Semantic Reconstruction and Computational Analysis

• ΣΠΥΡΟΠΟΥΛΟΣ ΗΛΙΑΣ – Π22158

• ΑΘΑΝΑΣΑΚΟΣ ΕΥΣΤΑΘΙΟΣ ΠΑΝΑΓΙΩΤΗΣ – Π22005

GitHub repository: https://github.com/nerflifepls/nlp

Introduction

In our daily communication, clarity is key. Often, texts written by non-native speakers or in a hurry contain grammatical errors or awkward phrasing that can obscure the intended message. **Semantic Reconstruction** is the process of correcting these errors while carefully preserving the original meaning.

This report explores this process by applying both manual and automated Natural Language Processing (NLP) techniques to improve two sample texts. We will not only demonstrate the reconstruction but also use computational methods to measure how well the meaning was preserved. This study highlights how modern NLP tools can be used to refine language and objectively evaluate the results.

Methodology

To improve the original texts, we used a combination of manual and automated strategies. To evaluate the outcome, we used several computational techniques to analyze the text embeddings.

1. Reconstruction Strategies

- **Manual Custom Reconstruction:** This was our most hands-on approach. We identified specific sentences with complex errors—ranging from incorrect grammar (`I am very appreciated`) to awkward phrasing (`Thank your message to show our words`)—and rewrote them to sound more natural and be grammatically correct. This method relies on human understanding of context and nuance.
- **Automated Library-Based Reconstruction:** We used three different Python libraries to automate the correction process on the full texts:
    - **LanguageTool:** A powerful, rule-based tool that checks for a wide variety of grammar, style, and spelling mistakes. It acts as a comprehensive automated proofreader.

- **SpaCy + Custom Rules:** A hybrid approach where we used the SpaCy library to correctly handle sentence structure and then applied a small set of our own specific rules (e.g., changing "updates was" to "updates were").
- **Pyspellchecker:** A specialized library focused on one task: finding and correcting spelling errors.

2. Computational Analysis Techniques

After creating several corrected versions of each text, we needed a way to measure if the meaning had changed.

- **Text Embeddings (Sentence-BERT):** We converted each text version into a sophisticated numerical representation, or "embedding," using the Sentence-BERT model. This model is excellent at capturing the contextual meaning of entire sentences and paragraphs, turning them into a vector of numbers. Texts with similar meanings will have similar vectors.
- **Cosine Similarity:** This is a mathematical metric that calculates the similarity between two vectors. It gives a score from -1 to 1, where 1 means the vectors (and thus the texts' meanings) are virtually identical. We used this to compare the embedding of each reconstructed text against its original version. A score close to 1.0 indicates that the meaning was successfully preserved.
- **PCA and t-SNE Visualization:** Since text embeddings have hundreds of dimensions, they are impossible for humans to visualize. We used two dimensionality reduction techniques, PCA and t-SNE, to compress the embeddings down to 2D. By plotting these 2D points, we can visually see how the different text versions relate to each other. Texts that are semantically similar should appear close together in the plots.

Experiments & Results

Our experiments involved applying the reconstruction methods and then analyzing the results computationally.

1. Before and After Reconstruction Examples

The manual reconstruction process produced significant improvements in readability and correctness.

- **Original:** `I am very appreciated the full support of the professor...`
- **Reconstructed:** `I very much appreciate the full support of the professor...`
- **Original:** `During our final discuss, I told him about the new submission...`
- **Reconstructed:** `During our final discussion, I told him about the new submission...`

- **Original:** `...the updates was confusing as it not included the full feedback...`
- **Reconstructed:** `...the updates were confusing as they did not include the full feedback...`

2. Computational Analysis

The cosine similarity scores were consistently high across all methods, generally **above 0.98**, indicating that the core semantic meaning was very well preserved.

- Simple fixes from libraries like **Pyspellchecker** resulted in scores of **1.0000**, meaning the changes were so minor they did not alter the semantic embedding at all.
- The more comprehensive **manual reconstruction** of Text 1 resulted in a slightly lower similarity of **0.9807**. This is an interesting finding: making a text sound more natural and correct can sometimes involve rephrasing that creates a small, measurable shift in its semantic vector, even while preserving the intended message.

3. Visualization of Embeddings

The PCA and t-SNE plots provide a clear visual confirmation of our findings.

(You can insert the image from your output here.)

- **PCA Plot (Left):** This plot shows two main clusters. The points for Text 1 and all its reconstructed versions are grouped tightly in the top left. The points for Text 2 are grouped in the center-right. The tightness of these clusters visually shows that the reconstructions remained very close in meaning to their originals.
- **t-SNE Plot (Right):** This plot, which is better at showing local similarities, gives a clearer separation. Again, the different versions of Text 1 form a distinct group, as do the versions of Text 2. The fact that the original text and its corrections are clustered together, and far from the other text's cluster, confirms that our reconstruction process successfully maintained the core topic and meaning of each document.

Discussion

How well did the embeddings capture meaning?
The embeddings, particularly from Sentence-BERT, were remarkably effective. The high cosine similarity scores and the tight visual clusters in the plots prove that the models correctly identified the reconstructed texts as being semantically almost identical to their originals.

What were the biggest challenges in reconstruction?
The primary challenge is balancing grammatical correction with the preservation of nuance. Fixing a

simple typo is easy. The difficult part is rewriting an awkwardly phrased sentence without accidentally changing its subtle intent. Automated tools are good at the former, but human understanding is often needed for the latter.

How can this process be automated using NLP?
This process can be significantly automated. While rule-based systems like LanguageTool are effective, the future lies in training advanced machine learning models. By providing a model (like T5 or BERT) with thousands of "before" and "after" examples, it can learn to perform sophisticated, context-aware corrections that go beyond simple rules. The bonus task in our script, which used a Greek BERT model to predict missing words, is a small-scale example of this predictive power.

Were there differences in quality between the methods?
Yes, there were clear differences:

- **Pyspellchecker:** Too limited. It only fixed spelling and missed all grammatical and stylistic errors.
- **LanguageTool:** A great general-purpose tool. It caught a wide range of common errors effectively.
- **Manual Reconstruction:** The gold standard for quality. A human can understand context and intent in a way that automated tools currently cannot, leading to the most natural-sounding text. However, this method is slow and not scalable.

Conclusion

This study successfully demonstrated that NLP techniques can be used to reconstruct and improve flawed texts. Our computational analysis, using cosine similarity and embedding visualizations, provided a robust and objective way to verify that the semantic meaning of the texts was preserved throughout the process.

While automated tools are incredibly useful for catching common errors, the best results often come from a human-in-the-loop approach, where the tool handles the simple fixes, and a person addresses the more nuanced issues. As NLP models continue to advance, the gap between automated and manual quality is shrinking, promising even more powerful tools for clear and effective communication in the future.

Bibliography

Models & Frameworks

- **Sentence-BERT (`all-MiniLM-L6-v2`):** A framework for creating state-of-the-art sentence and text embeddings. Used as the primary method for semantic analysis and visualization.
    - *Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.*
- **Greek-BERT (`nlpaueb/bert-base-greek-uncased-v1`):** A BERT model pre-trained on a large corpus of Greek text. Used in the bonus section for the masked language modeling task.
    - *Koutsikakis, J., et al. (2020). GREEK-BERT: The Greek BERT.*
- **spaCy (`en_core_web_lg`, `el_core_news_sm`):** An industrial-strength natural language processing library. Used for sentence segmentation, tokenization, and generating document vectors for similarity comparison.

Python Libraries

- **LanguageTool (`language_tool_python`):** A Python wrapper for the LanguageTool rule-based grammar and style checker, used for automated text reconstruction.
- **Pyspellchecker:** A library for identifying and correcting spelling errors, used for a focused reconstruction approach.
- **Scikit-learn:** A fundamental machine learning library for Python. Used to perform PCA and t-SNE for dimensionality reduction of the text embeddings.
- **Gensim:** A library for topic modeling and document similarity analysis. Used to load pre-trained Word2Vec and FastText models for comparative analysis.

Data & Ground Truth Source

- **OpenAI's ChatGPT:** The language model was utilized to provide the "source of truth"—the correct, original words—for the masked Greek legal text snippets in the bonus section of the analysis. This allowed for a comparison between the Greek-BERT model's predictions and a known correct answer.