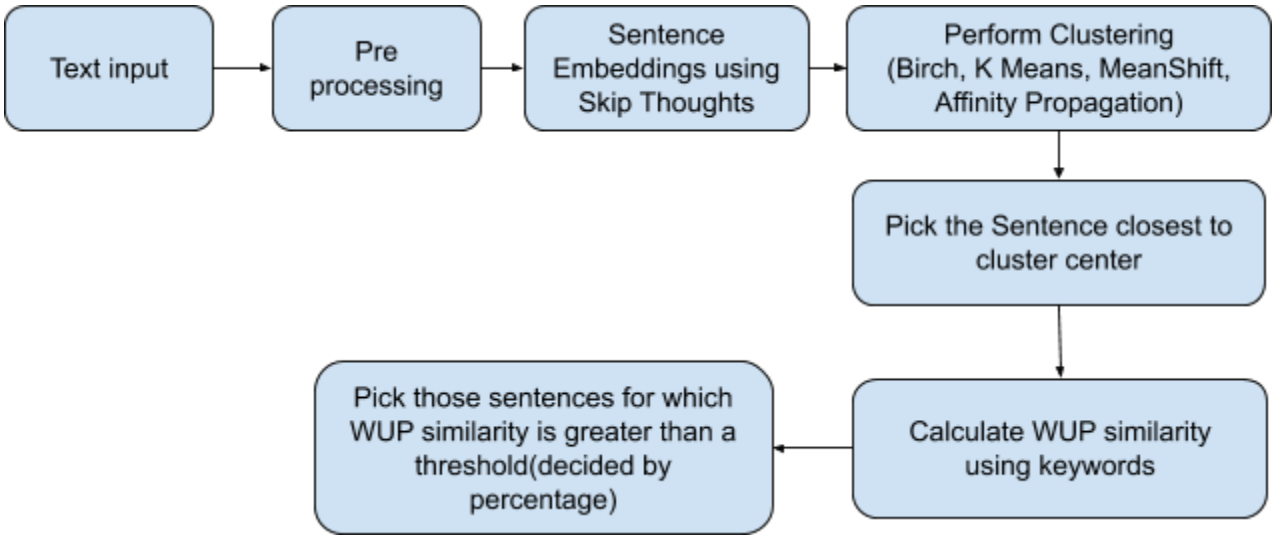
ML Summarization

Introduction

Given a document(plain text) , percentage, and keywords as an input to the module, a summary is generated. The algorithm uses sentence embeddings and performs clustering on these embeddings to generate a summary. Only unsupervised learning is tried since for supervised learning we need a large amount of data and its corresponding summary.

Workflow



Algorithm

➔ **Generate Sentence Embeddings:** We use skip thoughts[1] which is an encoder-decoder framework to generate sentence embeddings. It generates a fixed-length vector representation of the sentences. The algorithm uses the order of the sentence and its neighboring sentences to train itself due to which sentences which share similar semantic and syntactic properties are mapped to a similar vector representation which is better. The other sentence representations such as universal sentence embeddings or using word embeddings from Skip Gram Word2Vec or BERT embeddings and convert them to sentence embeddings by assuming the bag of words model won't give better results since our goal is to extract the meaning of the sentence in our vector representation.

➔ **Clustering of the sentence embeddings:** We will now divide these vector representations into a predefined number of clusters. The number of clusters will be the number of sentences in our summary. Each cluster will have a set of semantically similar sentences. The cluster center will be the sentence which we want to present in our summary. By analysis, it was found that if the number of clusters is half as the number of sentences present in the document the summary quality is high and it covers almost the whole aspect which the document signifies. If the number of clusters are more than half the number of sentences present in the document we get some sentences that represent the same meaning in the summary. Hence we have chosen our

optimal value to be 0.52. Now for clustering we are using many of the clustering algorithms like KMeans, Birch Clustering, Affinity Propagation, Mean Shift. Affinity Propagation and Mean Shift don't require a number of clusters to be initialized beforehand. Hence we have also tried a combination of KMeans with Affinity Propagation and we fed the number of clusters generated by Affinity Propagation(AP) to KMeans. Similarly, we can do with Birch and Affinity Propagation. After performing clustering we now pick the sentence which is closest to the cluster center. This is done for all labels. We also perform ordering on these sentences. The way in which sentences of a cluster are positioned in the document is taken into account and a summary is generated. Eg- If most of the sentences which are present in the cluster occur at the end of the document then that sentence(which has to be present in summary) will come in the end.

➔ **Using Keywords and percentage:** For the given keywords we generate their hypernyms. For every sentence which is present in cluster summary, we compare the hypernyms of words present in the sentence with hypernym. In this scenario, we are taking that value which is most similar. Now out of all values, we take a minimum of these values, and If this value is greater than the threshold value(decided by the percentage of summary) we include the sentence in our summary.

Results:

F1 - Score of Rouge-1

Pdf	Birch	KMeans	AP	AP+Birch	AP+KMeans	MeanShift
Deloitte Investment	0.52	0.48	0.48	0.31	0.28	0.09
Deloitte Wealth	0.51	0.54	0.45	0.14	0.17	0.08
Regtech-in-financial services	0.62	0.60	0.63	0.18	0.15	0.10
Wealth Management	0.33	0.28	0.34	0.10	0.07	0.05
Tech Wealth	0.42	0.50	0.41	0.25	0.30	0.02
IDFC dynamic	0.14	0.15	0.15	0.18	0.18	0.05
AWM	0.58	0.43	0.50	0.3	0.25	0.04
IDFC Asset	0.58	0.50	0.19	0.09	0.08	0.06
wms	0.20	0.34	0.24	0.19	0.23	0.10
Fintech	0.56	0.25	0.56	0.13	0.08	0.11
IDFC Securities	0.18	0.10	0.11	0.15	0.17	0.06
Regtech Revolution	0.38	0.31	0.32	0.21	0.19	0.09

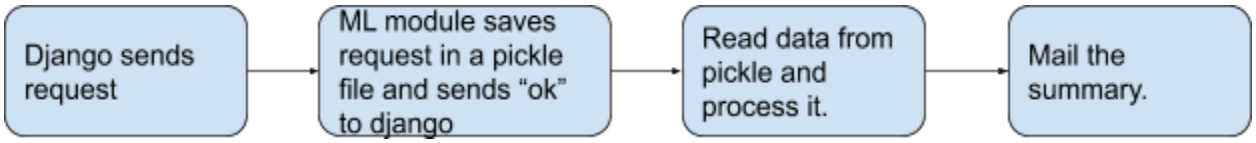
Observations-

➔ **Birch clustering** seems to perform better than all other methods of clustering implemented.

➔ If we fix the number of clusters with AP(i.e. Affinity Propagation) and perform either KMeans or Birch the quality of summary goes down and so does the rouge.

Integrate with System

The usual way of sending a request, processing it by generating summaries, and then sending it back didn't work for ML summary since it was computationally time expensive. The request consisted of the filename, keywords, data, and email of the person. The data is the raw text which we get after pdf parsing. Hence we tried various ways of resolving this issue. We are now saving the request in a pickle file. We send "ok" as a response to the Django server. Now we will read data from the pickle file, generate a summary and we will email the summary to the user.



References:

[1] skip-thoughts paper