Assignment: Medical Keyword Extraction

NAME : VARUN VASHISHTHA

COLLEGE : IIIT HYDERABAD

The task at hand involves the creation of a program dedicated to extracting keywords from medical transcriptions. These extracted keywords play a pivotal role in summarizing content, enhancing search capabilities, and facilitating efficient information retrieval within the medical domain. The development and testing phases will utilize an extensive collection of transcribed medical reports spanning various specialties.

Keyword extraction serves a crucial purpose by distilling essential terms or phrases from intricate medical transcriptions. This functionality proves invaluable for quickly understanding key points within medical content, streamlining the search process, and assisting practitioners in promptly accessing pertinent information. The approach centers around harnessing Natural Language Processing (NLP) techniques.

PREPROCESSING STEP

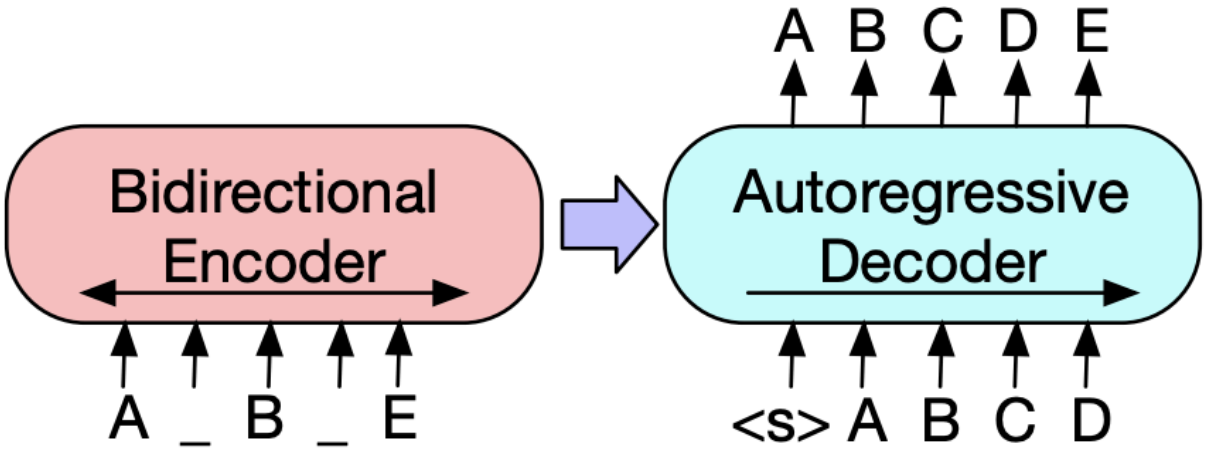
- Data Filtering:
  - The DataFrame dataframe undergoes filtration to exclude rows with missing values (NaN) in either the 'transcription' or 'keywords' column.
- Text Tokenization:
  - A tokenization function named tokenize is introduced, likely responsible for breaking down the text into individual words or tokens. This step is foundational for subsequent text processing.
- Text Lowercasing:
  - Another function, get_lower, is established to convert all text to lowercase. This normalization step ensures uniformity and consistency in the textual data.
- Punctuation Removal:
  - Although the code for removing punctuation is present but commented out, indicating that this step could be part of the pipeline. The removal of punctuation can be advantageous for text analysis, particularly if the focus is on individual words.

FINE-TUNING BART FOR KEYWORD EXTRACTION

BART (Bidirectional and Auto-Regressive Transformers), a transformer-based neural network architecture developed by Facebook AI, is specifically designed for sequence-to-sequence tasks.

Widely recognized for its proficiency in natural language processing, BART excels in applications like text summarization and language generation. Its unique feature involves utilizing a denoising autoencoder objective during pre-training, where the model learns to reconstruct a corrupted input sequence. BART has consistently demonstrated cutting-edge performance in various language generation tasks, making it an influential model for tasks that involve sequence transduction and generation.

The fine-tuning process for BART in keyword extraction involves tailoring the pre-trained BART model to focus on identifying and extracting keywords from text data. This adaptation includes customizing the model's training using a dataset that comprises examples of medical transcriptions and their corresponding keywords. Through this fine-tuning on task-specific data, BART becomes adept at generating precise and pertinent keywords, offering a specialized solution for keyword extraction within the medical domain. Once fine-tuned, the model can be applied to new medical transcriptions to automatically extract crucial terms, thereby enhancing content summarization and facilitating efficient information retrieval.




```
tokenizer = AutoTokenizer.from_pretrained("facebook/bart-base", padding_side="left",
                                          truncation_side='right')
model = BartForConditionalGeneration.from_pretrained("facebook/bart-base").to("cuda")
```

- Printing Metrics:
    - Prints the average loss and accuracy over the training batches.
    - And also printed the Rouge score

- Key Highlights of the Code:

text Function:

- Accepts parameters such as text input, a pre-trained model, and a tokenizer.
- Utilizes the provided tokenizer to tokenize the input text.
- Processes the tokenized input through the generate method of the model, obtaining generated keyword IDs.
- Decodes the generated keyword IDs using the tokenizer, excluding special tokens.
- Returns the decoded keywords.

summary Function:

- Applies the text function to each transcription in the DataFrame using the apply method.
- Introduces a new column 'Result' in the DataFrame to store the extracted keywords.
- Returns the modified DataFrame.

The BART Transformer model demonstrates effectiveness in extracting medical keywords from transcriptions. However, it is crucial to note that AI models may produce unrelated or misleading results, emphasizing the significance of human knowledge and domain understanding in interpreting the outputs.

To enhance the model's performance, expanding the data corpus is recommended. Training the model on a more extensive medical dataset can improve its ability to capture diverse medical terminology and nuances, leading to more accurate keyword extraction.

Furthermore, the flexibility of BART allows for fine-tuning on specific healthcare use cases. By tailoring the model to the intricacies of a particular medical domain through fine-tuning, it can be optimized for tasks such as summarization, information retrieval, and keyword extraction, contributing to its versatility in addressing various healthcare scenarios.

KEYWORD EXTRACTION USING KEYBERT

- Input Text:
  The content designated for keyword extraction originates from the transcription stored in the 10th row of the DataFrame: medical_df.iloc[10]['transcription'].

Keyword Extraction Parameters:

- keyphrase_ngram_range=(1, 2): Defines the range of n-grams considered for keyword extraction, encompassing both unigrams and bigrams.
- stop_words='english': Directs the exclusion of common English stop words during the keyword extraction process.
- use_maxsum=True: Indicates the preference for employing the MaxSum algorithm in keyphrase selection.
- nr_candidates=20: Specifies the quantity of candidate keyphrases to evaluate during the extraction procedure.
- top_n=10: Sets the criterion for retrieving the top 10 keyphrases as the final output.

Key Features of KeyBERT:

BERT Embeddings: KeyBERT utilizes BERT embeddings to represent words and phrases in a continuous vector space, considering contextual information.
Keyword Extraction Methods: It offers various methods for keyword extraction, including single- and multi-document extraction, as well as extraction based on user-provided queries.

The architecture of KeyBERT involves several components that work together to extract meaningful keywords from text using BERT embeddings. Here's a breakdown of its key components:

BERT Model Integration:

- KeyBERT leverages BERT, a transformer-based neural network architecture, for its underlying embeddings. BERT provides contextualized representations of words and phrases, capturing the nuances of language and context in the embedding space.

Embedding Extraction:

- The first step involves tokenizing the input text and obtaining BERT embeddings for each token. These embeddings represent the semantic meaning and context of words in the text.

Clustering (Aggregation) of Embeddings:

- To capture the diversity of the document and identify key themes, KeyBERT employs clustering. This involves grouping similar embeddings together. Users can customize the number of clusters, influencing the granularity of the extracted keywords.

Keyword Selection:

- Within each cluster, the algorithm selects representative keywords. This step involves choosing words that are central to the cluster and contribute significantly to its overall meaning. The selection process is based on the MaxSum algorithm, aiming to maximize the diversity and importance of selected keywords.

User-Defined Queries (Optional):

- KeyBERT allows users to provide queries that guide the keyword extraction process. If specified, the extraction focuses on terms related to the provided queries, enhancing the relevance of the extracted keywords.

Output:

- The final output is a list of keywords extracted from the input text. These keywords encapsulate the essential concepts and themes present in the document, providing a concise summary or representation.

USING COUNT VECTORIZER AND BERT

The goal is to extract keywords from a given input text. Here's a summary of the key steps:

Parameter Setup:

- n_gram_range is set to (1, 1), indicating the use of unigrams.
- stop_words is set to "english" to exclude common English stop words.

Load Sentence Transformer Model:

- The code loads a pre-trained Sentence Transformer model (DistilBERT) for encoding sentences into dense vectors.

Keyword Extraction Function:

- The generate_keywords function takes an input text and an optional parameter for the number of top keywords to generate.

Count Vectorization:

- CountVectorizer is used to tokenize and represent the input text in terms of unigrams.
- Candidate keywords are extracted based on these unigrams.

Encode Input and Candidates:

- The Sentence Transformer model encodes the input text and the candidate keywords into dense vectors.

Calculate Cosine Similarity:

- Cosine similarity is computed between the encoded input and candidate keyword embeddings.

Select Top Keywords:

- The top keywords are selected based on the highest cosine similarity scores.

Return Keywords:

- The function returns the generated keywords.

Simple Token embedding matching using bert:

In this I simply found the Bert embedding of the whole transcript and then for each word in the transcript I found the similarity score of each word bert embedding and whole transcript bert embedding and whoever is more important for the whole sentence will be getting a better cosine score.

So i set a limit lets say 0.6 and whoever is greater than 0.6 will be predicted

ROUGE SCORE:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used for the automatic evaluation of machine-generated text, particularly in the context of text summarization and machine translation. The purpose of ROUGE is to assess the quality of a generated summary or translation by comparing it to one or more reference summaries.

Key components of ROUGE include:

Recall:

- ROUGE primarily focuses on recall, which measures the overlap between the generated text and the reference summaries in terms of n-grams (consecutive sequences of words).
- Higher recall values indicate that the generated text contains more content similar to the reference summaries.

Precision:

- Although recall is emphasized, some ROUGE variants also consider precision. Precision measures how much of the generated content is relevant.
- Precision is particularly relevant in cases where brevity and avoiding unnecessary information are crucial.

F1 Score:

- The F1 score is the harmonic mean of recall and precision. It provides a balanced measure that considers both aspects of content overlap and relevance.
- F1 score is a commonly used metric in machine learning evaluation, striking a balance between precision and recall.

N-grams:

- ROUGE evaluates overlap at different levels of n-grams (unigrams, bigrams, etc.). For example, ROUGE-1 considers the overlap of single words, while ROUGE-2 considers the overlap of pairs of words (bigrams).
- This helps in capturing both local and global content overlap.

Variants:

- Different variants of ROUGE (e.g., ROUGE-N, ROUGE-L, ROUGE-W) emphasize different aspects of evaluation. ROUGE-N focuses on n-gram overlap, ROUGE-L considers the longest common subsequence, and ROUGE-W involves measuring word overlap.

Normalization:

- ROUGE scores are often normalized to ensure consistency across different lengths of summaries. This normalization accounts for the fact that longer summaries are expected to have higher raw recall scores.

In summary, ROUGE provides a set of metrics to quantitatively evaluate the quality of generated text summaries by comparing them to reference summaries. It is widely used in research and development to assess the performance of summarization systems and other natural language generation tasks.

Conclusion 👍
BART performed the best
Then keyBert
Then bert based embedding matching is giving good results.