TEXT Summarization

**Extractive model using news data**

Bayard Rucker  
 School of Computing and Engineering (SCE)  
 UMKC  
 Kansas City MO, USA  
[bbrvpb@mail.umkc.edu](mailto:bbrvpb@mail.umkc.edu)

Claire Ndofor  
School of Computing and Engineering (SCE)  
 UMKC  
 Kansas City MO, USA  
[cbnk6w@mail.umkc.edu](mailto:cbnk6w@mail.umkc.edu)

Vyoma Desai,  
School of Computing and Engineering (SCE)  
 UMKC  
 Kansas City MO, USA  
 [vhdmmh@umsystem.edu](mailto:vhdmmh@umkc.edu)

Charles Scola  
School of Computing and Engineering (SCE)  
 UMKC  
 Kansas City MO, USA  
[cvsqnf@mail.umkc.edu](mailto:cvsqnf@mail.umkc.edu)

**ABSTRACT**

For this group project our team created a text summarization model that would be able to translate articles and convert the translated summary to different languages. The simplest concept of this is a model which translates then reads a text and then produces a concise summary of it. Text summarization falls into two categories, 1) abstractive, and 2) extractive. Both approaches have advantages and disadvantages and we spent a good deal of time thinking through which approach to use. We settled on an extractive model because it is computationally less intensive and does not require a training dataset, which is perfect for converting the summary to different languages. The extractive model also uses the original text which the team thought would be more valuable to our target end-users, either subject matter analysts or graduate students. The dataset we use is a combination of web articles from news sources (specifically CNN) and any input the user uses. One of the main reasons for choosing this dataset was that the news articles would be fairly easy to summarize enabling both the model itself and our team's ability to manually check the model’s performance. We also liked news articles because they cover a broad array of topics so the model would need to be fairly robust.

**CCS CONCEPTS**

• Computing methodologies

• Mathematics of Computing

• NLP

• OOP language i.e python

* Cloud computing for building and running the model

**KEYWORDS**

•  Abstractive

* Extractive

1**Introduction**

The starting point of this project was asking the question, “How can people with finite energy and time keep up with the enormous amount of information that is being produced in today's media and academic environments?” While this initial question was very broad, over time we narrowed our scope to only text data specifically related to general news. This was for two reasons: first, news stories cover basically every topic at some point or another so any summarization model that works on the news would need to be very robust. The second reason is that news data fits nicely into our original concern that there is simply being too much data for a person to efficiently consume since there is a constant stream of news. It would be helpful to the end-user if they could stay well informed on issues while saving time by just reading the summarized information. Also, since news articles from other countries might generate useful daya for people, we can help people find worldwide articles. The target groups of people we considered as users were researchers and analysts that have to read in depth about a specific set of subjects and graduate students keeping up with their particular disciplines.

After several conversations, the team decided to go with an extractive model. One of our reasons for choosing this model was that it would be less resource-intensive. Also given the nature of our input data and the people we set out to build this model for in our original question the team felt that a summary consisting of chunks of unaltered text would be more useful for the students or an analyst, and also it would also avoid issues with losing the meaning of the text with translating back and forth.

Once the team had a well understood idea of the end user, the type of model, and the input dataset we outlined this formal hypothesis: If a busy end-user of our extractive text summarization model uses it on a text which they have a general understanding of what the text is about, they will be able to understand the salient points of the text faster than if they had read the entire article themselves.

We have tested this hypothesis against a several articles and found that it works. However, one outstanding question should be addressed as the use of our model permeates the target market is the user’s level of subject matter comprehension based on the summarized material compared to reading the full text.

**2 Analysis of Related Work**

To begin our analysis of related work we start with explaining the general method for text summarization. With all different types of methods for text summarization, they follow this in general outline: As seen in Fig 1, obtained in Tas and Kiyani’s paper [1], the general outline for a basic text summarization algorithm is first you preprocess the text. Preprocessing is basically POS tagging the words and stemming. Next is Pulling the Extraction of Features, which analyzes things like sentence position, term frequency, and sentence length. Next is Calculation of Sentence Score, which can be identified as something similar to finding the cosine similarity values. Finally is Extraction of Sentences, where you pull out the sentences used for the summary. The different methods vary and are considered more complicated, but this ‘outline’ in general is a good starting knowledge for text summarization.

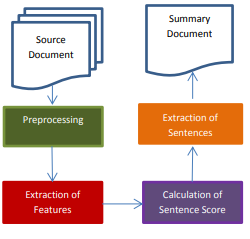


Fig 1: Text Summarization Outline

There are many different methods for text summarization. One method is using lexical chains, which is considered as a chain of cohesion “not only between two terms, but among sequences of related words” [2]. To do this, we find the coherence between the word structure and connect the words together in the chain based on those values. There are clear issues with this method, for instance the results are determined with sentences as whole units, meaning the longer the sentence the more likely it would be selected.

Another technique used for text summarization is called a cluster technique. The clustering technique is when you cluster the documents and sentences through the cosine similarity, putting together sentences that are the most similar and then output the best scoring sentences from the final sentence clusters into the final summary. The results of clustering both the document and sentence seem reliable, as shown in Fig 2 obtained in Deshpande’s study [3]. It increased the precision, recall, and F-Measure compared to statistical features, however it is important to note that document clustering alone is not enough, scoring worse than statistical features.

We have to now discuss the difference between extractive and abstractive summarization. Extractive summary, according to Bhatia and Jaiswal, “extracts the most important part based on statistical and linguistic gestures such as cue words, location, word frequency”. However, it does have issues like sentences are longer, not all relevant sentences are included, the accuracy of the information could be low, and you’ll be missing coherency between sentences [9]. In comparison, Abstractive summarization, according to Paulus et al, “generate new phrases, possibly rephrasing or using words that were not in the original text” [10]. The issue with this method, however, is that there is no clear way to analyze the summary, as the “ROUGE scores” discussed in the article “does not guarantee an increase in quality and readability” [10]. Due to our project revolving around translating text and abstractive methods revolving around creating new words or phrases to summarize the text, we decided to focus our project on Extractive methods in order to minimize any losses due to translation.

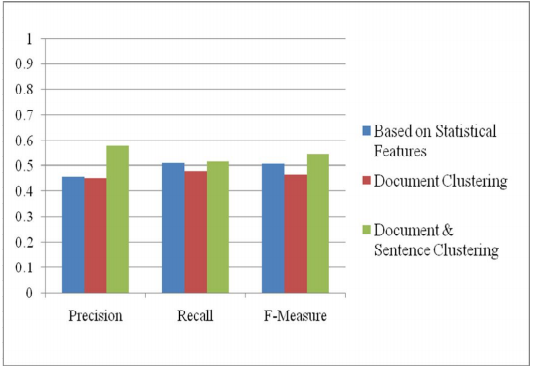


Fig 2: Summarization performed on Document Collection

If we had the time and equipment, we would use a form of Attentional Encoder-Decoder Recurrent Neural Network, as discussed in Nallapati et al’s study [4]. This option greatly increased the performance, however we are unable to run this model and this model has not been tested with use of multiple languages. This is the same with the Seq2Seq model [5]. While this approach looked promising we ultimately decided not to use it because of resource concerns both in terms of processing time in Google colab and constraints on the developer’s time. The team also felt that modifying the text would not be as helpful to our end users. Furthermore, using this new model created text in the summary always contains a risk of misleading the end user or misunderstanding the original material since does not take into account different languages.

There are multilingual extractive text summarization techniques, for instance “SimFinderML” and “MINDS”, however both methods require “developing modules conforming to interfaces for text pre-processing and primitive extraction for language”, and also requires one to “produce a shorter text in the same language that contains all the main points in the input text” through using a core engine which is generally “language independent” [6]. These models are great, however for the purposes of our model it is too ‘data expensive’ to be able to analyze the text and create a summary in an already different language.

For analyzing the summarization, Fang et al’s [7] word-sentence co-ranking for automatic extractive text summarization is closer to what we would like to implement. It takes into account modeling methods like TF-IDF features, however even though the model works in at least 2 different languages (Chinese and English), it does not take into account translating the models into a different language. Another implementation is Marek et al’s paper revolving around using named entities [8], however due to our dataset being very broad, we figured only using named entities is not a good way to analyze the performance.

**3 Proposed Work**

As discussed before, we’ve decided to use extractive model in order to reduce translation errors when translating to and from the summary. Discussing regular extractive methods discussed in the related work, our extractive method takes a much different approach. The method looks to analyze the text and then extract from it the critical original content and deliver it in summary form using cosine similarity. It effectively pulls out the most important subset of sentences. This method falls into the family of unsupervised learning; this means that we have the input data for the model, but we don't have any output set to map it to in testing. This method takes the approach of seeing how similar sentences are and then assigning them various weights so they can be ranked. Once they are ranked the top n sentences are selected and used as a summary. This method is similar to page rank which is the algorithm powering Google Search.

This project was built and run using the free version of Google Colab. This is an important tool that allows the team to collaborate easily and all the team members have plenty of experience using it. It's also a good tool for bundling models since it provides a lot of processing power that a local machine does not have. The one major downside however is that it still does not provide enough processing power for larger jobs. We had to consider the scalability of our model and we did address the limitations of Google Colab as a real world problem in deciding which approach to pursue.

The basic design was to build an extractive text summarization model using a .ipynb notebook and running it on Colab. The layout for the notebook is fairly standard; first, all the libraries needed for the project are imported. Then the source text is uploaded. Originally, we uploaded .csv files with each row containing a different text, however we had issues importing the .csv file to the google drive due to the file being too massive. For this project, the simplest way to do it was to upload .txt files directly from our local machines into Colab. (However, if the dataset had gotten bigger this would no longer be a good option.) Once the data is loaded the next steps are to tokenize the sentence and then use NLTK and Pandas to clean them, for example, removing stop words and capitalization. After this is completed we created a feature space where all the sentences collected and then we run cosine similarity on the sentences. We did this by pulling in a data set from Stanford NLP that is basically a huge list of words. Each word gets a value and is added to an array. After that, we use our tokenized and clean-up input text to create sentence vectors. This creates a huge array of scores. Finally, we loop through the array running cosine similarity on the sentence vectors. Once that is complete the sentences are ranked and the first n number of sentences are printed as the summary. We also translate the summary back to its original language at the end.

The evaluation of this model has been a challenge. We can easily check the input vs the output text and while these results look good it's in a sense the best and worst way to evaluate how the model works. If an end-user can read the summarization and understand the text then this shows the model is working. The issue here is that this evaluation is completely subjective. So the team has looked into other more objective ways to evaluate the model. This took a little extra bit of working with the model to find a good way to evaluate it. We’ve decided to focus on the cosine similarity of each sentence and also using topic modeling in order to analyze the sentences summarization.

Finding a way to evaluate this model was very trickey. So we looked into ways to evaluate the page rank model. one of the evaluation metrics we came across was MAP which stands for mean average precision.

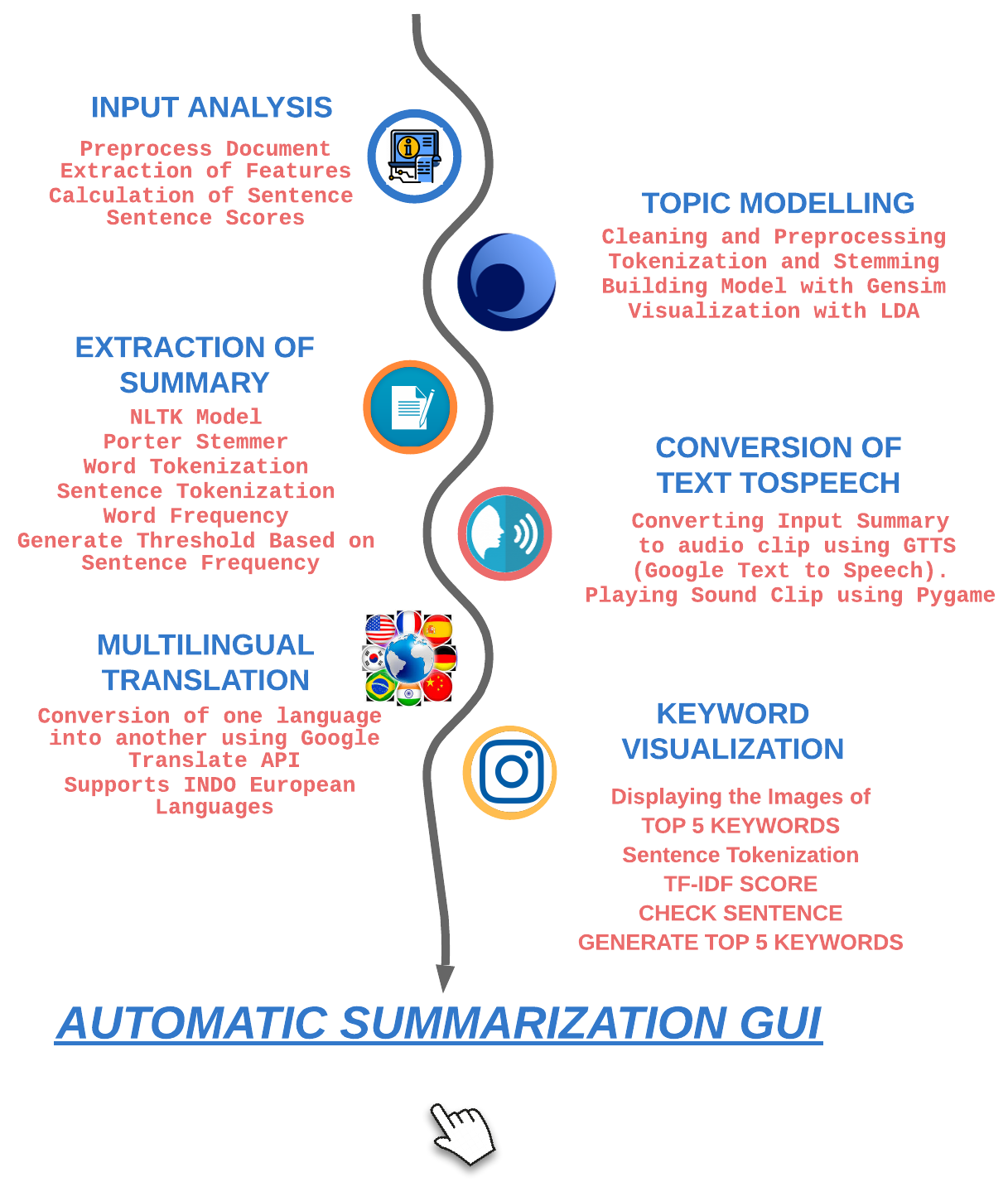


Fig 3: Text Summarization Outline

So we used the average precision score from the sklearn metrics library. We loaded in our cousin scores with a full vector of best possible scores. which in this case is 1. the output from the function returned 1. Which means our model is running very well. However the team is very cautious about these exciting findings. While at first glance this looks good. it's possible our model is overfitting or that trying to use evaluation metrics from page rank did not work as well as we had hoped and is now giving us a false positive score.

**4 Implementation and Evaluation**

Github: <https://github.com/brucker3/CS5560_proejct>

note\*

still need to add evaluation statistics and results

**4.1 Topic Modeling**

Topic Modeling is a type of statistical model that is used to discover abstract objects or hidden semantic structures in a collection of documents

For our project, we will perform topic modeling on some of the files we will be using for text summarization..This can aid when we need to find some specific words in our text which may be hidden. Topic modelling will help us see what topics the words fall under. We followed a structured workflow to build insightful topic models based on LDA. For this project, we will build the topic model based on the LDA algorithm

In performing our topic modeling, we imported packages like nltk which contributed to identifying and removing stopwords from our text ,pyLDAvis and gensim models which was used for building the model and performing visualization of the topics. The text was tokenized before implementing LDA.

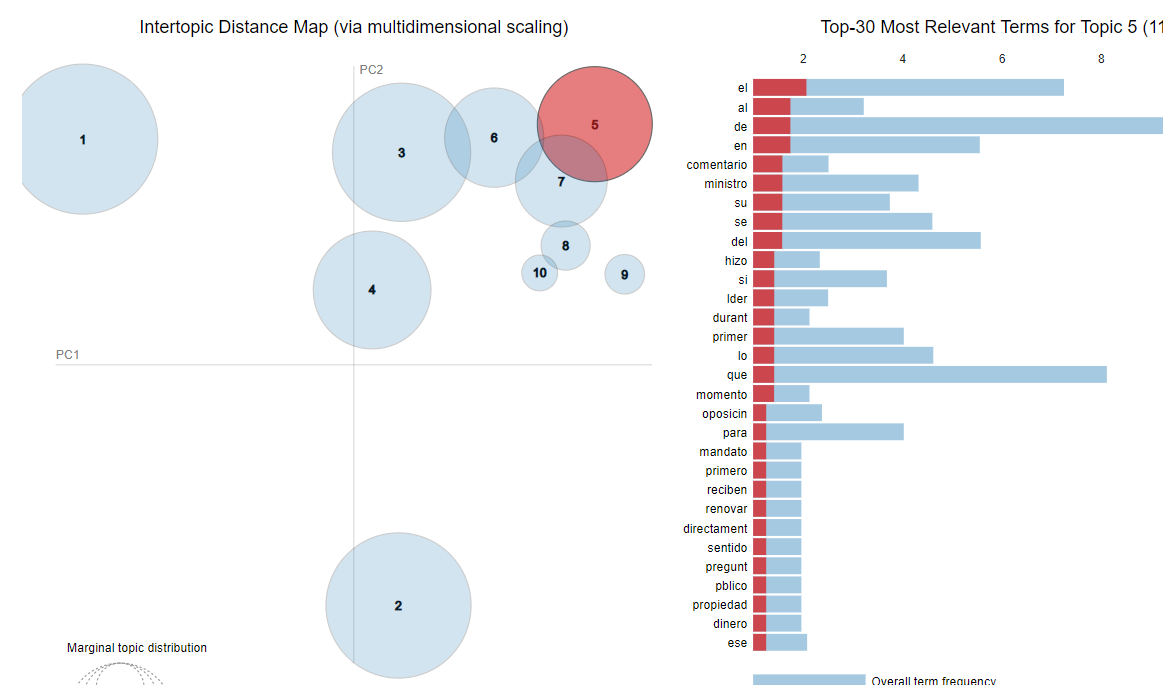
**Hyperparameters used**

**num\_topics=10, alpha=0.05, eta=0.61, num\_words=5**

**pyLDAvis**

LDA visualization is the most common and returns a nice visual of information contained in the topic model

.We used LDA for visualizing the results as seen below



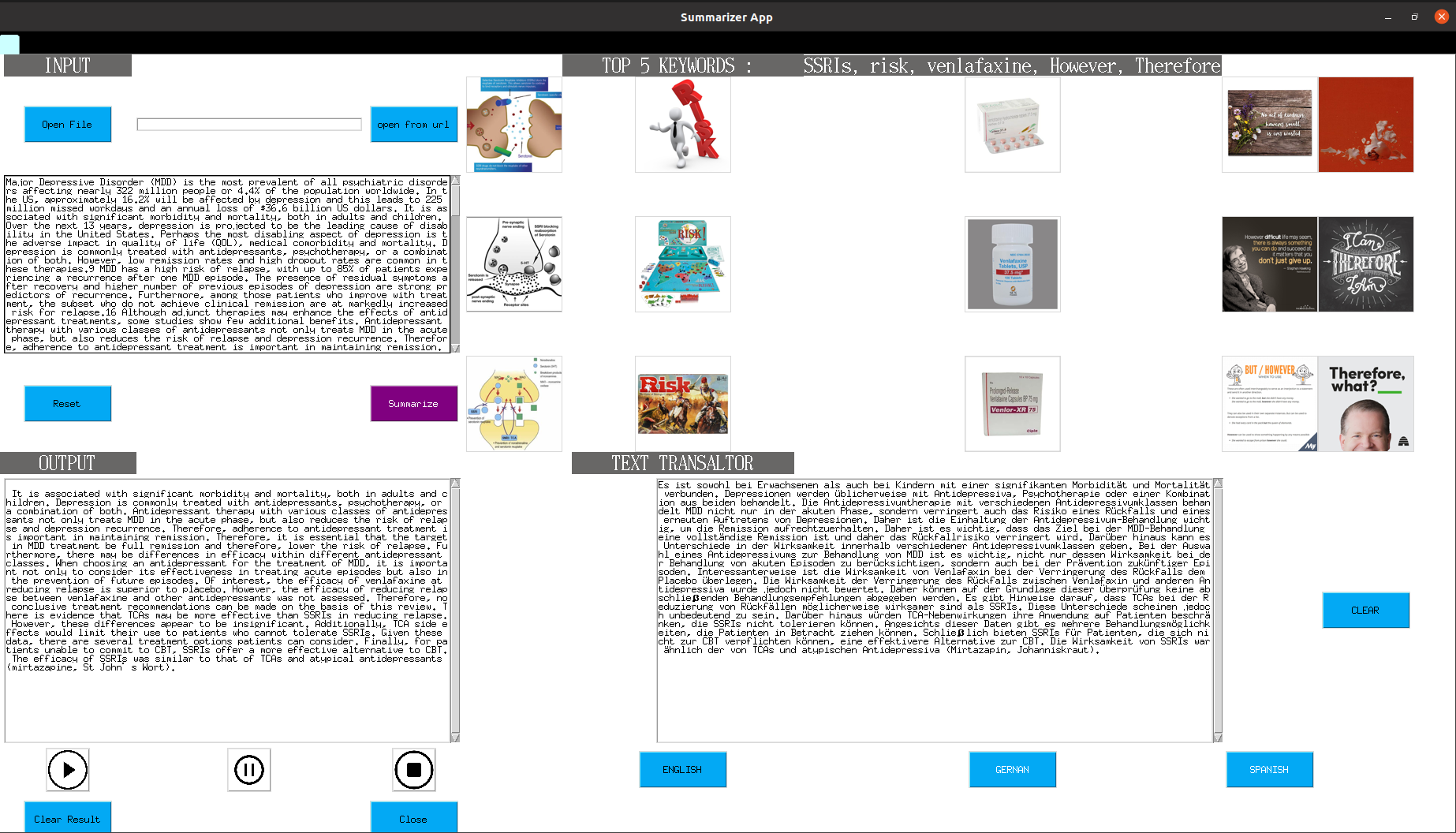
*Fig 4.1 LDA VISUALIZATION OF SPANISHARTICLE*

**Observation**

The visualization shows the topics in circles to the left and the words together with the term frequency on the right. we notice topic 1 and 2 are very spaced out together with topic 4 and 9 and they do not have overlaps but we can see the other topics are overlapping. The larger the circle, the higher the frequency of the words.

**4.2 GUI Application Functionality**

We created a Graphic User Interface which can receive a text file and provide the summary as our codes did in google colab. This Application is capable of performing text-to-speech as well.



*Fig 4.2 Screenshot of GUI application*

Application is useful for those who are very busy and then do not find time to read the entire article. It extracts the summary based on the entire article, so that they can get the acquired content. Our application provides an audio-visual facility, where we can play the content summary result and can pause-stop it anytime whenever we want. One can listen to when the play button is on and this will also help visually impaired people to listen to the entire article.

Our proposed GUI extracts TOP 5 keywords and it represents them in the form of Image. So one can read and understand the article just by looking at the images displayed and clarify the thought process. This application has the highest impact for the people who are lazy for reading the entire article, visually impaired for ones who are blind and one who loves reading summary in the form of images.

It also has a Multilingual Translator, this GUI converts script or languages into Indo European Languages such as English, Spanish, German, French and many others.

There are 7 parameters into considerations for building and implementing our TEXT Summary GUI. These parameters are as follows:

**INPUTS:**

**Three input sources are summarized as below.**

1. **OPEN FROM FILE** : One can upload the text file from the input open file type and generate the summary.
2. **OPEN FROM URL** : One can extract summary based on the URL.
3. **TEXT BOX :** One can type the article manually and can also copy-paste the entire original contents.

**OUTPUTS :**

**SUMMARY GENERATOR** : Summary can be extracted from the article by any one of the input methods. GUI represents the summary extractor based on the model defined in our system.

**TEXT-TO-SPEECH:** The integration of TEXT-TO-SPEECH in this application provides an audio for the extracted summary. It helps visually impaired people to listen to the summary by just pressing the play button. This integration comes with ;

**PLAY**: To start listening to the audio

**PAUSE:** To pause the audio

**STOP:** To stop the audio

**MULTILINGUAL TRANSLATOR :** What if one cannot understand other languages except mother tongue? This application provides a way to convert the extracted summary from native language into INDO European languages.

**KEYWORD VISUALIZATION :** Can we generate TOP 5 keywords based on the article. YES, we can extract TOP 5 keywords based on our article by running this GUI. Those keywords are converted into image visualization to clearly and easily understand the picture of the summary.

**OTHER BUTTONS :**

**RESET AND CLEAR :** This button will help us to clear and reset the input text box and we can open new file from the computer or copy-paste new content or can upload URLs and generate the summary out of it.

We tested our application on 3 documents (document1, document2 and a SpanishArticle). These documents and their summaries are provided in github.

**4.2 Evaluation and Results:**

6 **Conclusion**

One of the big challenges we faced was changing our original dataset we had originally planned on using a large dataset of CNN stories and questions in a .csv file. But this dataset was too big for our local machines and was giving Colab issues as well. So we had to look for other ways to get news sources. This led to finding articles and simply converting them to .text files. This process was more manual than we would have liked but it works well for our model. However, we were able to implement the model in a GUI application. We also had issues with choosing our model and ended up going through a couple of different ideas for how to build and implement it. While the idea of an abstractive model was appealing, the team felt that the extractive model gave certain advantages to our end-user, like being able to pull quotes directly from the summary. This did mean however that we had to add an assumption about our end-user that they would have to have some baseline understanding of the text. After building and running this model against several articles the team is pleased with the summarization generated by the model and feels that the end-user would benefit from using it as well.

**REFERENCES:**

[1]Tas, Oguzhan, and Farzad Kiyani. "A survey automatic text summarization." *PressAcademia Procedia* 5.1 (2007): 205-213.

[2] Barzilay, Regina, and Michael Elhadad. "Using lexical chains for text summarization." *Advances in automatic text summarization* (1999): 111-121.

[3]Deshpande, Anjali R., and L. M. R. J. Lobo. "Text summarization using clustering technique." *International Journal of Engineering Trends and Technology* 4.8 (2013): 3348-3351.

[4] Nallapati, Ramesh, et al. "Abstractive text summarization using sequence-to-sequence rnns and beyond." *arXiv preprint arXiv:1602.06023* (2016).[1602.06023.pdf, (arxiv.org)](https://arxiv.org/pdf/1602.06023.pdf,)

[5]Khatri, Chandra, Gyanit Singh, and Nish Parikh. "Abstractive and extractive text summarization using document context vector and recurrent neural networks." *arXiv preprint arXiv:1807.08000* (2018). [Abstractive and Extractive Text Summarization using Document Context Vector and Recurrent Neural Networks (arxiv.org)](https://arxiv.org/pdf/1807.08000.pdf)

[6][jetwi0203.pdf (psu.edu)](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.348.3840&rep=rep1&type=pdf#page=104) Gupta, Vishal, and Gurpreet Singh Lehal. "A survey of text summarization extractive techniques." *Journal of emerging technologies in web intelligence* 2.3 (2010): 258-268.

[7][Word-sentence co-ranking for automatic extractive text summarization - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S0957417416306959)Fang, Changjian, et al. "Word-sentence co-ranking for automatic extractive text [8]summarization." *Expert Systems with Applications* 72 (2017): 189-195.

[8]P. Marek, Š. Müller, J. Konrád, P. Lorenc, J. Pichl, and J. Šedivý, “Text Summarization of Czech News Articles Using Named Entities,.” Accessed: Apr. 25, 2021. [Online].Available: <https://arxiv.org/pdf/2104.10454.pdf>.

[9Bhatia, Neelima, and Arunima Jaiswal. "Trends in extractive and abstractive techniques in text summarization." *International Journal of Computer Applications* 117.6 (2015).

[10]Paulus, Romain, Caiming Xiong, and Richard Socher. "A deep reinforced model for abstractive summarization." *arXiv preprint arXiv:1705.04304* (2017).