Natural Language Processing
AI-829

Nested Summarization with Heading Hierarchy

Mandate-4 Report

Presented By:
Aprajita Kumari(MT2023037)
Nandini Yadav(MT2023096)

Contents

1. Recap of Syntactic Processing……..3
2. Overview……4
3. Loading Data/Formatting Data……5
4. Train -Test Split……7

5. Text Summarization……..……8
   5.1 Fine-tuning BART in Our Project

6. Title Generation……..…12
   6.1 Pre-Processing……..…12
   6.2 Sentence Clustering……..…13
   6.3 Topic Modeling……19
   6.4 Tuning BART and using Seq2SeqBatchTokenize…26

7. Conclusion……..28
8. Challanges Faced……..…..31
9. Collab Notebook……31

1.Recap of Syntactic Processing

- The code uses special tools like NLTK, Stanza, and spaCy to understand text better. These tools help it look at words and sentences in different ways, like a detective examining clues.

- It figures out what each word is doing in a sentence. Is it a person, an action, or something else? This helps the code understand the sentence better.

- It also looks at how words are connected in a sentence. Like, who is doing what to whom? Understanding these connections helps make sense of the sentence.

- The code then makes parse trees to show these connections visually.Seeing the structure of the parse tree helps us understand how the words are related.

- It also tries to understand when different words are talking about the same thing. For example, if a person is called "he" and "John" in the text, it knows they're talking about the same person.

- Making all these techniques work together smoothly is important, even though it can be tricky sometimes. But when they do work together well, it helps us understand text much better.

2. Overview

- This project revolves around condensing extensive articles into summaries while simultaneously generating headings to encapsulate the main points of the content.

- Our project addresses this challenge by leveraging advanced natural language processing (NLP) techniques to automate text summarization and topic modeling tasks. The primary objective is to develop a system capable of summarizing documents and identifying key topics within them, thereby enhancing decision-making processes across various domains.

- An important aspect of the mandate is fine-tuning the pre-trained BART (Bidirectional and Auto-Regressive Transformers) model for text summarization tasks.

- Latent Dirichlet Allocation (LDA) is implemented for topic modeling.

- In addition to topic modeling, our project involves generating titles or headings for the documents based on the extracted topics.

3. Loading Data/Formatting Data

We are formatting the dataset which are a set of articles stored in separate files by putting them all together in a data frame.

```
[ ]
    from os import listdir
    from string import punctuation
    punctuation+='\n'
    import re
    from bs4 import BeautifulSoup
    import pandas as pd
    import math
    from tqdm import tqdm
```

```
[ ] def load_doc(filename):
        file = open(filename, encoding='utf-8')
        text = file.read()
        file.close()
        return text
```

```
def split_story(doc):
    index = doc.find('@highlight')
    story, highlights = doc[:index], doc[index:].split('@highlight')
    highlights = [h.strip() for h in highlights if len(h) > 0]
    return story,
```

```
[ ] def load_stories(directory):
        all_stories = list()
        for name in tqdm(listdir(directory)):
            filename = directory + '/' + name
            doc = load_doc(filename)
            story, highlights = split_story(doc)
            all_stories.append({'story':story, 'highlights':highlights})
        return all_stories
```

```
directory = '/content/cnn/stories'
data = load_stories(directory)
print('Loaded Stories %d' % len(data))
```

```
100%|██████████| 119/119 [00:00<00:00, 15099.75it/s]Loaded Stories 119
```

```python
directory = '/content/cnn/stories'
data = load_stories(directory)
print('Loaded Stories %d' % len(data))
```

```
100%|██████████| 119/119 [00:00<00:00, 15099.75it/s]Loaded Stories 119
```

```python
stories = [story['story'] for story in data]
```

```python
data_df = [[d['story'], d['highlights']] for d in data]
```

```python
df = pd.DataFrame(data_df).rename(columns={0:"text", 1:"summary"})
df.head()
```

	text	summary
0	Greenville, South Carolina (CNN) -- Some are girlish 22-year-olds; others are women approaching 40. They come from South Carolina's rural counties and its booming cities. They are loud and muted, lively and vacant, hopeful and desperate.\n\nAs different as they are, they share a connection to two powerful forces : their addictions and their babies.\n\nThey are swallowed by the same shameful past. They don't know if they can be good mothers. They don't know if they can be clean mothers.\n\nThey're here at a state-run drug-treatment program to learn how to do both.\n\nOn this summer day, Ash...	[Serenity Place in South Carolina caters treatment programs to pregnant addicts, South Carolina has been known to prosecute women in their third trimester, A pregnant Ashley Hendrix sought treatment earlier this year for her addiction, "If these women don't get this treatment, they are going to die," a counselor says]
1	Riverside, California (CNN) -- Authorities said Wednesday they are reasonably sure that the body found inside the burned cabin near Big Bear Lake, California, is that of Christopher Dorner, the rogue ex-cop who had been pursuing a vendetta against his fellow officers.\n\"We believe that this investigation is over, at this point, and we'll just need to move on from here," San Bernardino Sheriff John McMahon told reporters.\n\nAlthough the description and behavior of the man who was killed are consistent with Dorner, officials "cannot absolutely, positively confirm it was him," McMahon	[Slain officer's widow thanks mourners: "A lot of people loved Mike", Villaraigosa says police have a "reasonable belief" that Dorner died in a mountain standoff, Authorities have not conclusively identified the body found near Big Bear Lake]

```python
df['text'] = df['text'].apply(lambda x: x.replace('\n',''))
df['text'] = df['text'].apply(lambda x: x.replace("\"", ""))
df['summary'] = df['summary'].apply(lambda x: ". ".join(x))
df.head()
```

	text	summary
0	Greenville, South Carolina (CNN) -- Some are girlish 22-year-olds; others are women approaching 40. They come from South Carolina's rural counties and its booming cities. They are loud and muted, lively and vacant, hopeful and desperate.As different as they are, they share a connection to two powerful forces : their addictions and their babies.They are swallowed by the same shameful past. They don't know if they can be good mothers. They don't know if they can be clean mothers.They're here at a state-run drug-treatment program to learn how to do both.On this summer day, Ashley Hendrix, 24,...	Serenity Place in South Carolina caters treatment programs to pregnant addicts. South Carolina has been known to prosecute women in their third trimester. A pregnant Ashley Hendrix sought treatment earlier this year for her addiction. "If these women don't get this treatment, they are going to die," a counselor says
1	Riverside, California (CNN) -- Authorities said Wednesday they are reasonably sure that the body found inside the burned cabin near Big Bear Lake, California, is that of Christopher Dorner, the rogue ex-cop who had been pursuing a vendetta against his fellow officers.We believe that this investigation is over, at this point, and we'll just need to move on from here, San Bernardino Sheriff John McMahon told reporters.Although the description and behavior of the man who was killed are consistent with Dorner, officials cannot absolutely, positively confirm it was him, McMahon said.We're not c...	Slain officer's widow thanks mourners: "A lot of people loved Mike". Villaraigosa says police have a "reasonable belief" that Dorner died in a mountain standoff. Authorities have not conclusively identified the body found near Big Bear Lake
2	(CNN) -- It was a year of crippling -- and creative -- hacks, a year of blockbuster sequels and (yet again) a year of slinging vengeful birds at corpulent pigs.It was also a year in which gaming continued to expand beyond the traditional console to your phone, your tablet, your TV and your Facebook page.Video gaming in 2011 had plenty of ups and downs -- and we're not even talking about the actual video game titles that were released.Here, in no particular order, are our choices for gaming's 10 biggest stories of the year:Sony's PlayStation Network hackedWhile the explosion of social and m...	The top gaming stories of 2011 revolved around hacks, sequels, innovation. It was the year of 3 : "Modern Warfare 3," "Uncharted 3," and "Battlefield 3". Hackers created new uses for Microsoft's motion-controlled Kinect system. "Angry Birds" moved from just a game to pop culture phenomenon
3	(CNN) -- The flagging economy has been the clear-cut No. 1 issue for this year's presidential race.But some other concerns are just timeless.Don't underestimate the pull of safety and security in the minds of American voters, said Candy Crowley, host of CNN's State of the Union and moderator of last week's presidential debate. There is no issue closer to home and hearth than the safety and security of your family. That is the most basic question the federal government has to	Foreign policy is the focus of the third and last presidential debate Monday night. Safety, security have always been important issues to the American public. Decisions on Afghanistan, Iran, Syria might

4. Train -Test Split

We have divided our dataset into two parts training dataset and testing dataset as follows:

Train-test split

```
[ ] train_size = math.floor(0.9*len(df))
    train_df = df[:train_size]
    test_df = df[train_size:].copy()
```

```
[ ] test_df['text']
```

```
107    (CNN) -- North Korea's longtime leader Kim Jong Il, the embodiment of the reclusive state where his cult of
personality is deeply entrenched, has died.He was believed to be 69.Regarded as one of the world's most-repressive
leaders, Kim Jong Il always cut a slightly bizarre figure. His diminutive stature and characteristically bouffant hair
have been parodied by some in the West.He's a mysterious person -- I think by design, said Han S. Park, director of
the Center for the Study of Global Issues at the University of Georgia and a frequent visitor to North Korea. Mystery
is a source of lever...
108    Washington (CNN) -- When President Barack Obama headed to the Jersey Shore this week to tout the recovery
effort from Superstorm Sandy, congressional Republicans sought to keep the nation's focus on Washington.On the same
day Obama played Touchdown Fever with New Jersey Gov. Chris Christie on the Point Pleasant boardwalk and urged
Americans to come spend money in the devastated region, House GOP committee leaders ensured that a series of
controversies dominating the start of the president's second term remained in the headlines.Obama quoted from the Tom
Waits song Jersey Girl popularized b...
109    Kiev, Ukraine (CNN) -- Did a Russian fire the missile that downed Malaysia Airlines Flight 17? That depends on
who you ask.A top Ukrainian official says he has no doubt.Vitaly Nayda, Ukraine's director of informational security,
told CNN the person who shot down the flight was absolutely a Russian. A Russian-trained, well-equipped, well-educated
officer ... pushed that button deliberately, he said.We taped conversations between a Russian officer and his office
in Moscow, Nayda said. We know for sure that several minutes before the missile was launched, there was a report to a
Russian offic...
110    Tacloban, Philippines (CNN) -- The Philippines struggled to bury the dead and get food, water and medicine to
the living Tuesday, four days after Super Typhoon Haiyan claimed untold lives and flattened countless buildings.Right
now, we don't have enough water, typhoon survivor Roselda Sumapit told CNN in Tacloban, a city of more than 200,000
that was flattened by the storm. What they can get may not be clean, she said -- but she added, We still drink it,
because we need to survive.The government's confirmed death toll was 1,774 early Tuesday, said Jose Lampe Cuisa Jr.,
```

5.Text Summarization

5.1 Fine-tuning BART in Our Project:

- In our project, we fine-tune the pre-trained BART model on a specific dataset to adapt it to the summarization task. Fine-tuning involves updating the model's parameters using the dataset to optimize its performance for the summarization task. This process allows the model to learn task-specific patterns and features from the dataset, thereby improving its ability to generate accurate and coherent summaries.
- BART is a powerful transformer-based model known for its effectiveness in various NLP tasks, including text generation and summarization.

Summarization

We use the pre-trained BART model and fine-tuned it for our dataset (CNN-Dailymail).

Fine-tuning BART

BART uses a standard Transformer-based neural machine translation architecture. It uses a standard seq2seq/machine translation architecture with a bidirectional encoder (like BERT) and a left-to-right decoder (like GPT). Here, BART is trained over the cnn-dailymail dataset.

```
[ ]

    !pip install bert-score -q
    !pip install blurr
```

```
Requirement already satisfied: blurr in /usr/local/lib/python3.10/dist-packages (0.4.1)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from blurr) (6.0.1)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from blurr) (2.8.2)
Requirement already satisfied: docopt in /usr/local/lib/python3.10/dist-packages (from blurr) (0.6.2)
Requirement already satisfied: boto3 in /usr/local/lib/python3.10/dist-packages (from blurr) (1.34.91)
Requirement already satisfied: smart-open in /usr/local/lib/python3.10/dist-packages (from blurr) (6.4.0)
Requirement already satisfied: botocore<1.35.0,>=1.34.91 in /usr/local/lib/python3.10/dist-packages (from boto3->blurr)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from boto3->blurr) (1.
Requirement already satisfied: s3transfer<0.11.0,>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from boto3->blurr)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil->blurr) (1.16.0
Requirement already satisfied: urllib3!=2.2.0,<3,>=1.25.4 in /usr/local/lib/python3.10/dist-packages (from botocore<1.35.
```

```
[ ] !pip install ohmeow-blurr
```

```
from transformers import AutoModel, AutoTokenizer  # Import specific modules or classes from the transformers library
import sentencepiece
!pip install rouge
from rouge import Rouge
```

```
Requirement already satisfied: rouge in /usr/local/lib/python3.10/dist-packages (1.0.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from rouge) (1.16.0)
```

```
[ ] import torch
    import pandas as pd
    from rouge import Rouge
    from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

    import plotly.express as px
    import matplotlib.pyplot as plt

    from plotly.offline import init_notebook_mode
    init_notebook_mode(connected=True)
```

```
[ ] from transformers import BartForConditionalGeneration
```

```
[ ] bart_checkpoint = "facebook/bart-large-cnn"
```

```
[ ] device = "cuda" if torch.cuda.is_available() else "cpu"
```

- The code snippet uses the **BART** model and a tokenizer (AutoTokenizer)to generate summaries for a list of input texts. It iterates through the input, processes them, and stores the generated summaries in the 'SystemSummary' list using beam search and specified constraints on length.

```
[ ] tokenizer_bart = AutoTokenizer.from_pretrained(bart_checkpoint,
                                                   do_lower_case=False,
                                                   use_fast=False,
                                                   keep_accents=True)

    model_bart = AutoModelForSeq2SeqLM.from_pretrained(bart_checkpoint).to(device)
```

Double-click (or enter) to edit

```
[ ] bos_id = tokenizer_bart._convert_token_to_id_with_added_voc("<s>")
    eos_id = tokenizer_bart._convert_token_to_id_with_added_voc("</s>")
    pad_id = tokenizer_bart._convert_token_to_id_with_added_voc("<pad>")
```

```
[ ] text=test_df['text']
```

```
%%time

SystemSummary = []

for i, input_text in enumerate(text):

    inp = tokenizer_bart(input_text, add_special_tokens=False,
                         truncation=True, return_tensors="pt",
                         padding='max_length', max_length=1024)['input_ids'].to(device)

    model_output = model_bart.generate(inp, use_cache=True,
                                       num_beams=4,
                                       max_length=70,
                                       min_length=30,
                                       early_stopping=True,
                                       pad_token_id=pad_id,
                                       bos_token_id=bos_id,
                                       eos_token_id=eos_id,
                                       decoder_start_token_id=tokenizer_bart._convert_token_to_id_with_added_voc("<2en>

    decoded_output = tokenizer_bart.decode(model_output[0],
                                   skip_special_tokens=True,
                                   clean_up_tokenization_spaces=False)

    SystemSummary.append(decoded_output)
```

```
CPU times: user 8min 1s, sys: 18.2 s, total: 8min 19s
Wall time: 8min 32s
```

- The code generates summaries using BART model for provided text inputs, stores these in a DataFrame, 'Summaries', containing gold and BART-generated summaries, then displays the initial rows of the DataFrame.

```
Summaries = pd.DataFrame(list(zip(test_df['summary'], SystemSummary)), columns =['OldSummary', 'BartSummary'])
Summaries.head()
```

	OldSummary	BartSummary
0	Kim Jong Il took over when his father died of a heart attack in 1994. He will be remembered as one of the world's most repressive leaders. Kim was well regarded within North Korea. He played a poor hand of cards skillfully, analysts say	Kim Jong Il was the first-born son of an iron-fisted dictator. His father, Kim Il Sung, founded North Korea with Soviet backing after World War II. Kim Jong Il's younger brother drowned as a child and his mother died when he was 7 years old.
1	NEW: In midst of controversies, Obama pushes Congress on student loans. Obama administration faces congressional probes of IRS, Benghazi, AP. Republicans say the issues reflect government gone wild. Democrats contend GOP foes politicize the process	NEW: Obama pushes Congress to extend student loan rates that are set to double on July 1. The White House is trying to shift the focus from the controversies that Republicans seek to emphasize. The GOP narrative depicts a government gone wild, with party leaders and strategists accusing the Obama administration of routinely abusing its powers.
2	NEW: It's unclear how much of a role Russia played, U.S. intelligence says. The European Union agrees on a plan for new sanctions to include Russia. Ukraine opens criminal cases against Russian defense minister and businessman. Dutch official: At least 200 bodies were on the train; more could be at crash site	NEW: A top Ukrainian official says he has no doubt that a Russian fired the missile. NEW: The person who shot down the flight was absolutely a Russian, the official says. Dutch PM: Identifying bodies could take months. A train carrying the remains of crash victims arrives in Kharkiv, Ukraine.
3	NEW: Disaster management agency says 2,487 people have been injured. 1,774 people are confirmed dead from Haiyan. Another 14 dead in Vietnam and five in China, those governments say. International relief heads for stricken islands, but roads a problem	NEW: A new tropical depression dumps heavy rain on the area. The government's confirmed death toll was 1,774 early Tuesday. The storm has injured 2,487 more, and displaced 660,000 people from their homes.
4	Shinseki rescinds 2013 bonus for Phoenix VA director. President Obama defends VA Secretary Shinseki, promises accountability. The number of VA facilities under investigation has expanded to 26. NEW: House panel wants three VA officials to appear at a meeting on Thursday	NEW: Shinseki rescinds $8,495 bonus awarded to Phoenix VA Director Sharon Helman. President Obama says he needs more time to review what's going on at the VA. The number of VA facilities under investigation has expanded to 26, the agency says. CNN first reported the problem last November in a detailed investigation.

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** is used in natural language processing to assess the quality of generated text against reference (old) summaries. It measures the similarity between the model-generated summaries and human-written references, providing a standardized way to evaluate the effectiveness and accuracy of text summarization models.

- The code uses Rouge to evaluate BART model-generated summaries against gold standards, computing recall, precision, and f-measure. Results are stored in a DataFrame, 'BartRouge,' summarizing evaluation metrics for the model's performance compared to the reference summaries.

```
[ ] rouge = Rouge()
    score = rouge.get_scores(Summaries['BartSummary'], Summaries['OldSummary'], avg=True)
    BartRouge = pd.DataFrame(score).set_index([['recall','precision','f-measure']])
    BartRouge
```

	rouge-1	rouge-2	rouge-l
recall	0.292441	0.078604	0.272207
precision	0.281168	0.073370	0.262821
f-measure	0.283620	0.074568	0.264585

6. Title Generation

After summarisation we come to our next part of the project **TITLE GENERATION**.Here we are trying to generate title for a particular summary as we had faced challenges while plugging the entire set of data into the model.

6.1 Pre-Processing:

Before Title Generation ,we perform pre-processing on the summarized data in order to get rid of unwanted words such as stopwords , special characters etc.

Preprocessing

Expanding Contractions

Eg: he's -> he is

```
[ ]
    !pip install contractions
    import contractions
    from nltk.tokenize import word_tokenize
    import re
```

```
Requirement already satisfied: contractions in /usr/local/lib/python3.10/dist-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in /usr/local/lib/python3.10/dist-packages (from contractions) (0.0.24
Requirement already satisfied: anyascii in /usr/local/lib/python3.10/dist-packages (from textsearch>=0.0.21->contractions
Requirement already satisfied: pyahocorasick in /usr/local/lib/python3.10/dist-packages (from textsearch>=0.0.21->contrac
```

```
def expand_contractions(sentence):
    contractions_expanded = [contractions.fix(word) for word in sentence.split()]
    return ' '.join(contractions_expanded)
```

```
[ ] document = expand_contractions(document)
```

```
[ ] def split_into_sentences(doc):
        return [x for x in doc.split('.') if x != '']
```

```
[ ] def remove_punctuation(tokens):
        return [re.sub(r'[^\w\s]', '', word) for word in tokens]
    def lower_case(tokens):
        return [word.lower() for word in tokens if word != '']
```

```
[ ] def sentences_to_vectors(doc):
    # doc is a list of sentences
    # tokenize every sentence, convert to lower case and remove punctuation
        vectors = [word_tokenize(sentence) for sentence in doc]
        for i in range(len(vectors)):
            vectors[i] = lower_case(remove_punctuation(vectors[i]))
        return vectors
```

```
[ ] import nltk
    nltk.download('punkt')
```

```
    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
    True
```

```
[ ] sentences = split_into_sentences(document)
    sentence_vectors = sentences_to_vectors(sentences)
    preprocessed_sentences = [" ".join(sentence) for sentence in sentence_vectors]
```

6.2 Sentence Clustering :

- Sentence clustering is vital in topic modeling as it groups semantically similar sentences, aiding in identifying coherent topics. By clustering, related sentences are grouped together, facilitating the extraction of underlying themes. This process enhances topic coherence and reduces noise, improving the accuracy of topic modeling algorithms.

- In our project, Sentence Clustering plays a crucial role in organizing and grouping sentences within textual documents based on their semantic similarity or thematic relevance. The primary objective is to partition the sentences into distinct clusters, where sentences within the same cluster share common themes or topics. This process serves multiple purposes, including:

    - Facilitating text summarization by identifying key themes or topics within the document.

- Enabling topic modeling by grouping sentences related to the same latent topic.
- Improving document organization and comprehension by structuring sentences into coherent groups.

Sentence Clustering

Since this is an unsupervised task, we use hierarchical agglomerative clustering. Tf-Idf is used to obtain vectors corresponding to the sentences. The optimal number of clusters is found using hierarchy.linkage() which gives the clusters at every iteration of the hierarchical clustering.

Tf-idf

```
[ ] from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[ ] tfidf = TfidfVectorizer(analyzer='word', min_df=0)
```

```
[ ] tfidf_wm = tfidf.fit_transform(preprocessed_sentences)
    tfidf_tokens = tfidf.get_feature_names_out()
    X = pd.DataFrame(data=tfidf_wm.toarray(), columns=tfidf_tokens)
```

6.2.1 TF-IDF Vectorization:

- To begin with, we preprocess the text data by expanding contractions, splitting sentences, and removing punctuation. Next, we employ TF-IDF vectorization to represent each sentence numerically based on the importance of its terms relative to the entire document collection.

- Example:

  Consider the document comprising sentences related to healthcare:

1. "Regular exercise promotes good health."
2. "Healthy eating habits contribute to overall well-being."
3. "Processed foods are detrimental to health."

Following TF-IDF vectorization, each sentence is represented as a numerical vector reflecting the importance of its terms within the document collection.

6.2.2 Hierarchical Clustering:

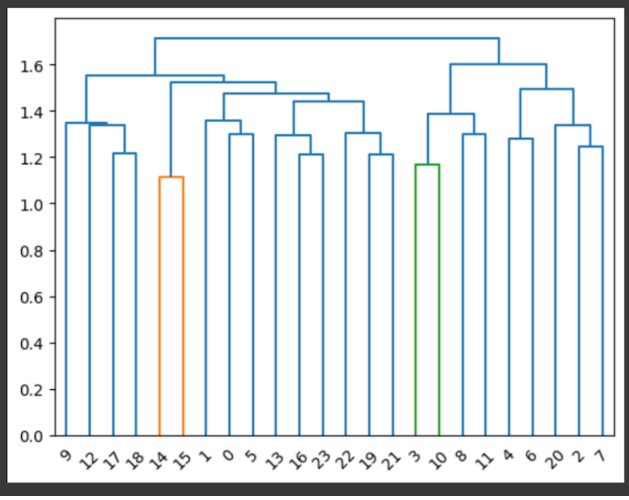
- With the TF-IDF representations in hand, we then apply hierarchical clustering to group similar sentences together. This clustering approach allows us to iteratively merge clusters based on their similarity until all sentences belong to a single cohesive cluster.
- Example:
    - Using hierarchical clustering, above sentences 1 and 2 might be grouped together in one cluster due to their similar content promoting healthy habits, while sentence 3 forms a separate cluster emphasizing the negative impact of processed foods on health.
- Conclusion:
    - By leveraging TF-IDF vectorization and hierarchical clustering techniques, our project effectively organizes and interprets the textual data, enabling us to extract meaningful insights and identify common themes or topics within the documents. This facilitates further analysis and decision-making processes based on the clustered information.

Hierarchical clustering

```
[ ] from scipy.cluster import hierarchy
    from sklearn.cluster import AgglomerativeClustering
```

```
[ ] Z = hierarchy.linkage(X, 'ward')
```

```
[ ] dend = hierarchy.dendrogram(Z)
```



Agglomerative Clustering:

- Agglomerative Clustering is a hierarchical clustering algorithm used for grouping data points into clusters. It starts with each data point considered as a single cluster and then iteratively merges the closest pairs of clusters until only a specified number of clusters remain. The algorithm proceeds by continuously joining clusters based on a linkage criterion, such as "ward" linkage, which minimizes the variance of merged clusters.
- Let's delve deeper into the explanation of the Agglomerative Approach:

1. Initialization:

- At the beginning of the Agglomerative Approach, each data point is considered its own cluster. This means that initially, there are as many clusters as there are data points.

2. Pairwise Distance Calculation:

- The next step involves calculating the pairwise distances between all clusters. This can be done using various distance metrics, such as Euclidean distance or cosine similarity, depending on the nature of the data and the problem at hand.

3. Merging of Closest Clusters:

- Once the pairwise distances are calculated, the algorithm identifies the two closest clusters based on the chosen distance metric. These two clusters are then merged together to form a larger cluster.

4. Updating Distance Matrix:

- After merging the two closest clusters, the pairwise distances between the newly formed cluster and all other clusters need to be updated. This involves recalculating the distances between the new cluster and all remaining clusters.

5. Iterative Process:

- Steps 3 and 4 are repeated iteratively until only one cluster remains, encompassing all data points. This iterative process continues until a stopping criterion is met, such as reaching a predetermined number of clusters or a specific threshold distance.

6. Dendrogram Visualization:

- Throughout the Agglomerative Approach, a dendrogram is constructed to visualize the clustering process. A dendrogram is a tree-like diagram that illustrates the hierarchical relationships between clusters. The x-axis represents individual data points or clusters, while the y-axis represents the distance or dissimilarity between them.

This visualization aids in understanding the clustering structure and determining the optimal number of clusters.

```
[ ] def find_num_clusters(Z):
        # sort by distance
        Z = Z[Z[:, 2].argsort()]
        max_diff = 0
        max_index = 0
        for i in range(len(Z)-1):
            if Z[i+1][2]-Z[i][2] > max_diff:
                max_diff = Z[i+1][2]-Z[i][2]
                max_index = i
        return len(Z)-max_index
```

```
[ ]
    num_clusters = find_num_clusters(Z)
```

```
    hc = AgglomerativeClustering(n_clusters=num_clusters, affinity='euclidean', linkage='ward')
```

```
[ ] y_hc = hc.fit_predict(X)
```

```
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_agglomerative.py:983: FutureWarning:

    Attribute `affinity` was deprecated in version 1.2 and will be removed in 1.4. Use `metric` instead
```

```
[ ] y_hc
```

```
    array([1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
           1, 1])
```

```
[ ] clustered_sentences = [[] for i in range(num_clusters)]
```

```
[ ] for i, sentence in enumerate(preprocessed_sentences):
        cluster_id = y_hc[i]
        clustered_sentences[cluster_id].append(sentences[i])
```

```
[ ] for i, sentence in enumerate(preprocessed_sentences):
        cluster_id = y_hc[i]
        clustered_sentences[cluster_id].append(sentences[i])
```

```
    clustered_sentences

    [[' It cannot be stressed enough that health is the primary thing after which everything else follows',
      ' When you maintain good health, everything else falls into place',
      'Similarly, maintaining good health is dependent on a lot of factors',
      ' Health has a lot of components that carry equal importance',
      ' If even one of them is missing, a person cannot be completely healthy',
      ' First, we have our physical health',
      ' When you have good physical health, you will have a longer life span',
      ' One may maintain their physical health by having a balanced diet',
      ' It makes able a person to gain excessive weight which is called as obesity',
      ' It cannot be stressed enough that health is the primary thing after which everything else follows',
      ' When you maintain good health, everything else falls into place',
      'Similarly, maintaining good health is dependent on a lot of factors',
      ' Health has a lot of components that carry equal importance',
      ' If even one of them is missing, a person cannot be completely healthy',
      ' First, we have our physical health',
      ' When you have good physical health, you will have a longer life span',
      ' One may maintain their physical health by having a balanced diet',
      ' It makes able a person to gain excessive weight which is called as obesity'],
     ['Health was earlier said to be the ability of the body functioning well',
      ' However, as time evolved, the definition of health also evolved',
```

6.3 Topic Modeling

In this part of the project topic we are extracting words that frequently occur in the summary and generate heading out of it.

LDA:Latent Dirichlet Allocation (LDA)

- **<u>How LDA Works:</u>**

1. **Assumptions:**
   - LDA assumes that each document in the corpus is a mixture of various topics, and each word in the document is attributable to one of the document's topics.
   - It further assumes that topics themselves are probability distributions over words, meaning each topic consists of a set of words that occur together frequently.

2. **Model Structure:**
   - LDA represents documents as distributions over topics and topics as distributions over words.
   - It defines two probability distributions:
     - The distribution of words in documents given the topics.
     - The distribution of topics in documents.

3. **Process:**
    - LDA iterates over each document in the corpus and:
        - Assigns each word in the document to a topic randomly.
        - Adjusts these assignments based on the observed words in the document and the topics assigned to them.
    - After multiple iterations, LDA converges to a stable assignment of words to topics.

4. **Inference:**
    - Once the model is trained, LDA allows for inference about the topics present in each document and the distribution of topics across the entire corpus.
    - It assigns a probability score to each word for each topic, indicating the likelihood of that word belonging to the topic.

6.4.1 Topic modeling using LDA :

Here we extract the first topic for a given document. It enables us to identify key themes or topics present in the document's content.
Why We Used LDA in Our Project:

1. **<u>Topic Identification:</u>**
    - In our project, we used LDA to identify topics or themes present in a collection of documents or sentences.
    - By analyzing the distribution of words across topics, LDA helps us identify coherent themes that emerge from the text data.

2. **<u>Unsupervised Learning:</u>**

- LDA is an unsupervised learning technique, meaning it does not require labeled data.
- This makes it suitable for exploratory analysis of text data where the underlying topics or themes are unknown.

3. **Interpretability:**
   - LDA produces interpretable results by assigning probabilities to each word for each topic.
   - This allows us to understand the main topics within a document or corpus and the words associated with each topic.

4. **Dimensionality Reduction:**
   - LDA helps in reducing the dimensionality of the data by representing documents as distributions over a smaller set of topics.
   - This enables us to summarize large volumes of text data into a smaller number of meaningful topics, facilitating easier analysis and interpretation.

How LDA Helps Us:

1. **Topic Summarization:**
   - LDA helps us summarize the main themes or topics present in a document or collection of documents.
   - This facilitates understanding and interpretation of the text data by providing a concise representation of its content.

2. **Content Exploration:**
   - LDA enables us to explore the content of documents by identifying the underlying topics or themes.

- This helps in discovering patterns, trends, and relationships within the text data.

3. **Information Retrieval:**
   - LDA aids in information retrieval by allowing users to search for documents or sentences based on their relevance to specific topics.
   - This improves the efficiency of searching and accessing relevant information within large text corpora.

Therefore, LDA is a powerful tool in our project for uncovering latent topics within text data, facilitating analysis, interpretation, and retrieval of meaningful information.

Implementation:

1. Clean for Topic Modeling:

- This function takes a list of documents (docs) as input.
- It removes stopwords from each document using the remove_stopwords function.
- Stopwords are common words (e.g., "the", "and", "is") that often do not contribute much to the meaning of the text and are typically removed to focus on more meaningful words.

Title Generation/ Topic Modeling

```
[ ] import nltk
    from nltk.corpus import stopwords
    import gensim
    from gensim import corpora
```

```
[ ]
    nltk.download('stopwords')
```

```
    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Package stopwords is already up-to-date!
    True
```

```
[ ]
    stopwords = set(stopwords.words('english'))
    def remove_stopwords(tokens):
        return [word for word in tokens if word not in stopwords and word]
```

```
[ ]
    def clean_for_topic_modeling(docs):
    #
        return [remove_stopwords(doc) for doc in docs]
```

2. Get Topics:

- This function takes preprocessed sentence tokens (sentence_tokens) as input along with optional parameters for the number of topics (num_topics) and the number of words per topic (num_words).
- It creates a dictionary of words from the sentence tokens using corpora.Dictionary.
- It converts the sentence tokens into a document-term matrix using bag-of-words representation (doc2bow).
- It initializes an LDA (Latent Dirichlet Allocation) model using gensim.models.ldamodel.LdaModel.
- The LDA model is trained on the document-term matrix with the specified number of topics and passes.
- Finally, it returns the top words for each topic using print_topics.

```
[ ] def get_topics(sentence_tokens, num_topics=1, num_words=2):
        dictionary = corpora.Dictionary(sentence_tokens)
        doc_term_matrix = [dictionary.doc2bow(doc) for doc in sentence_tokens]
        Lda = gensim.models.ldamodel.LdaModel
        ldamodel = Lda(doc_term_matrix, num_topics, id2word=dictionary, passes=30)
        return ldamodel.print_topics(num_topics, num_words)
```

3. Topic for Paragraph:

- This function takes a document (doc) as input along with optional parameters for the number of topics (num_topics) and the number of words per topic (num_words).
- It splits the document into sentences using split_into_sentences.
- It converts the sentences into vectors using sentences_to_vectors.
- It preprocesses the sentence vectors by removing stopwords using clean_for_topic_modeling.
- Finally, it retrieves the topics for the preprocessed sentences using get_topics and returns them.

```
def topic_for_para(doc, num_topics=1, num_words=2):
    sentences = split_into_sentences(doc)
    sentence_vectors = sentences_to_vectors(sentences)
    sentence_tokens = clean_for_topic_modeling(sentence_vectors)
    return get_topics(sentence_tokens, num_topics, num_words)
```

```
[ ]
topic_for_para(document, num_topics=1, num_words=3)

[(0, '0.031*"health" + 0.024*"foods" + 0.021*"junk"')]
```

4. Get First Topic:

- This function takes the topics generated by get_topics as input.
- It extracts the first topic from the list of topics by finding the indices of quotation marks (") in the topic string.
- It then extracts the text between each pair of quotation marks, representing individual words or phrases in the topic.
- Finally, it returns the extracted topics as a list.

```
[ ] import re
    def get_first_topic(topics):
        indexes = [m.start() for m in re.finditer('\"', topics[0][1])]
        extracted_topics = []
        for i in range(0,len(indexes)-1,2):
            extracted_topics.append(topics[0][1][indexes[i]+1:indexes[i+1]])
        return extracted_topics
```

5. Execution:

- These lines of code execute the topic_for_para function to obtain topics for the document.
- Then, they pass the generated topics to the get_first_topic function to extract the first topic.
- The extracted topic is assigned to the variable h1_heading.

```
[ ] topics = topic_for_para(document, 1, 3)
    topics

    [(0, '0.031*"health" + 0.024*"foods" + 0.021*"junk"')]
```

```
[ ] h1_heading = get_first_topic(topics)
    h1_heading

    ['health', 'foods', 'junk']
```

6.5 Tuning BART and using Seq2SeqBatchTokenizer

We are tuning the BART model with a different tokenizer named Seq2SeqBatchTokenizer this is helping us in creating summarize text and producing headings and subheadings:

```
[ ] paragraphs = [{'text': ". ".join(i)} for i in clustered_sentences]
```

```
[ ] pretrained_model_name = "facebook/bart-large-cnn"
    hf_arch, hf_config, hf_tokenizer, hf_model = get_hf_objects(pretrained_model_name,
                                                                model_cls=BartForConditionalGeneration)

    hf_arch, type(hf_config), type(hf_tokenizer), type(hf_model)

    ('bart',
     transformers.models.bart.configuration_bart.BartConfig,
     transformers.models.bart.tokenization_bart_fast.BartTokenizerFast,
     transformers.models.bart.modeling_bart.BartForConditionalGeneration)
```

```
[ ] hf_batch_tfm = Seq2SeqBatchTokenizeTransform(hf_arch, hf_config, hf_tokenizer, hf_model,is_split_into_words=True, task='
    text_gen_kwargs={'max_length': 248,
     'min_length': 56,
     'do_sample': False,
     'early_stopping': True,
     'num_beams': 4,
     'temperature': 1.0,
     'top_k': 50,
     'top_p': 1.0,
     'repetition_penalty': 1.0,
     'bad_words_ids': None,
     'bos_token_id': 0,
     'pad_token_id': 1,
     'eos_token_id': 2,
     'length_penalty': 2.0,
     'no_repeat_ngram_size': 3,
     'encoder_no_repeat_ngram_size': 0,
     'num_return_sequences': 1,
     'decoder_start_token_id': 2,
     'use_cache': True,
     'num_beam_groups': 1,
     'diversity_penalty': 0.0,
     'output_attentions': False,
     'output_hidden_states': False,
     'output_scores': False,
     'return_dict_in_generate': False,
     'forced_bos_token_id': 0,
     'forced_eos_token_id': 2,
     'remove_invalid_values': False})
```

```
[ ] blocks = (Seq2SeqTextBlock(hf_arch, hf_config, hf_tokenizer, hf_model, task='summarization'), noop)

    dblock = DataBlock(blocks=blocks, get_x=ColReader('text'), get_y=ColReader('summary'), splitter=RandomSplitter())
```

```
[ ] seq2seq_metrics = {
            'rouge': {
                'compute_kwargs': { 'rouge_types': ["rouge1", "rouge2", "rougeL"], 'use_stemmer': True },
                'returns': ["rouge1", "rouge2", "rougeL"]
            },
            'bertscore': {
                'compute_kwargs': { 'lang': 'fr' },
                'returns': ["precision", "recall", "f1"]
            }
        }
```

```
[ ] dls = dblock.dataloaders(train_df, bs=2)

    /usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:3921: UserWarning:

    `as_target_tokenizer` is deprecated and will be removed in v5 of Transformers. You can tokenize your labels by using the
```

```
[ ] model = BaseModelWrapper(hf_model)
    learn_cbs = [BaseModelCallback]
    fit_cbs = [Seq2SeqMetricsCallback(custom_metrics=seq2seq_metrics)]

    learn = Learner(dls,
                    model,
                    opt_func=ranger,
                    loss_func=CrossEntropyLossFlat(),
                    cbs=learn_cbs,
                    splitter=partial(blurr_seq2seq_splitter, arch=hf_arch)).to_fp16()

    learn.create_opt()
    learn.freeze()

    /usr/local/lib/python3.10/dist-packages/datasets/load.py:759: FutureWarning:

    The repository for rouge contains custom code which must be executed to correctly load the metric. You can inspect the re
    You can avoid this message in future by passing the argument `trust_remote_code=True`.
    Passing `trust_remote_code=True` will be mandatory to load this metric from the next major release of `datasets`.

    Downloading builder script: [progress bar] 5.65k/? [00:00<00:00, 278kB/s]
    /usr/local/lib/python3.10/dist-packages/datasets/load.py:759: FutureWarning:

    The repository for bertscore contains custom code which must be executed to correctly load the metric. You can inspect th
    You can avoid this message in future by passing the argument `trust_remote_code=True`.
    Passing `trust_remote_code=True` will be mandatory to load this metric from the next major release of `datasets`.

    Downloading builder script: [progress bar] 8.10k/? [00:00<00:00, 442kB/s]
```

7. CONCLUSION

As we are concluding this project,we have been able to summarize the articles into short paragraphs and generated hierarchy of headings by extracting frequently generated words from the clusters of similar sentences.

1. Summarizing the Document:

- This function takes a document (doc) as input and generates a summary using the fine-tuned BART model (learn).
- It uses the blurr_generate method to generate the summary.
- The early_stopping=False parameter ensures that the model generates the entire summary without early stopping.
- The num_return_sequences=1 parameter specifies that only one summary should be generated.

- The summarize function encapsulates the process of generating summaries for a collection of text data.
- By leveraging the progress_apply method from tqdm, it provides a visual progress indicator to monitor the summarization process.
- This function facilitates batch summarization of text data, making it efficient to process large volumes of text and generate summaries for each item in the input dataset.

```
def summarize_doc(doc):
    return learn.blurr_generate(doc, early_stopping=False, num_return_sequences=1)[0]['generated_texts']
```

Steps:

1. **Progress Bar Setup:**
   - It sets up a progress bar using tqdm to track the progress of the summarization process. The description of the progress bar is set to “Summarization".

2. **Summarization Process:**
   - It applies the summarize_doc function to each item in the DataFrame or Series using data.progress_apply().
   - For each text item (x), the summarize_doc function is called to generate a summary.

```
[ ] def summarize(data):
    #     Takes a pd data series as input
        tqdm.pandas(desc='Summarization')
        return data.progress_apply(lambda x: summarize_doc(x))
```

3. **Return:**

   - The function returns a Pandas Series or DataFrame containing the generated summaries for each item in the input data.
   - This loop iterates over each paragraph in the paragraphs list.
   - For each paragraph, it generates a summary using the summarize_doc function and stores it in the 'summary' key of the paragraph dictionary.
   - It then extracts topics from the generated summary using the topic_for_para function and stores the first topic as the 'h2_heading' key of the paragraph dictionary.

```python
for paragraph in paragraphs:
    paragraph['summary'] = summarize_doc(paragraph['text'])
    topics = topic_for_para(paragraph['summary'], 1, 3)
    print(get_first_topic(topics))
    paragraph['h2_heading'] = get_first_topic(topics)
```

```
['health', 'person', 'one']
['high', 'health', 'time']
```

4. Writing to Markdown File:

```python
output_file = open("./output.md", "w")
```

```python
def write_h1(f, h1_heading):
    f.write("# ")
    for h in h1_heading:
        f.write(h + " | ")
    f.write("\n")
```

```python
def write_sections(f, paragraphs):
    for para in paragraphs:
        f.write("## ")
        for h in para['h2_heading']:
            f.write(h + " | ")
        f.write('\n')
        f.write(para['summary'])
        f.write('\n')
```

```python
write_h1(output_file, h1_heading)
write_sections(output_file, paragraphs)
```

```python
output_file.close()
```

```python
output_file = open("./output.md", "r")
```

```python
print(output_file.read())
```

```
# health | foods | junk |
## health | person | one |
Health has a lot of components that carry equal importance. If even one of them is missing, a person cannot be completely
## high | health | time |
Health was earlier said to be the ability of the body functioning well. However, as time evolved, the definition of healt
```

7.Challanges Faced

1. We encountered difficulties when attempting to summarize our test dataset using the BART+Seq2Seq tokenizer. The issue arose when trying to generate multiple summaries within the dataframe. Although we could not identify the problem in the code, the Rouge score indicated satisfactory performance. Consequently, we resorted to using it for a particular text and generated a summary. For the remaining dataset, we opted to utilize the Agglomerative tokenizer instead.

2. In our project, we faced memory constraints, which limited our ability to use a large dataset. As a result, we had to work with a smaller dataset to generate title hierarchies.
While this approach allowed us to work within our memory limitations, it's important to acknowledge that the results may be influenced by the reduced scope of the dataset.

7.Collab Notebook

https://colab.research.google.com/drive/1GmdsD8OeV9EnwhOiFCqKDjr9Loz1cxyt?usp=sharing