Text Classification with

Machine Learning Techniques

Fatemeh Alizadeh_918943

Faezeh Azhir_909890

University of Milano Bicocca

Text Mining and search course

Text Summarization Explanation

1. Introduction

Text summarization is a critical task in Natural Language Processing (NLP) that aims to condense long pieces of text into concise summaries while retaining their core meaning. It has numerous applications in fields like news aggregation, academic research, legal document summarization, and customer review analysis.

There are two main approaches to text summarization:

1. **Extractive Summarization**: Focuses on extracting and concatenating the most important sentences or phrases from the text without altering their content.
2. **Abstractive Summarization**: Generates entirely new sentences that capture the essence of the original content. This method requires understanding the text and involves complex NLP models.

Dataset Description

The dataset used is the **Amazon Fine Food Reviews**, which contains over 500,000 customer reviews on various products. Key aspects of the dataset include:

- Review texts (input data).
- Corresponding summaries (target data).

In this project:

- The focus is on abstractive summarization, where the generated summaries are not just extracted sentences but rewritten, capturing the core meaning of the reviews.

- Various Python libraries and tools were used, including:
  - NumPy & Pandas: Data manipulation and analysis.
  - TensorFlow & Keras: Model building using deep learning layers.
  - NLTK: Stopword removal.
  - Transformers: Advanced language models.
  - TXTAI: Pre-built summarization pipelines.
  - ROUGE Scorer: Evaluating summary quality.

Data Preprocessing:

Preprocessing is a crucial step in text summarization because raw text often contains unnecessary noise, which can reduce the accuracy and performance.

In this project, the following preprocessing steps were performed:

- The main dataset contains 568,454 records and 10 features
- The features are : Id ,ProductId , UserId, ProfileName ,HelpfulnessNumerator, HelpfulnessDenominator,Score,Time, Summary,Text
- 100 rows were sampled for analysis
- After dropping missing values the dataset contains 99 unique, non-duplicate reviews with no missing values, across 10 columns
- Convert everything to lowercase
- Remove HTML tags
- Contraction mapping
- Remove (‘s)
- Remove any text inside the parenthesis ( )
- Eliminate punctuations and special characters
- Remove stopwords
- Remove short words

Methods:

Abstractive Summarization

Expresses the ideas in the source document using different words. The model forms its own phrases and sentences to offer a more coherent summary, like what a human would generate. This approach is definitely more appealing, but much more complex and computationally expensive than extractive summarization.Abstractive summarization is a text summarization technique that generates a concise summary by creating new sentences that convey the main ideas of the original text. Unlike extractive summarization, which directly selects sentences or phrases from the source text, abstractive methods aim to produce summaries that are closer to how a human might summarize, often rephrasing or synthesizing information

Key Features:

1. **Generative Approach**: Abstractive summarization involves generating new text that captures the essence of the input while using different phrasing and sentence structures.
2. **Contextual Understanding**: Advanced models analyze the text to understand its overall meaning and produce coherent summaries.
3. **Flexibility**: This method allows for the integration of knowledge, inference, and rephrasing, making it suitable for summarizing complex and varied content

Part 1: Model Selection and Comparison

In this step, we will import and compare three different **text summarization models** from the Hugging Face Model Hub. Each model is designed for summarization tasks but might differ in architecture, training datasets, and output quality.

Selected Models:

1. **facebook/bart-large-cnn:**
    - A transformer-based model fine-tuned on CNN/DailyMail news articles.
    - Known for generating high-quality abstractive summaries.
    - Suitable for general-purpose summarization tasks.
2. **Falconsai/text_summarization:**
    - A fine-tuned summarization model optimized for various text datasets.
    - Balances between extractive and abstractive summarization.
    - Known for its adaptability across different domains.
3. **SEBIS/code_trans_t5_small_code_documentation_generation_php_multitask_finetune:**
    - Originally fine-tuned for **code documentation generation** tasks.

- Despite its primary focus on code summarization, it can be evaluated for textual summarization tasks.
- Based on the **T5 architecture**, known for its flexibility.

we compared **three different text summarization models** to analyze their performance on the same dataset. Each model was applied to generate summaries for cleaned text data, and the results were compared against the original human-generated summaries.

Each model was applied to the cleaned text column in the dataset.

The max_length parameter was set to 10, ensuring concise outputs

Evaluation of Text Summarization Models

In this step, we evaluated the performance of the three text summarization models using **ROUGE** and **BLEU** scores, two widely accepted metrics for summarization quality.

---

1. Evaluation Metrics

Metric Breakdown

- ROUGE-1: Measures the overlap of unigrams (single words) between the generated summary and the reference. It is a basic measure of how much content from the reference is captured.

Interpretation: Higher is better. Low scores suggest the generated summaries capture little overlap with the reference summaries.

- ROUGE-2: Measures the overlap of bigrams (pairs of consecutive words). It evaluates fluency and coherence.

Interpretation: Higher is better. Low scores here suggest the generated summaries lack alignment with the reference's phrase structures.

- ROUGE-L: Measures the longest common subsequence between the reference and generated summary. It reflects sentence-level structure retention.

Interpretation: Higher is better. Low scores indicate that the generated summaries fail to retain important sequential information.

- BLEU: Measures how many n-grams (e.g., words or phrases of different lengths) in the generated summary match the reference, with a penalty for overly short outputs.

Interpretation: Higher is better. Low BLEU scores indicate a poor match between the generated and reference summaries.

---

2. Evaluation Process

- For each review in the dataset:
    1. **Reference Summary:** Taken from the cleaned_summary column.
    2. **Predicted Summaries:** Generated from each model (predicted_summary1, predicted_summary2, predicted_summary3).
    3. **ROUGE and BLEU Scores Calculated:** Compared each predicted summary against the reference.
- The scores were then **averaged across all reviews** for each model to produce the final evaluation metrics.

Observations:

- Model 1 performs best: It has the highest scores across all metrics, albeit still low.
- Model 2 is slightly worse: Its scores are close to Model 1 but slightly lower.
- Model 3 performs the worst: It has significantly lower scores, particularly for ROUGE-2 and BLEU.

Metric	Model 1 (facebook/bart-large-cnn)	Model 2 (Falconsai/text_summarization)	Model 3 (SEBIS/code_trans_t5)
ROUGE-1	0.1662	0.1556	0.1038
ROUGE-2	0.0552	0.0455	0.0076
ROUGE-L	0.1598	0.1519	0.0970
BLEU	0.0254	0.0247	0.0172

Part 2: Text Summarizer Using Deep Learning

This section implements an abstractive summarizer using deep learning techniques with a Seq2Seq model and attention mechanism.

In this section, we implemented a **text summarization model using deep learning techniques**. While summarization tasks generally require **GPU acceleration and high-memory systems** to handle large datasets effectively, we conducted this analysis using a **CPU** on a smaller dataset subset (**1000 rows**) and a limited number of **epochs (20)**.

Although the final results are not optimal, the process followed remains **methodologically sound and valuable** for building scalable text summarization solutions.

3. Data Preprocessing

Effective preprocessing ensures the input data is clean and structured for the model. Below are the steps:

3.1 Cleaning Review Texts

- Convert text to lowercase.
- Remove HTML tags using regular expressions.
- Expand contractions (e.g., "don't" → "do not").
- Remove possessives and words within parentheses.
- Eliminate special characters, punctuation, and stopwords (like "the," "is").
- Filter out short words (e.g., words with fewer than 3 characters).

3.2 Cleaning Summaries

- Follow similar cleaning steps as the reviews.
- Add special tokens START and END to mark the beginning and end of summaries for easier decoding.

---

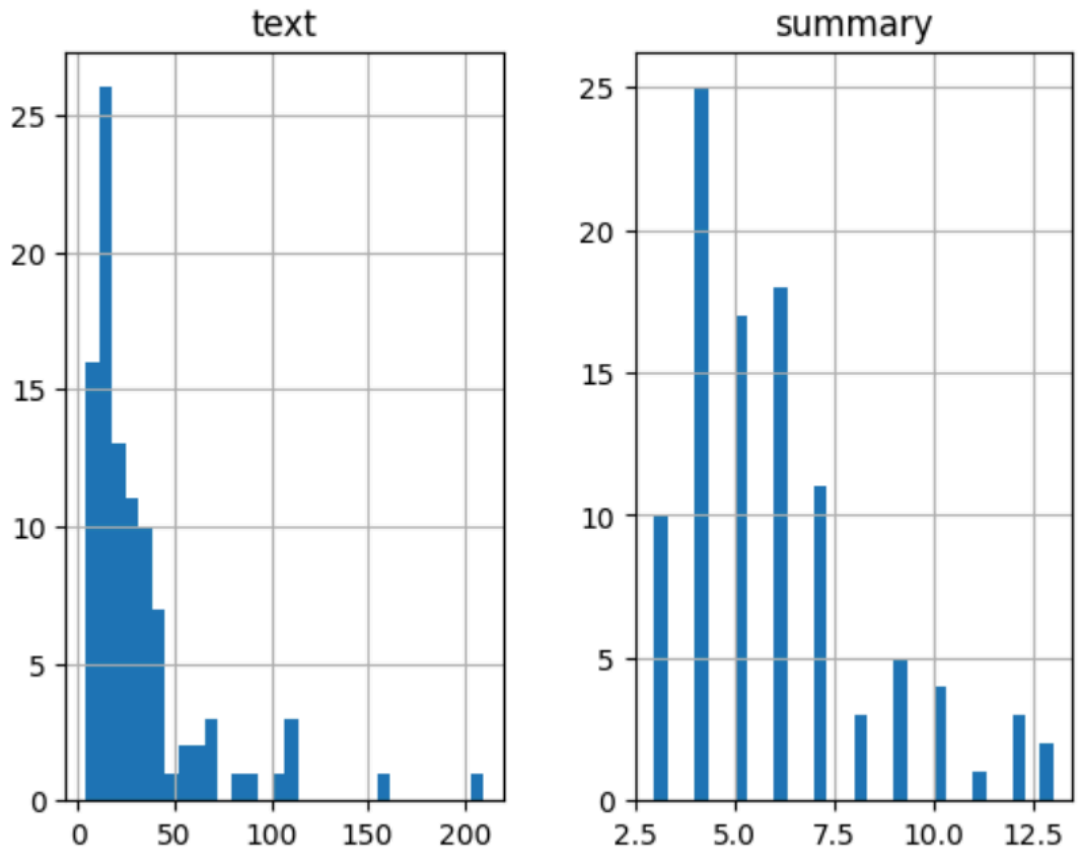
4. Exploratory Data Analysis

The notebook examines the length distributions of both review texts and summaries to:

- Determine appropriate maximum sequence lengths.
- Avoid padding or truncating excessively.

Observations:

- Most reviews are fewer than 100 words.
- Most summaries are shorter than 15 words.
- Final sequence lengths:
    - Review Text: 80 tokens.
    - Summary: 10 tokens.



5. Tokenization and Padding

- **Tokenizer Creation**: Tokenizers for reviews and summaries are built to convert text into sequences of integers, facilitating numerical input for the model.
- **Vocabulary Building**: Top 50,000 unique words are retained to limit the vocabulary size.
- **Padding**: Sequences shorter than the max length are padded, and longer ones are truncated.

6. Model Architecture

The model employs a **Seq2Seq architecture** with an **attention mechanism**:

6.1 Encoder

- Consists of stacked LSTM (Long Short-Term Memory) layers.

- Encodes the input text into a fixed-length context vector.

6.2 Decoder

- Uses another set of LSTM layers to generate summaries from the context vector.
- Integrates an attention mechanism, which dynamically focuses on relevant parts of the input text during each decoding step.

6.3 Attention Mechanism

- Computes a weighted context vector that allows the model to "pay attention" to specific words or phrases in the input sequence.

6.4 Key Parameters

- Embedding size: 300.
- Hidden units in LSTM: 256.
- Dropout: 0.4 (to reduce overfitting).

7. Training Process

The model is trained on the cleaned and tokenized dataset. Key aspects include:

- **Loss Function**: Sparse Categorical Cross-Entropy, suitable for multi-class classification tasks.
- **Optimizer**: Adam optimizer, chosen for its efficiency in training deep neural networks.
- **Early Stopping**: Monitors validation loss and stops training if no improvement is observed, preventing overfitting.

Training is usually conducted for 50 epochs with a batch size of 64, though early stopping often halts training sooner.

8. Model Evaluation

After training, the model is evaluated using example reviews to verify its ability to generate coherent summaries. Metrics such as BLEU or ROUGE scores could be used to quantify performance, though the code focuses on qualitative analysis. Generated summaries are compared with actual summaries.

Observations:

- Generated summaries do not always match word-for-word with the target.
- They effectively capture the main idea, demonstrating the model's capability in abstractive summarization.

9. Conclusion

The code illustrates the implementation of an abstractive text summarizer using deep learning. It highlights:

- The importance of robust preprocessing for model accuracy.
- The effectiveness of Seq2Seq models with attention mechanisms in generating concise summaries.
- Areas for future improvement, such as hyperparameter tuning and using pre-trained embeddings like GloVe or Word2Vec for better context understanding.

Further Insights and Enhancements

1. **Evaluation Metrics**
   Consider using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores to quantitatively measure the overlap between generated and target summaries.

2. **Data Augmentation**
   Increase dataset diversity by using back-translation or paraphrasing techniques to augment training data.

3. **Model Variants**
   Experiment with Transformer-based models like BERT, GPT, or T5, which have shown state-of-the-art results in NLP tasks.