End-to-end pipeline of the Research Paper Simplifier

Team:

Dreamy Pujara - 202211005
Aayushi Patel - 202211025
Madhavi Aghera - 202211049
Misha Patel - 202211060

Preprocessing Module

For Decoder (Output) :

GPT-2 is a variant of the GPT (Generative Pre-trained Transformer) model, and it employs tokenization as a critical step in processing and understanding text. Here's how the GPT-2 Tokenizer typically works:

1. Text Input: You start with a piece of text that you want to process or feed into the GPT-2 model.

2. Tokenization: The GPT-2 Tokenizer breaks down the input text into smaller units called tokens.

3. Special Tokens: Like in other tokenizers for transformer models, the GPT-2 Tokenizer often adds special tokens to the list, including a "start of text" token, an "end of text" token, and a "padding" token. These special tokens help the model understand the structure of the text and improve text generation.

4. Vocabulary: GPT-2 models have a fixed vocabulary or token set that they understand. The tokenizer maps the text tokens to corresponding IDs from this vocabulary. For instance, the word "cat" might be tokenized into an ID like 231.

5. Variable Token Length: Tokens can have varying lengths. A short word might be represented by a single token, while a longer word or phrase could be split into multiple tokens. The tokenizer keeps track of the token length information.

6. Encoding: The GPT-2 Tokenizer encodes the list of tokens into a format that can be used as input for the GPT-2 model. This format is usually a numerical array or tensor of token IDs, where each ID corresponds to a token in the vocabulary.

7. Padding: If the input text is shorter than a predefined maximum sequence length, the tokenizer pads it with the "padding" token to match the model's input size. Padding ensures that all input sequences have the same length.

8. Output: The output of the GPT-2 Tokenizer is the encoded input, which can be fed into the GPT-2 model for various natural language understanding tasks, such as text generation, text completion, question-answering, and more.

The specifics of tokenization may differ depending on the implementation and the specific variant of the GPT-2 model you are using. GPT-2 models often use subword tokenization methods (such as Byte-Pair Encoding or SentencePiece) to handle out-of-vocabulary words and reduce the size of the vocabulary. However, the fundamental process I described above is common to most text processing tasks with GPT-2 models.

For Encoder (Input) :

The BERT (Bidirectional Encoder Representations from Transformers) Tokenizer is an essential component for processing text data before feeding it into BERT-based models. Here's how a typical BERT Tokenizer works:

1. Text Input: You start with a piece of text that you want to process or use as input for a BERT-based model. This text can be a sentence, paragraph, or any text input.

2. Tokenization: The BERT Tokenizer breaks down the input text into smaller units called tokens. Tokens can be words, subwords (subword pieces like "un-" and "happiness"), or even individual characters. BERT tokenization differs from traditional tokenization in that it uses a subword-level tokenization method, which means that words can be split into multiple subword tokens.

3. Special Tokens: The tokenizer adds special tokens to the list of tokens, including a "CLS" (classification) token, a "SEP" (separator) token, and potentially a "MASK" token. These special tokens are important for various BERT-related tasks.

- "CLS" Token: This token is typically added at the beginning of the input text and is used for classification tasks, where BERT predicts a label or class for the input.
- "SEP" Token: This token is used to separate different segments of text, especially in tasks involving multiple sentences or segments.

- "MASK" Token: In pre-training BERT, some tokens are randomly replaced with the "MASK" token, and the model learns to predict the original tokens. However, this token is not always used during fine-tuning for specific tasks.

4. Vocabulary: BERT models have a fixed vocabulary or token set that they understand. The tokenizer maps the text tokens to corresponding IDs from this vocabulary. For example, the subword "unhappiness" might be tokenized into a sequence of subword tokens and their corresponding IDs.

5. Variable Token Length: Tokens can have varying lengths. A short word might be represented by a single token, while a longer word could be split into multiple subword tokens. The tokenizer keeps track of the token length information.

6. Encoding: The BERT Tokenizer encodes the list of tokens into a numerical format that can be used as input for the BERT model. Typically, the encoded input is a numerical array or tensor of token IDs, where each ID corresponds to a token in the vocabulary.

7. Special Tokens in Encoding: The encoded sequence includes the special tokens like "CLS" and "SEP" in their respective positions, which allows the model to understand the structure of the input.

8. Padding and Truncation: BERT models have a fixed maximum sequence length. If the input text is longer than this maximum length, it is truncated. If it's shorter, padding is added to match the maximum length.

9. Output: The output of the BERT Tokenizer is the encoded input, which can be fed into BERT-based models for various natural language processing tasks, including text classification, question-answering, and text representation tasks.

The specific tokenization and encoding details may vary depending on the BERT model variant and the implementation you are using. Subword tokenization methods like Byte-Pair Encoding (BPE) or SentencePiece are often used to handle subword-level tokenization in BERT models. However, the fundamental process described above is common to most BERT-based models.

Definition of simplification

The project focuses on the simplification of English sentences. When provided with an English sentence as input, the model's goal is to restructure or substitute words and phrases in order to

enhance comprehension without compromising the essential information conveyed by the original sentence. This initiative aims to develop an effective approach for achieving favorable results in the task of sentence simplification, utilizing a word-based modeling approach.

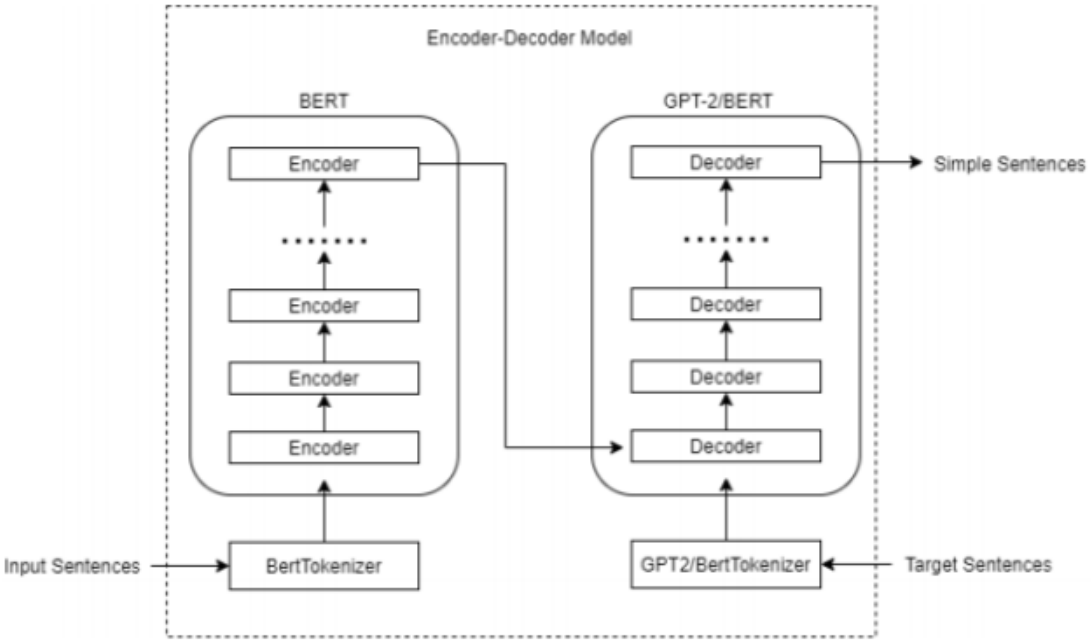
Scope of simplification

The project's results have the potential to benefit various NLP tasks that demand simplified sentences, including applications like Machine Translation, Summarization, and Classification. This project employs a combination of the Bert model for encoding and the GPT-2 model for decoding, allowing for versatile applications in the NLP domain.

Approach

This project offers a comprehensive pipeline designed for simplification tasks, implementing a supervised approach that leverages state-of-the-art transformer models. The core functionality involves accepting regular sentences as input, which are subsequently transformed into token embeddings via the utilization of the BERTTokenizer. These token embeddings are then passed through an encoder-decoder model. As the final step in the process, the model generates token embeddings for simplified sentences, which are ultimately transformed into coherent sentences using the GPT2Tokenizer.

Loss Function : Log Softmax
Evaluation Matrix : BLEU score



Timeline for completion of Project

Data Preprocessing	Week 1
Model Training and Optimization	Week 2
Evaluation and Testing	Week 3
Final Refinements	Week 4