TLN Part 2 - Automatic Summarization with Lexical Vectors

Paletto Andrea, Tuninetti André

September 5, 2023

Contents

1 Introduction

This algorithm aims to reduce the document size by a certain percentage (e.g., 10%, 20%, 30%) through an extractive approach based on the following steps:

1. **Topic Identification:** Identify the topic of the text to be summarized. The topic can be represented as a set of NASARI vectors.
2. **Context Creation:** Create the context by collecting the vectors of terms. This step can be repeated, gradually incorporating the contribution of associated terms at each round.
3. **Sentence Retention:** Retain paragraphs whose sentences contain the most salient terms. Use the Weighted Overlap (WO) metric to determine the salience of terms in paragraphs. The WO metric is calculated as WO(v1, v2).
4. **Paragraph Reranking:** Re-rank the retained paragraphs' weight by applying at least one of the following approaches:
   - Title-based approach
   - Cue-based approach
   - Phrase-based approach
   - Cohesion-based approach

2 Methodology

2.1 Functions Overview

1. `create_nasari_dict()`: This function creates a Nasari dictionary from a given file containing NASARI vectors. It reads the NASARI file, processes each row to extract the key and corresponding vector values, and constructs a dictionary mapping keys (words) to Nasari vectors.
2. `read_document(document)`: This function reads a document file and extracts paragraphs from it. It skips comment lines and short paragraphs to assemble a list of extracted paragraphs, which will be used for further processing.
3. `get_topic_words(paragraphs)`: This function extracts topic words from the title and selected paragraphs. It combines words from the specified indices (title, first paragraph, last paragraph) to form a set of topic words that capture the main theme of the document.
4. `get_context(topic_words, nasari_dict)`: This function generates a Nasari context for a given set of topic words. It retrieves the Nasari vectors corresponding to the topic words from the Nasari dictionary and constructs a set of Nasari vectors representing the context.

5. `rank(word, vector)`: This function ranks a word within a Nasari vector. Given a word and a Nasari vector, it returns the rank of the word in the vector. The rank indicates the position of the word in the vector.

6. `weighted_overlap(v1, v2)`: This function calculates the weighted overlap between two Nasari vectors. It takes two Nasari vectors and computes their weighted overlap score, which indicates the similarity between the vectors based on the ranks of overlapping words.

7. `rank_paragraphs(paragraphs, context, dict_nasari)`: This function ranks paragraphs based on their similarity to a given context represented by Nasari vectors. It computes a rank score for each paragraph by calculating the weighted overlap between the paragraph and the context. Paragraphs are then ranked based on their rank scores.

8. `summarize(document, ranks, title, paragraphs, c_rate)`: This function generates a summary for a document based on the provided ranks and compression rate. It writes the summary to a file, considering the compression rate and the accumulated word count of the selected paragraphs. Additionally, it calculates BLEU and ROUGE metrics to evaluate the quality of the generated summary.

9. `bleu_rouge(retrieved_document, c_rate, document)`: This function calculates BLEU and ROUGE metrics to assess the quality of the generated summary. It compares the set of retrieved words in the summary with the set of relevant words from the original document to calculate precision and recall scores.

2.2 Workflow

This overview outlines the steps involved in the extractive summarization process. The process involves extracting key information from a collection of documents and generating concise summaries based on identified context and topic words.

2.2.1 Initialization and Setup

- The process begins with the setup, where required libraries and modules are imported.
- A list of English stop words is prepared to aid in text processing.
- A Nasari dictionary is created to store word-to-Nasari-vector mappings.

2.2.2 Document Summarization Loop

- A loop iterates through a set of documents in a designated directory, aiming to generate summaries for each document.

2.2.3 Key Steps in the Loop

1. **Document Extraction:**
   - Document contents are extracted into paragraphs using the `read_document()` function.
2. **Topic Identification:**
   - Topic words are extracted from the title and specific paragraphs using the `get_topic_words()` function. These words help capture the document's main theme.
3. **Title Extraction:**
   - The title of the document is extracted and removed from the list of paragraphs.
4. **Context Generation:**
   - A context is generated using the topic words and Nasari vectors through the `get_context()` function.
5. **Paragraph Ranking:**
   - The remaining paragraphs are ranked based on their similarity to the generated context using the `rank_paragraphs()` function.
6. **Summarization at Different Compression Rates:**
   - The script iterates over various compression rates, and for each rate, a summary is generated using the `summarize()` function. The generated summaries are saved for evaluation.

2.2.4 Evaluation

- For the evaluation, we created manually the summaries using https://resoomer.com/en.
  Our automatic-created summary are evaluated for quality using BLEU and ROUGE metrics by comparing them with the manually created summaries.

3 Conclusion

The provided results showcase the performance of the extractive summarization process across various documents and compression rates. The evaluation is conducted using BLEU and ROUGE metrics.

3.1 Observations

- The BLEU precision scores indicate the level of overlap between the generated summary and the original document, where higher scores signify greater overlap.
- The ROUGE recall scores reflect the ability of the summary to capture important information present in the document, with higher scores indicating better recall.

3.2 Document-Specific Insights

- For the document "Ebola-virus-disease.txt," at a compression rate of 10%, the generated summary achieves high precision and recall, indicating a well-balanced and informative summary.
- The document "Napoleon-wiki.txt" demonstrates consistently high BLEU precision scores across different compression rates, suggesting that the summary effectively captures the core content.
- The document "Life-indoors.txt" showcases notable precision at 10% compression, while there is a decrease in recall as the compression rate increases to 30%.
- In the case of "Trump-wall.txt," higher compression rates lead to a slight decrease in both BLEU precision and ROUGE recall scores, indicating potential loss of context.
- The document "Andy-Warhol.txt" demonstrates a favorable balance between BLEU precision and ROUGE recall, with improved recall at higher compression rates.