This project has been carried out under the supervision of Professor Srinath Srinivasa

- Aparajita Kumari(MT2023037)
- Nandini Yadav(MT2023096)

Nested Summarization with Heading hierarchy

Contents

- **Overview**
- **Objectives**
- **Implementation**
    - Lexical Processing
    - Syntactic Processing
    - Semantic Processing
- **Deliverables**
    - Text Summarization
    - Title Generation
- **Challenges Faced**
- **Conclusion**
- **References**
- **Collab Notebook**

Overview

Context

- Nowadays people are consumers of short type of content.
- Properly structured headings improve accessibility by providing clear navigation cues.

Problem Statement

- Develop algorithms to analyze the content structure and identify hierarchical relationships between different sections and subsections.
- Develop algorithms to generate concise summaries that capture the main points and essential information by maintaining sense and relevance.

Objectives

Collecting Dataset

Data collection and preparation of the dataset to contain the relevant features appropriately.

Preprocessing

Remove any irrelevant metadata or columns that are not needed for analysis.

Remove special characters, punctuation, and non-alphanumeric characters.

Text Summarization

Generate a summary by paraphrasing and synthesizing the main points of the text. This typically involves deep learning models such as sequence-to-sequence models with attention mechanisms e.g. BART

Title Generation

Review the summarized content and identify the main themes, topics, or ideas that are prominently featured. These could be the central concepts or takeaways from the summary.

Intuition

- **Text summarization** will condense key information into succinct summaries, saving time and effort for users. Summaries capture essential points while eliminating redundant details, enhancing readability and comprehension.

- **By combining heading hierarchy with summarization**, users can quickly locate relevant sections using headings and access concise summaries to grasp the main ideas, reducing cognitive load and improving efficiency.

Implementation

- **Lexical Processing**
  - Removing unwanted urls,tags and stopwords using nltk library.
  - We implemented canonicalisation with the help of WordNet Lemmatizer.

- **Syntactic Processing**
  - It performs part-of-speech tagging to identify the grammatical categories of words in the text.
  - Additionally, it uses dependency parsing to analyze the syntactic structure of sentences.

- **Semantic Processing**
  - An important aspect of this processing 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.

Lexical Processing

- Removing unwanted urls,tags and stopwords using nltk library.
- We implemented **canonicalisation** with the help of WordNet Lemmatizer.
- To streamline the preprocessing process, all the aforementioned techniques are integrated into a cohesive pipeline. This pipeline serves as a standardized workflow, ensuring that each document undergoes the same preprocessing steps consistently.

Combining into a pipeline

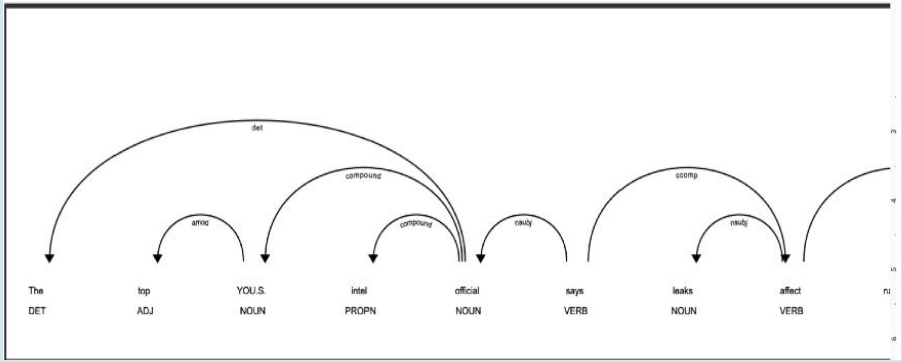
```
[ ] def process(stories):
        stories = remove_url(stories)
        stories = remove_html(stories)
        stories = remove_bracket(stories)
        stories = remove_digit(stories)
        stories = remove_underscore(stories)

        processed_list = []
        for story in stories:
            processed = expand_contractions(story)
            processed = tokenize(processed)
            processed = lower_case(processed)
            processed = remove_punctuation(processed)
            processed = remove_stopwords(processed)
            processed_list.append(processed)

        return lemmatizer(processed_list)
```

Syntactic Processing

- The code utilizes natural language processing (NLP) tools such as NLTK, Stanza, and spaCy for further analysis.
- It performs **part-of-speech tagging** to identify the grammatical categories of words in the text.
- Additionally, it uses **dependency parsing** to analyze the syntactic structure of sentences.
- Finally, it visualizes the **dependency parse tree** using the displaCy library from spaCy
- The code also aims to implement **coreference resolution** to identify and link coreferent mentions in the text, enhancing semantic understanding.

```
tagged = pos_tag(word_tokenize(text))
tagged
```

```
[('Six', 'CD'),
 ('people', 'NNS'),
 ('have', 'VBP'),
 ('been', 'VBN'),
 ('killed', 'VBN'),
 ('in', 'IN'),
 ('three', 'CD'),
 ('incidents', 'NNS'),
 ('over', 'IN'),
 ('the', 'DT'),
 ('past', 'JJ'),
 ('two', 'CD'),
 ('months', 'NNS'),
 ('in', 'IN'),
 ('Honduras', 'NNP'),
 ('.', '.'),
 ('The', 'DT'),
 ('incidents', 'NNS'),
 ('happened', 'VBD'),
 ('during', 'IN'),
 ('the', 'DT'),
 ('course', 'NN'),
```



Semantic Processing

- An important aspect of this 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.
- LDA plays a crucial role in semantic processing by uncovering latent topics within text data, facilitating semantic analysis.

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
```

```python
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)
```

Deliverables

- Text Summarization
- Title Generation

Text Summarization

- 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.

Text Summarization

- 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.

```
%%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 old 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.

Title Generation

- **Preprocessing:**
  - 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 != '']
```

Title Generation

- **Sentence Clustering:**
  - It facilitates text summarization by identifying key themes or topics within the document.
  - It enables topic modeling by grouping sentences related to the same latent topic.
  - It improves document organization and comprehension by structuring sentences into coherent groups.

- **TF-IDF Vectorization:**
  - We employ TF-IDF vectorization to represent each sentence numerically based on the importance of its terms relative to the entire document collection.
  - Following TF-IDF vectorization, each sentence is represented as a numerical vector reflecting the importance of its terms within the document collection.

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)
```

Title Generation

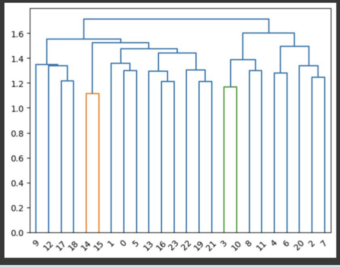
- **Hierarchical Clustering:**
  - With the TF-IDF representations in hand, we then apply hierarchical clustering to group similar sentences together.
  - This clustering approach allows us to iteratively merge clusters based on their similarity until all sentences belong to a single cohesive cluster.
  - **Agglomerative Clustering** is a hierarchical clustering algorithm used for grouping data points into clusters.
  - The algorithm proceeds by continuously joining clusters based on a linkage criterion, such as "ward" linkage, which minimizes the variance of merged clusters.

Hierarchical clustering

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

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

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




```
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',
```

Title Generation

- **Topic Modeling:**
    - Here,we are extracting words that frequently occur in the summary and generate heading out of it.
    - **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.
    - **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.
    - This algorithm assigns a probability score to each word for each topic, indicating the likelihood of that word belonging to the topic.

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]
```

```
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"')]
```

Challenges Faced

- 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.

- 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.

Conclusion

- As we near the conclusion of this project, we have successfully condensed lengthy articles into concise paragraphs through summarization techniques.
- Additionally, we have established a structured hierarchy of headings by analyzing clusters of sentences that share similar content.
- This hierarchical organization is achieved by identifying and extracting frequently occurring words, which serve as the basis for categorizing and structuring the summarized content.

```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']
```

```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 completel
## high | health | time |
Health was earlier said to be the ability of the body functioning well. However, as time evolved, the definition of heal
```

References

- https://github.com/Jay-Suthar/TEXT-SUMMARIZATION-USING-BART-T5-PROPHETNET-PEGASUS/blob/master/group-22-ir-project.ipynb
- Topic modeling using LDA

Collab Notebook

- Our collab notebook:.
- https://colab.research.google.com/drive/1GmdsD8OeV9EnwhOiFCqKDjr9Loz1cxyt?usp=sharing

Thank You