DSCI 6004-02
Natural Language Processing

By:Mummadi Roshan Reddy

News Summarization using transformers

Statement of Project Objectives

Develop a system for automatic summarization that can be used in real-world settings like news aggregation platforms, offering users quick, valuable insights. The system should generate brief overviews of news articles by identifying key points, significant events, and notable quotes. Implement and compare both extractive and abstractive summarization techniques to assess their effectiveness in producing coherent and informative summaries. Ensure the algorithms and models are capable of processing high volumes of articles in real-time to deliver timely summaries. Integrate user feedback and preferences into the process to customize summaries based on individual interests and reading patterns.

Statement of value – why is this project worth doing?

- This project is important for several reasons. Firstly, with the abundance of news articles available online, users may become overwhelmed. News summarization helps by condensing lengthy articles into shorter summaries, allowing users to access information quickly and efficiently. Since not everyone has the time or desire to read full articles, summaries provide an easy way to stay informed about current events without having to go through large amounts of text.

- Secondly, concise and clear summaries can enhance user engagement with news content. When users can quickly access the main points, they are more likely to stay informed and interested.

- Finally, summaries can support decision-making by delivering essential information in a compact form. Whether it's keeping up with current news or making informed decisions based on news analysis, summaries prove to be useful tools.

APPROACH

- The research centered on utilizing Neural Networks, specifically the Transformer model. The scaled_dot_product_attention function calculates attention weights by taking the dot product of query and key vectors, scaling the result, applying a mask if needed, and normalizing with the SoftMax function.

- The MultiHeadAttention class splits query, key, and value vectors into multiple heads for independent attention processing, then merges the outputs to capture different input features simultaneously. The EncoderLayer consists of a multi-head self-attention mechanism and a feed-forward neural network, while the DecoderLayer includes two multi-head attention mechanisms: one for encoder output and one for self-attention.

- The Encoder and Decoder classes are built with multiple respective layers. The encoder processes the input sequence, and the decoder generates the output using the encoder's data. Positional encoding is added to provide positional context. The Transformer class combines these components to create the model, processing input and target sequences with masking and padding to produce the final output.

Here is the Flowchart

Data Collection	*Model Selection*	*Fine-Tuning*	*Inference*	*Evaluation*
Step-1	***Step-2***	***Step-3***	***Step-4***	***Step-5***

Deliverables

- Jupyter Notebook: This notebook contains a prototype implementation for summarizing news articles. It illustrates how the system works, covering aspects like data collection, processing, and generating summaries.

- Performance Evaluation Results: This report presents an evaluation of the prototype’s performance and accuracy. It highlights how well the system meets its goals and identifies areas that could be improved upon in future iterations.

- Project Presentation: This presentation provides a summary of the project’s aims, methods, findings, and suggestions. It is designed to convey the project's importance and relevance to stakeholders, including website managers and users.

Evaluation methodology

- Text Preprocessing: The input document is tokenized using the document_tokenizer and padded to a fixed length (encoder_maxlen) for encoder input.
- Encoder Input: The processed document is expanded to include a batch dimension and fed into the Transformer model's encoder, generating output representations of the document.
- Decoder Input Initialization: The decoder is initialized with a start-of-sequence token ("<go>") to indicate the beginning of the summary generation.
- Summary Generation: The decoder predicts one word at a time, using the current sequence (starting with the "<go>" token and previously generated words) to generate the next word. The word with the highest predicted probability is selected and added to the summary. This continues until the maximum summary length (decoder_maxlen) is reached or the stop token ("<stop>") is produced.
- Output: The generated summary is returned by the evaluate function.
- Summarize Function: The summarize function calls evaluate to generate the summary and converts the tokenized output back into text with the summary_tokenizer.

Thank you!