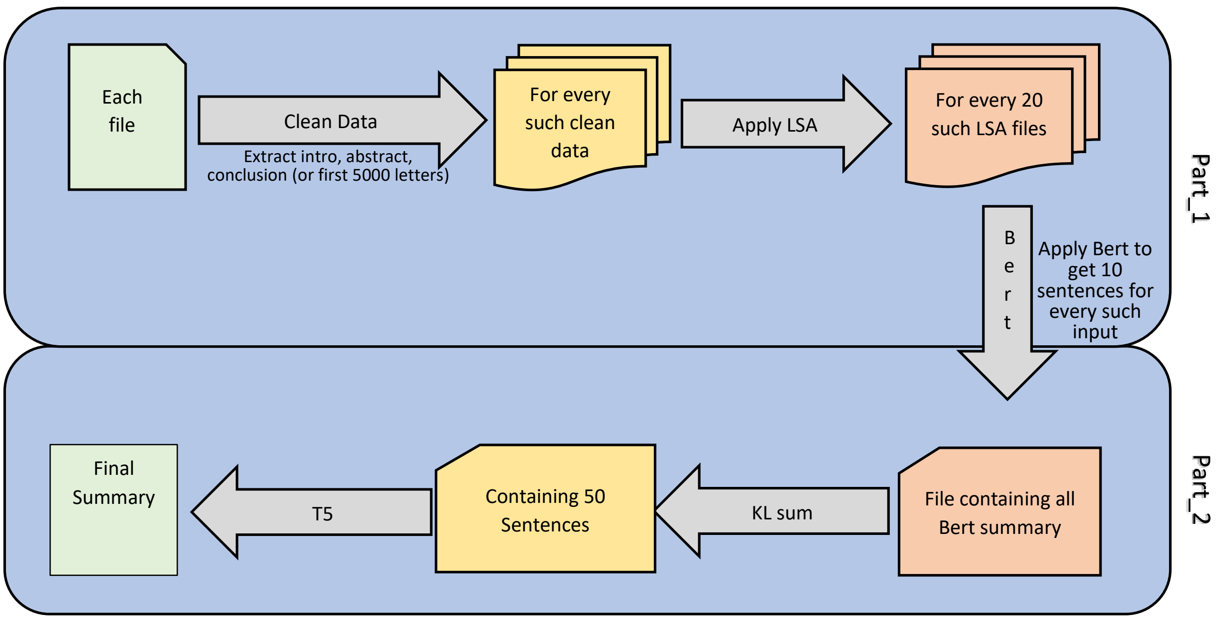
General Overview of pipeline:

(Refer to the explanation Section for to the point explanation)


Each file
Clean Data
Extract intro, abstract, conclusion (or first 5000 letters)
For every such clean data
Apply LSA
For every 20 such LSA files
Bert
Apply Bert to get 10 sentences for every such input
Part_1
Part_2
File containing all Bert summary
KL sum
Containing 50 Sentences
T5
Final Summary

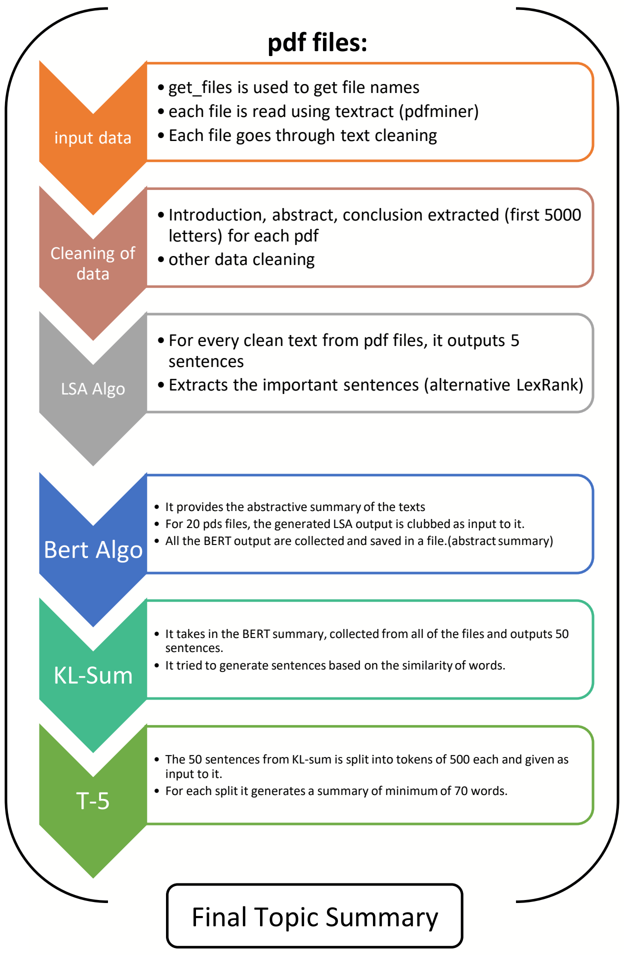
Code workflow:

Pipeline:

1. **input data:** for txt and pdf we get the file names using get_files and for csv we get the dataframe
2. **clean data:** We extract the introduction, abstract and conclusion if present in the file, else we take the first 5000 letters. The number of files to be applied on depends on the file type, (refer the precise pipeline section)

   Note: only 5000 letters of abstract and introduction (combined) are taken (if present) from each file
   - Then do the text cleaning. In this the text inside brackets , digits, brackets and quotations are removed. Also we convert words like ain't to is not etc. Stopwords are removed and letters smaller than 2 letters are removed.
   - Full stops , commas and semicolons are not removed.
3. **LSA applied:** It selects the most frequent words among the text and generates the output summary. We have set the number of output sentences to 5 or 4 depending upon the number of files in a topic.
   - For pdfs, this summarization is done on the cleaned text for each files one at a time. So we get about 5 sentences containing most common words from each file.
4. **Then Bert applied:** Then we collect the 20 output from the LSA and collectively apply Bert on that. As this is abstractive summarization , so it provides us the overall picture from the most frequent words.
   - This is where the part_1 ends and we also save the abstractive summary from bert.
   - Though we also save the extractive summary from LSA, but didn't get time to explore it. The LSA summary is already taken as input to BERT.
5. **KL applied :** It takes in the bert summary as input and outputs the 50 sentence summary . This is based on KL-divergence and is an extractive summary method to get the most similar sentences together.
6. **T5 applied:** The output from KL algo is taken as input. As it can only take 512 tokens as input, so we divide the input into parts with each part having 500 tokens. And each part generating minimum 70 words of summary.
7. This is saved as final summary.

Precise pipeline according to file type:



pdf files:

input data
- get_files is used to get file names
- each file is read using textract (pdfminer)
- Each file goes through text cleaning

Cleaning of data
- Introduction, abstract, conclusion extracted (first 5000 letters) for each pdf
- other data cleaning

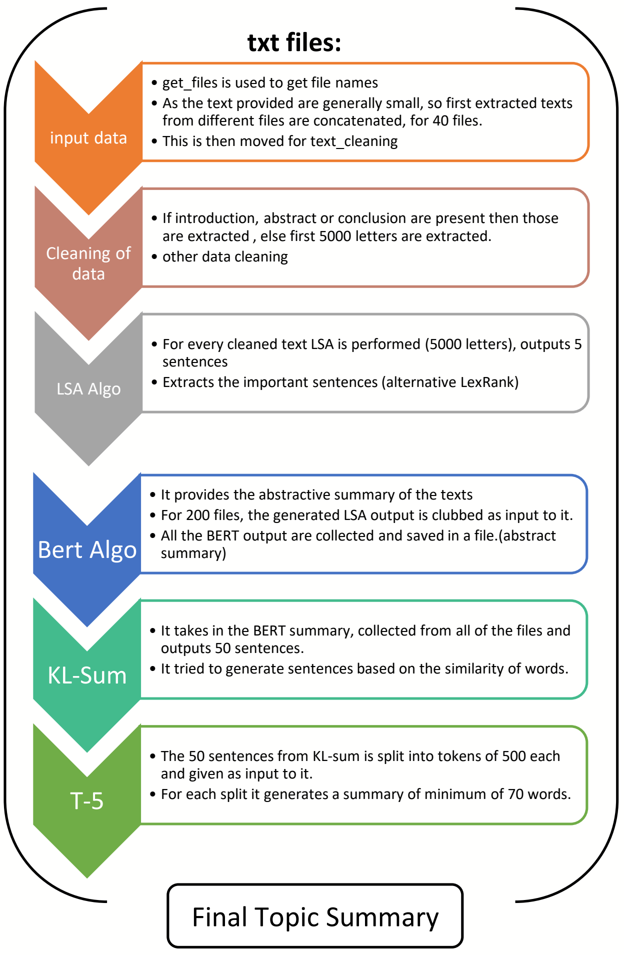
LSA Algo
- For every clean text from pdf files, it outputs 5 sentences
- Extracts the important sentences (alternative LexRank)

Bert Algo
- It provides the abstractive summary of the texts
- For 20 pds files, the generated LSA output is clubbed as input to it.
- All the BERT output are collected and saved in a file.(abstract summary)

KL-Sum
- It takes in the BERT summary, collected from all of the files and outputs 50 sentences.
- It tried to generate sentences based on the similarity of words.

T-5
- The 50 sentences from KL-sum is split into tokens of 500 each and given as input to it.
- For each split it generates a summary of minimum of 70 words.

Final Topic Summary



txt files:

input data
- get_files is used to get file names
- As the text provided are generally small, so first extracted texts from different files are concatenated, for 40 files.
- This is then moved for text_cleaning

Cleaning of data
- If introduction, abstract or conclusion are present then those are extracted , else first 5000 letters are extracted.
- other data cleaning

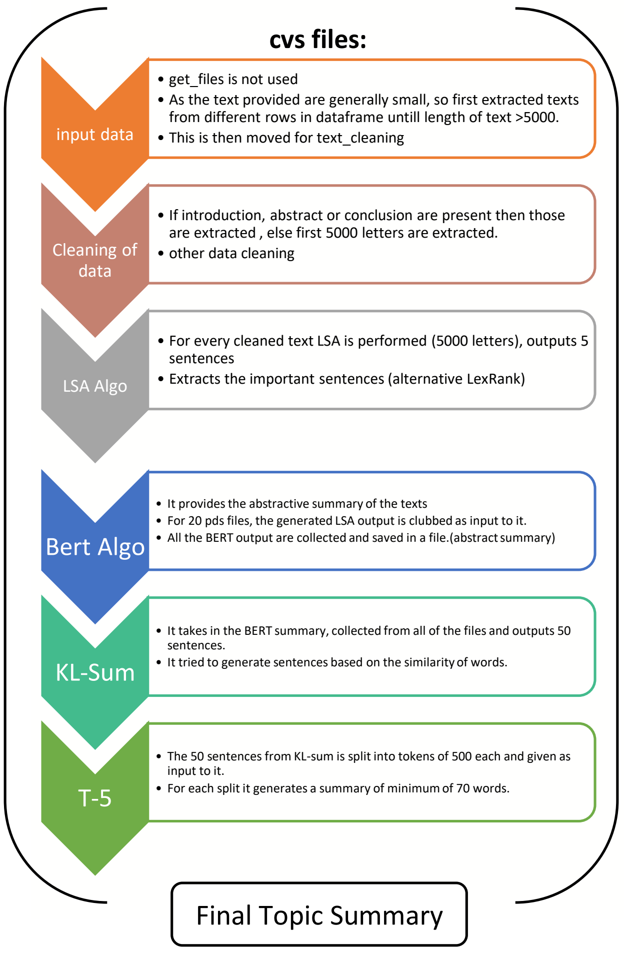
LSA Algo
- For every cleaned text LSA is performed (5000 letters), outputs 5 sentences
- Extracts the important sentences (alternative LexRank)

Bert Algo
- It provides the abstractive summary of the texts
- For 200 files, the generated LSA output is clubbed as input to it.
- All the BERT output are collected and saved in a file.(abstract summary)

KL-Sum
- It takes in the BERT summary, collected from all of the files and outputs 50 sentences.
- It tried to generate sentences based on the similarity of words.

T-5
- The 50 sentences from KL-sum is split into tokens of 500 each and given as input to it.
- For each split it generates a summary of minimum of 70 words.

Final Topic Summary



cvs files:

input data
- get_files is not used
- As the text provided are generally small, so first extracted texts from different rows in dataframe untill length of text >5000.
- This is then moved for text_cleaning

Cleaning of data
- If introduction, abstract or conclusion are present then those are extracted , else first 5000 letters are extracted.
- other data cleaning

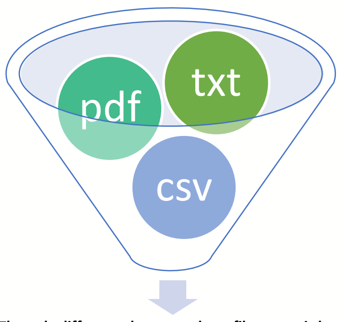
LSA Algo
- For every cleaned text LSA is performed (5000 letters), outputs 5 sentences
- Extracts the important sentences (alternative LexRank)

Bert Algo
- It provides the abstractive summary of the texts
- For 20 pds files, the generated LSA output is clubbed as input to it.
- All the BERT output are collected and saved in a file.(abstract summary)

KL-Sum
- It takes in the BERT summary, collected from all of the files and outputs 50 sentences.
- It tried to generate sentences based on the similarity of words.

T-5
- The 50 sentences from KL-sum is split into tokens of 500 each and given as input to it.
- For each split it generates a summary of minimum of 70 words.

Final Topic Summary



The only difference between these files types is how they are collected and processed before they are sent to cleaning. The pipeline after that is same.

Functions:

1. **text_cleaning(text):** It is used for cleaning of text. Functionality already explained in Cleaning text.

2. **def get_files(topic):**
   Input: topic name
   Output: list containing the name of pfd and txt files

3. **fun_lsa(text):**
   Input: clean text,
   Output: (string) output sentences
   Implements the LSA algorithm, no of sentences fixed at 5

4. **def fun_t5(text):**
   input: (string) text
   output: (string) output summary
   By default: (as we needed the summary to be between 150 – 200 words)
   Min_length = 70 (no of tokens)
   Max_length = 150
   Penalty = 3
   Model used: t5-large

5. **def fun_kl(text)**
   Input: text (string)
   Output: (string) output summary containing 50 sentences

6. **def fun_part_1(files,topic,iteration):**

   input:
   files: (list) name of files or (dataframe) for topic_6
   topic: (string) name of topic
   iteration: (int) to keep the experiment count
   output: (string) output summary
   This function cleans the text, applies LSA and Bert algorithm to the texts and collects the summary.
   It reduces the whole topic into just 50 sentences, and also saves it.
   It also saves the complete LSA summary in case to use in another experiments.

7. **def fun_part_2(abs_summary,topic):**
   input:
   abs_summary: (string) summary from part one
   topic: (string): topic name
   Output: (string) final summary
   It applies Kl sum and T5 algorithm and generates the final summary.

Explanation / feature engineering :

Assuming average word size as 10
So 5000 letters approximates to 500 words.

Note: As the number of pdfs are in majority so explanation is given keeping that in view. Some parameters may vary if we take txt or csv files. But the process and explanation remains the same. (refer to precise pipeline section for the difference)

First, the pipeline tries to identify abstract, introduction and conclusion from each file. If they are present then, it extracts the first 5000 letters from intro and abstract. We also add some words from conclusion.
Reason : Dealing with such huge number of files, we assume that most of the information from the file should be present in the introduction and abstract.

It these are not present we simply take the first 5000 letters. The process tried to extract the text such that the chunk that we put into cleaningis around 5000 letters on an average, across the formats pdfs, txts and csvs. SO the pipeline is designed in such a way, that it takes care of that.

Second, **In the text cleaning**, we remove stopwords, brackets, digits, words smaller than two letters, etc. The usual text cleaning that is done for NLP.

In this model, we have removed stopwords for some topics where as kept for some: Each one has its merit and demerit.
Merit: If stopwords are removed then the final summary does give better summary of the topic, and the sentences also makes sense.
Demerit: The summary lacks helping verbs like is , am are, etc so the sentence is not grammatically correct.
On the other hand, if we keep the stopwords:
Merit: The final summary looks very real and human like.
Demerit: But the overall summary of the topic is not covered. It somehow missed many points from the topic.

Third, applying LSA algorithm in each file provides us the summary of that file. As a result we get the most important information from each file in just 4-5 sentences.
Imporance: This helps in drastically reducing the amount of text, while retaining the important information.
We then collect these sentences together for 20 such pdf files and apply Bert on them.(This number is a bit ditterent for txt and csv files)
Reason: Bert is an abstractive summary method. It will provide the abstract view of the (20 X 5) sentences, In just 10 sentences. So, we basically reduce the 20 pdfs to 10 sentences, just keeping the important information.
BART and GPT were also experimented with, but only with a small sample. As bert gave better result on that sample so BERT was selected. From the research point of view, we can also try these on the overall dataset and can compare the results.

Forth, the collective output from bert is collected and is then fed in KL-sum algorithm. It outputs 50 sentences for the collective bert input.
Reason: KL-sum is an extractive summarization method. It gives important to a word on how similar it is to other words. So, it tries to give retain the most important information. As a result topics which have occurred more across the files will be selected.

Final, the 50 sentences from KL-sum goes into T5 algorithm (t5-large) and gives us the final summary. As T5 model takes in tokens of size 512, so the input is first divided into parts of 500 tokens each and the summary is generated for each part. With minimum length 70 words.
Reason: The algorithms before T5, provides us the very important and most frequent occurring features and information. But T5 generates and presents the information in precise and in a semantic way, presenting us the overall summary.