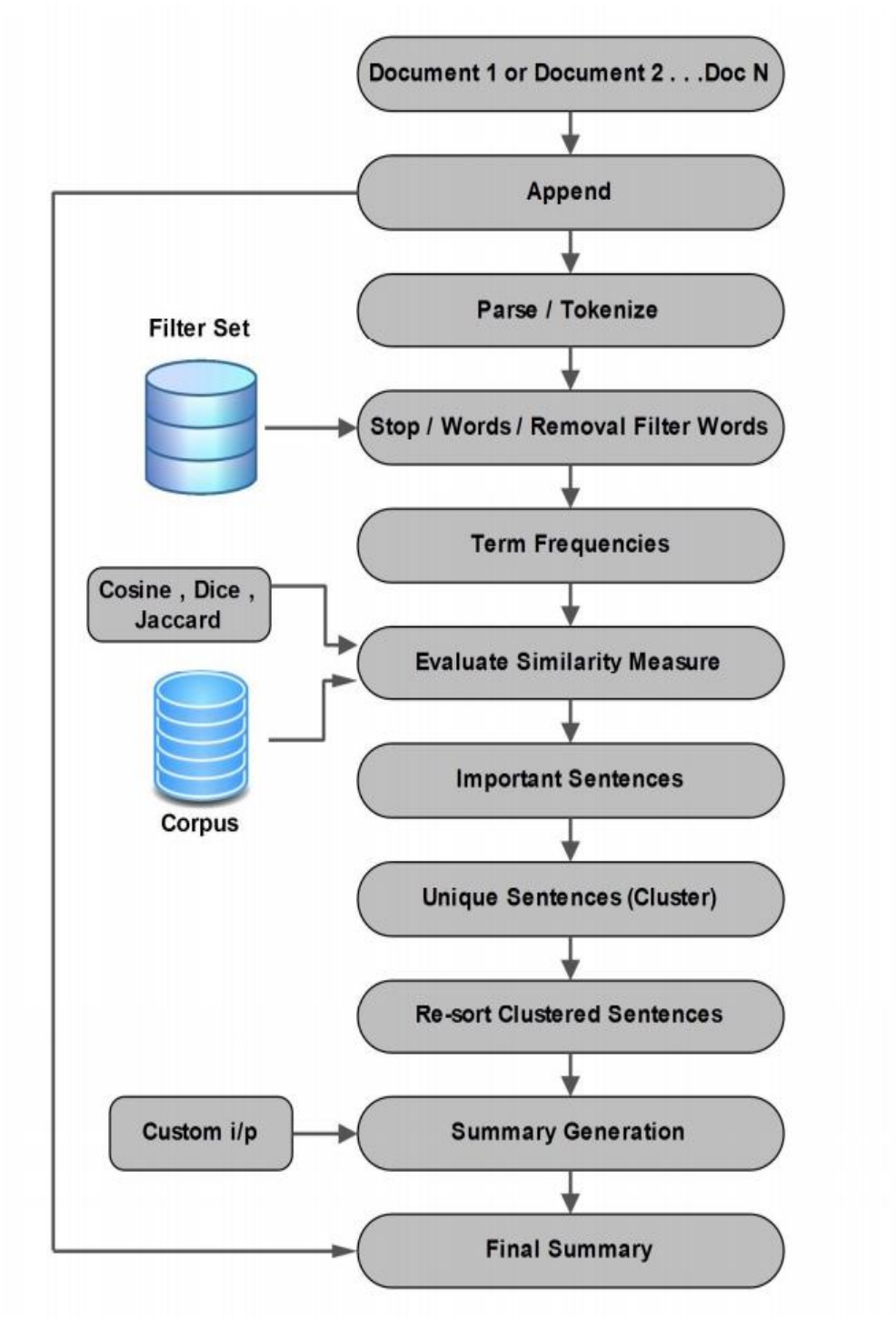


Fig. Proposed methodology for news article summarization.

The process of building the project is as follows –

Data Cleaning:

- **Selection of news article from the internet**. In the first step news articles which are required to be summarized are given by the user. We have taken the dataset and cleaned the dataset by removing the contractions words (like I'm, we'll), apostrophe, exclamation other symbols, numbers, question marks etc.
- **Append and Tokenization -** Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. News articles are appended and then the file content is tokenized into individual word.
- **Lemmatization and Removal of Stop Words** –After word is divided into smaller tokens we find out the root of a word. Lemmatization helps us to achieve the root forms (sometimes called synonyms in search context) of inflected (derived) words. A lemmatizer, a tool from Natural Language Processing which does full morphological analysis to accurately identify the lemma for each word. Doing full morphological analysis produces at most very modest benefits for retrieval. There are words which give us little or no context related information and such words are also removed from the list

Classification:

For classification we used random forest algorithm. It is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees. It is ensemble algorithm. *Ensembled algorithms* are those which combines more than one algorithms of same or different kind for classifying objects. Random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size.

Given the training dataset with targets and features, the decision tree algorithm will come up with some set of rules. The same set rules can be used to perform the prediction on the test dataset.

It works in four steps:

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

We have taken test size to be 20% and test size to be 80% with the number of estimators=300 and depth of tree to be 150 for the better classification of dataset. Once the model is trained we are storing the model in the pickle file for using the trained model on the other dataset with help of library function job.lib.dump(joblib.dump(value, filename, compress=0, protocol=None, cache_size=None)) which returns The list of file names in which the data is stored. If compress is false, each array is stored in a different file.

Summary: Sentence Generation

- **Generation of List of Frequent Words(BOW) -** After eliminating stop words the term-frequent data and inverse document frequency is calculated from text documents and frequent terms are selected which are used to generate text document summary for which we used the concept of Bag of Words. It refers to the representation of text which describes the presence of words within the text data. The intuition behind this is that two similar text fields will contain similar kind of words, and will therefore have a similar bag of words. Further, that from the text alone we can learn something about the meaning of the document.

- **Cosine Similarity:** Similarity measure is evaluated using cosine algorithm and important sentences are generated. Cosine similarity functions in way that it, generates metric that can be used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the

Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity. The cosine similarity helps overcome this fundamental flaw in the 'count-the-common-words' or Euclidean distance approach. Unique sentences are clustered and re-sort and finally, the summary is generated.

- **Word Embedding:** The other approach we used is Word embedding's and the word mover distance. It is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Using the Glove library, we are taking vectors for a word for which there is list of vectors of length 100, consisting of words which are common in syntax and semantic

- **Word Mover Distance**: It uses word embedding to calculate the distance so that it can calculate even though there is no common word. The assumption is that similar words should have similar vectors. Retrieve vectors from any pre-trained word embedding models. It can be Glove, word2vec, fast text or custom vectors. After that it using normalized bag-of-words (nBOW) to represent the weight or importance. It assumes that higher frequency implies that it is more important.

The tokens are given to the word present in the sentence. These sentences are then ranked on the basis of the tokens given, after which the sentences are taken in account for the proper summary. We have also compared the Tf-Idf and Word Embedding's summarization on the basis of features.

Update Details in Database - When the summary is generated then its details is stored in the database and is available to the user for information analysis.

Setup Web Service - A web service to provide summary of given text documents, will be set up. The Web Service client will send request message consisting of document then the server sends the summary as the response message.

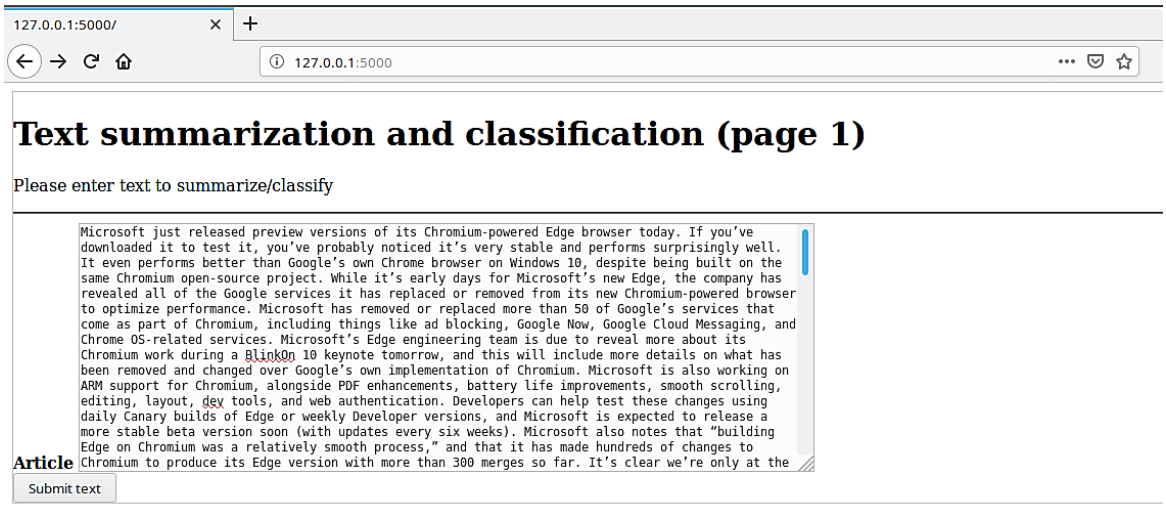
Finally, the summarized articles are deployed on a webpage where each individual user preferences will have noted over the time and a user will receive only those news article suggestions which he/she likes to read the most. This is an recommendation system also. This innovation in the project will keep the users interest in the news and current affairs which is what my ultimate aim is.

Execution steps:

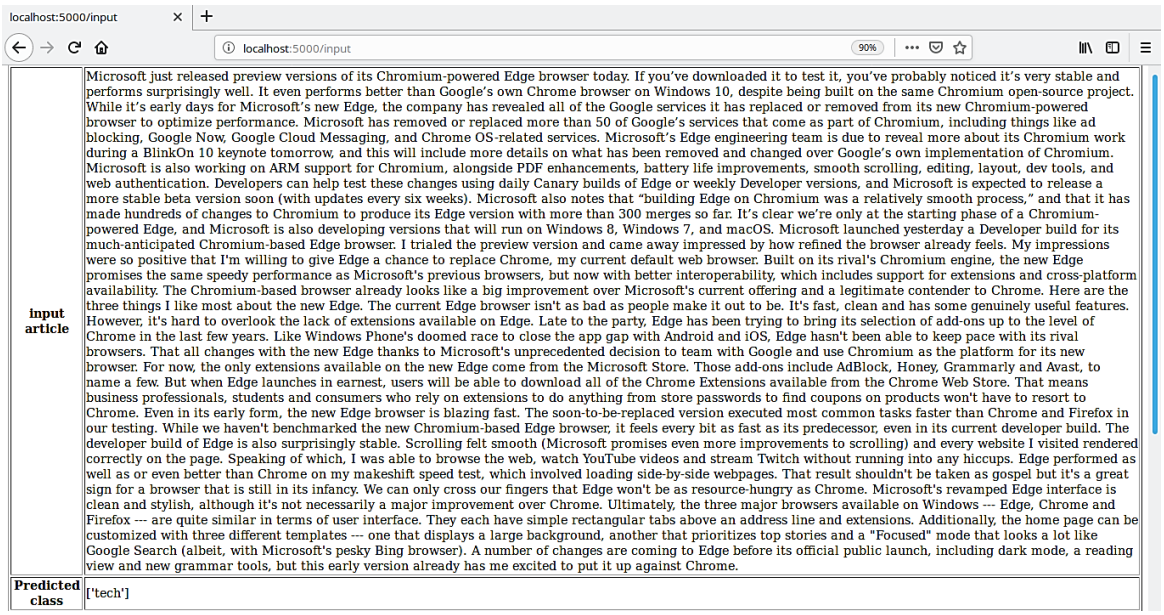
Step 1: Export the python file to FLASK_APP and use flask run to execute.

```
dhruvsgarg@dhruvsgarg:~/NLPProject/BBC-Dataset-News/model$ export FLASK_APP=model2.py
dhruvsgarg@dhruvsgarg:~/NLPProject/BBC-Dataset-News/model$ flask run
 * Serving Flask app "model2.py"
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
```

Step 2: After the models are loaded, the Flask web app can be accessed from the browser. We input the text to summarize/classify.



Step 3: The HTML form when submitted invokes the Python script and the classification of the article is done. Here, the Random Forest Classifier model correctly classifies the News article as one of “Technology news”



input article	Microsoft just released preview versions of its Chromium-powered Edge browser today. If you've downloaded it to test it, you've probably noticed it's very stable and performs surprisingly well. It even performs better than Google's own Chrome browser on Windows 10, despite being built on the same Chromium open-source project. While it's early days for Microsoft's new Edge, the company has revealed all of the Google services it has replaced or removed from its new Chromium-powered browser to optimize performance. Microsoft has removed or replaced more than 50 of Google's services that come as part of Chromium, including things like ad blocking, Google Now, Google Cloud Messaging, and Chrome OS-related services. Microsoft's Edge engineering team is due to reveal more about its Chromium work during a BlinkOn 10 keynote tomorrow, and this will include more details on what has been removed and changed over Google's own implementation of Chromium. Microsoft is also working on ARM support for Chromium, alongside PDF enhancements, battery life improvements, smooth scrolling, editing, layout, dev tools, and web authentication. Developers can help test these changes using daily Canary builds of Edge or weekly Developer versions, and Microsoft is expected to release a more stable beta version soon (with updates every six weeks). Microsoft also notes that "building Edge on Chromium was a relatively smooth process," and that it has made hundreds of changes to Chromium to produce its Edge version with more than 300 merges so far. It's clear we're only at the starting phase of a Chromium-powered Edge, and Microsoft is also developing versions that will run on Windows 8, Windows 7, and macOS. Microsoft launched yesterday a Developer build for its much-anticipated Chromium-based Edge browser. I trialed the preview version and came away impressed by how refined the browser already feels. My impressions were so positive that I'm willing to give Edge a chance to replace Chrome, my current default web browser. Built on its rival's Chromium engine, the new Edge promises the same speedy performance as Microsoft's previous browsers, but now with better interoperability, which includes support for extensions and cross-platform availability. The Chromium-based browser already looks like a big improvement over Microsoft's current offering and a legitimate contender to Chrome. Here are the three things I like most about the new Edge. The current Edge browser isn't as bad as people make it out to be. It's fast, clean and has some genuinely useful features. However, it's hard to overlook the lack of extensions available on Edge. Late to the party, Edge has been trying to bring its selection of add-ons up to the level of Chrome in the last few years. Like Windows Phone's doomed race to close the app gap with Android and iOS, Edge hasn't been able to keep pace with its rival browsers. That all changes with the new Edge thanks to Microsoft's unprecedented decision to team with Google and use Chromium as the platform for its new browser. For now, the only extensions available on the new Edge come from the Microsoft Store. Those add-ons include AdBlock, Honey, Grammarly and Avast, to name a few. But when Edge launches in earnest, users will be able to download all of the Chrome Extensions available from the Chrome Web Store. That means business professionals, students and consumers who rely on extensions to do anything from store passwords to find coupons on products won't have to resort to Chrome. Even in its early form, the new Edge browser is blazing fast. The soon-to-be-replaced version executed most common tasks faster than Chrome and Firefox in our testing. While we haven't benchmarked the new Chromium-based Edge browser, it feels every bit as fast as its predecessor, even in its current developer build. The developer build of Edge is also surprisingly stable. Scrolling felt smooth (Microsoft promises even more improvements to scrolling) and every website I visited rendered correctly on the page. Speaking of which, I was able to browse the web, watch YouTube videos and stream Twitch without running into any hiccups. Edge performed as well as or even better than Chrome on my makeshift speed test, which involved loading side-by-side webpages. That result shouldn't be taken as gospel but it's a great sign for a browser that is still in its infancy. We can only cross our fingers that Edge won't be as resource-hungry as Chrome. Microsoft's revamped Edge interface is clean and stylish, although it's not necessarily a major improvement over Chrome. Ultimately, the three major browsers available on Windows --- Edge, Chrome and Firefox --- are quite similar in terms of user interface. They each have simple rectangular tabs above an address line and extensions. Additionally, the home page can be customized with three different templates --- one that displays a large background, another that prioritizes top stories and a "Focused" mode that looks a lot like Google Search (albeit, with Microsoft's pesky Bing browser). A number of changes are coming to Edge before its official public launch, including dark mode, a reading view and new grammar tools, but this early version already has me excited to put it up against Chrome.
Predicted class	['tech']

Step 4: The summary of the input article is generated through 3 different methods as shown below.

Article summary [Glove]	That all changes with the new Edge thanks to Microsoft's unprecedented decision to team with Google and use Chromium as the platform for its new browser. Microsoft has removed or replaced more than 50 of Google's services that come as part of Chromium, including things like ad blocking, Google Now, Google Cloud Messaging, and Chrome OS-related services. For now, the only extensions available on the new Edge come from the Microsoft Store. Developers can help test these changes using daily Canary builds of Edge or weekly Developer versions, and Microsoft is expected to release a more stable beta version soon (with updates every six weeks). The Chromium-based browser already looks like a big improvement over Microsoft's current offering and a legitimate contender to Chrome
Article summary [BOW]	For now, the only extensions available on the new Edge come from the Microsoft Store. But when Edge launches in earnest, users will be able to download all of the Chrome Extensions available from the Chrome Web Store. Here are the three things I like most about the new Edge. That all changes with the new Edge thanks to Microsoft's unprecedented decision to team with Google and use Chromium as the platform for its new browser
Article summary [WMD]	Here are the three things I like most about the new Edge. It's fast, clean and has some genuinely useful features. Those add-ons include AdBlock, Honey, Grammarly and Avast, to name a few. We can only cross our fingers that Edge won't be as resource-hungry as Chrome. The current Edge browser isn't as bad as people make it out to be. Speaking of which, I was able to browse the web, watch YouTube videos and stream Twitch without running into any hiccups. However, it's hard to overlook the lack of extensions available on Edge. I trialed the preview version and came away impressed by how refined the browser already feels. That means business professionals, students and consumers who rely on extensions to do anything from store passwords to find coupons on products won't have to resort to Chrome. The developer build of Edge is also surprisingly stable

Fig. The above images show the entire workflow from news article submission to predicted class and extractive summary generation.

3. Results and Discussion

An interactive Web App (Flask) is developed, the classification accuracy for various types of news articles using Random forest classifier was found to be 95 %.
This classification model generalized well and was able to correctly predict the category of all the news articles given to it.We used two approaches for feature extraction, Bag of word based TF-IDF vectorizer word embedding based Glove. Text summarization was successfully implemented using the two different features and comparison could be made. It was found that Word embedding feature produce a much better summary, when compared to bag of Word features. The Word embedding based summarization was able to use the context and semantic information to produce richer summarize text.

4. Conclusion

My innovation in the project to build was to provide targeted news article to individual user according to their reading habits and interests. This will always keep the user engaged to the app and would always want to come back and read more news articles that he/she likes according to their preferences. Various methods of extractive approach have emerged in the past. But it is hard to say how much greater interpretive sophistication, at sentence or text level contributes to performance. Without the use of Natural Language Processing, the generated summaries may not be much accurate in terms of semantics. If the input documents cover multiple topics, it becomes difficult to generate a balanced summary.

5. REFERENCES

a. **News Summarization using Text Mining**
https://www.irjet.net/archives/V5/i11/IRJET-V5I1138.pdf

b. **Word Distance between Word Embeddings – Towards Data Science**
https://medium.com/intel-student-ambassadors/diving-into-abstractive-text-summarization-part-1-e8570d370021
https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

c. **Algorithms - Create similarity matrix - Data Science Stack Exchange**
https://datascience.stackexchange.com/questions/23059/create-similarity-matrix

d. **Why Google PageRank still matters in 2018**
https://www.link-assistant.com/news/page-rank-2018.html

e. **joblib.dump — joblib 0.13.2 documentation**
https://joblib.readthedocs.io/en/latest/generated/joblib.dump.html

f. https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1

*****