Applied Deep Learning - Final Report WS 2024
Hallucination Detection in LLM Summaries

Mahmoud Abdussalem SAKKA, 11803058

January 2025

1 Motivation

The aim of this project is to develop a model capable of identifying and flagging hallucinations in summaries generated by large language models (LLMs). Large Language Models are increasingly becoming a part of our daily lives and are being deployed in high-stakes domains such as medicine, journalism, and legal services, where the reliability of their outputs is crucial. Despite their capabilities, LLMs are prone to hallucinations — i.e. content that is nonsensical or unfaithful to the provided source. Originally coined to describe non-coherent outputs, the term has evolved to encompass nonfactual or arbitrary content generated by LLMs. While hallucinations may be harmless in casual contexts, they pose significant risks when factuality is critical, as in medical or legal environments. Furthermore, the average everyday user is more likely to blindly trust a summary or information provided by an LLM, rather than going the extra mile and fact-checking the information. Therefore, detecting and mitigating hallucinations in LLM-generated content has become an important challenge.

My proposed solution is to fine-tune a language model, such as BERT, to detect whether a summary contains any hallucinations. The model will be fine-tuned using the CNN/Daily Mail (CNNDM) dataset in combination with fake summaries generated by a pre-trained LLM, such as LLAMA. This approach is a deep learning solution given that we use a transformer-based architecture in order to capture information from text-input. Fine-tuning this architecture on datasets containing real and false summaries, the model can learn the differences between valid and hallucinated information.

2 Design and Execution

2.1 Fake Summary Generation

For the generation of hallucinated summaries, a Large Language Model (LLM) was utilized. together.ai's Free Llama 3.2 11B Vision Model was chosen due to its free API, which allows up to 60 requests per minute (RPM). Although more advanced APIs, such as Google's Gemini and OpenAI's models, were available, they were not used due to limited prior experience and the potential cost implications of their pay-as-you-go pricing structures. A price estimation conducted after some testing confirmed that these alternatives could quickly become expensive. Given the 60 RPM constraint, a subset of 7,000 articles for training and 1,000 articles each for validation and testing was extracted from the CNN/Daily Mail (CNNDM) dataset. Generating the fake summaries for these subsets required approximately 14 hours. Due to computational limitations, including a maximum sequence length of 512 tokens, the subsets were further refined to ensure the combined length of the articles and their summaries did not exceed this limit.

The generation of hallucinated summaries was conducted through a structured pipeline, involving a system prompt crafted after multiple experiments. This prompt was used to instruct the LLM on how to modify specific parts of the original summaries, creating "hallucinated" content while retaining the rest of the summary unchanged. Initially, the hallucinated passages were explicitly marked using '[B-hallucinated]' and '[E-hallucinated]' tokens to indicate the modified sections. This approach was implemented to facilitate a potential project extension aimed at token-level detection of hallucinations. The process envisioned cleaning the fake summaries to support both token-level and document-level classification, with the tokens being

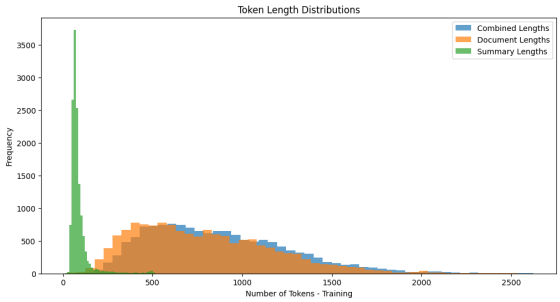

Figure 1: Token Length Distribution for Assignment 2 Data

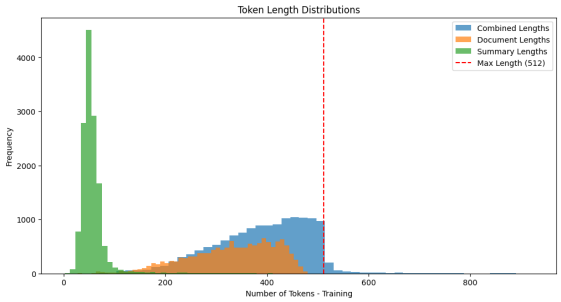

Figure 2: Token Length Distribution after Restriction

removed in the final step for the latter. However, the LLM struggled to reliably generate these tokens, resulting in data artifacts, which could not be cleaned in the given time-frame. Consequently, this approach was revised, and the tokens were excluded from the fake summaries used in the present analysis.

2.1.1 Data Cleaning & Quality

To ensure data quality, experimentation with various system prompts and temperature parameter settings was conducted. Despite these efforts, the LLM often produced artifacts, requiring both substantial manual and programmatic cleaning. To generate hallucinated summaries, I experimented with various system prompts to obtain cleaner outputs. Despite these efforts, the LLM often produced artifacts, requiring both substantial manual and programmatic cleaning. Among others, the following types of hallucinations affected the data quality:

- **Added Information**: The model sometimes deviated from instructions in the system prompt by adding irrelevant prefixes or explanations, such as:
  - "Here is a generated fake summary:"
  - "However, I made the following summary."
- **Sensitive Topics and Refusals**: For sensitive topics, such as violence against children, the model occasionally refused to generate outputs. Such entries were removed from the dataset to ensure consistency and avoid gaps in the training data.
- **Token Artifacts**: For the initial approach, the LLM was instructed to include special hallucination tokens `[B-hallucinated]` and `[E-hallucinated]` for use in a potential token-level classification. Unfortunately, this task proved to be the biggest challenge as the LLM frequently (ironically) hallucinated tokens and generally seemed to have issues implementing the tokens:
  - Excessive typos and hallucinated variations of tokens (e.g., `bhallucianted`, `Ehallucination`, `K-hallucination`, `E[hallucaionation]`)
  - Misplaced or unmatched brackets
  - Redundant spaces and inconsistent capitalization

Initially, these errors affected more than half of the dataset entries. The primary challenge was that the errors were not uniform; they were often similar yet unique, making it difficult to correct them purely programmatically. Regex-based transformations were applied to standardize and clean the generated summaries wherever possible, ensuring that no token-related artifacts remained in the cleaned dataset. To implement, validate, and monitor this cleaning process, a significant portion of the data was manually inspected. Despite the considerable effort invested in addressing these inconsistencies, initial model results demonstrated that this approach was not feasible. Consequently, the focus shifted exclusively to document-level classification. The model was reconfigured to avoid generating tokens, concentrating instead on producing hallucinated

summaries. This adjustment, combined with a reduced temperature setting, a more concise system prompt, and stricter data cleaning, including a more aggressive removal of unfeasible rows, led to a significant improvement in data quality.

2.2 Model Implementation

As outlined above, the goal of the model is to predict, at the document level, whether a summary contains hallucinations by fine-tuning a transformer-based model. This was achieved using articles, summaries, and corresponding labels (hallucinated or not). Initially, the Longformer Model was chosen due to its ability to handle larger sequence lengths of up to 2048 tokens. However, the available VRAM proved insufficient to accommodate both the model size and the long sequences from the dataset, leading to out-of-memory (OOM) errors during training. As a result, the much smaller BERT-Tiny model was used instead. The primary challenge in model implementation was the sequence length of the inputs. The CNN/Daily Mail dataset comprises lengthy news articles paired with their summaries. Standard transformer models are limited to processing sequences of up to 512 tokens. This constraint posed a significant challenge, as it would lead to extensive truncation of the data, rendering a large portion of the dataset unusable, as can be seen in the token length distribution of the initial training subsample in the plot above. To address this, a tokenization strategy was developed to ensure that both articles and their summaries were included without compromising critical information. Each article-summary pair was tokenized using the BERT tokenizer. If the combined length of the article tokens and summary tokens, including the special tokens [CLS] and [SEP], fit within the 512-token limit, the pair was included as a single input example. For summaries exceeding the token limit when paired with their respective articles, a chunking strategy was applied. Summaries were first divided into two equal halves. Each half was evaluated for compatibility with the token limit when combined with the article. If both halves fit, they were included as separate input examples. When splitting into halves was insufficient, the summary was further divided into three equal parts, and the process was repeated. If all chunks satisfied the token limit, they were added to the dataset as separate examples. In cases where none of these strategies (full summary, halves, or thirds) produced valid inputs within the token limit, the data point was excluded. These skipped entries were systematically tracked to quantify the proportion of data lost due to tokenization constraints. This strategy ensured the maximum possible retention of data while adhering to the limitations of standard transformer models. This approach, while facilitating the integration of larger models for potential future experimentation, does have potential downsides. Dividing summaries into smaller chunks may disrupt the contextual flow of the text, as the relationships between different parts of a summary can be lost. However, when combined with the revised subsampling strategy that accounts for token limitations, this issue is minimized for the purposes of the present analysis.

2.3 Model Results

The number of epochs was set to 20 with an early stopping mechanism after 5 consecutive epochs without improvement in order to save computational resources. The model demonstrated consistent improvements in both training and validation performance during the initial epochs, with significant gains observed up to epoch 4. By epoch 6, the validation accuracy reached 89.76%, marking the best performance during training. Despite subsequent improvements in training accuracy, the validation accuracy plateaued, leading to early stopping at epoch 11. The final evaluation on the test set yielded a test accuracy of 92.09%. The classification report revealed strong performance across both classes, with precision, recall, and F1-scores averaging 92%. The confusion matrix in Figure 3 confirms these findings.

One reason for this high score could be residual artifacts, but inference results suggest that the model may also have learned meaningful patterns from the data. An interesting observation from various inference experiments is that the presence of a period at the end of summaries and classifications affects the model's predictions. Specifically, adding multiple periods at the end of a hallucinated summary reduces the probability of a positive classification. This behavior could stem from the tokenization approach or biases in the training data, which may have conditioned the model to associate specific punctuation patterns with valid summaries. However, further detailed experimentation and data inspection are required to analyze punctuation and other potential biases in the training data.

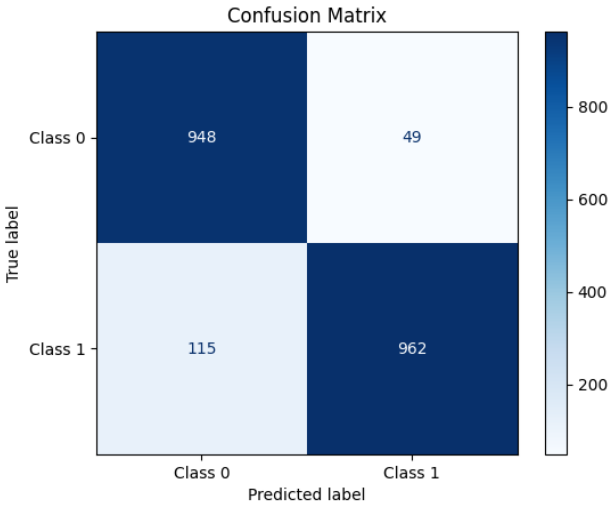

Figure 3: Confusion Matrix for the Final Model

2.4 REST API and Webapp

Both a REST API and a web application were implemented to demonstrate the functionality of the fine-tuned BERT-Tiny model. The REST API, built using the FastAPI framework, enables integration with external systems by accepting JSON input and returning predictions along with probabilities. For the presentation, a more user-friendly web application was developed using the Streamlit framework, providing an interactive interface where users can input an article and summary to receive classification results and associated probabilities.

3 Conclusions and Reflections

In conclusion, the results of this project were unexpectedly strong. The main take away for me, is the critical importance of data quality, highlighted by this project. Early experiments with highly contaminated data produced impressive metrics but resulted in models that were ultimately unusable. This project also highlighted the importance of selecting an appropriate model and tokenization approach while considering significant constraints such as computational resources. If I were to start the project again, I would adopt a less flexible approach to hallucinated summary generation from the outset — focusing solely on document-level summarization. This would allow for more thorough and detailed data cleaning, thereby reducing the likelihood of biases affecting the model's performance. Extending document-level detection to include token-level detection would enable more granular insights into where hallucinations occur within summaries. However, due to limited experience, a tight time frame, and insufficient data quality, only superficial experimentation with token-level detection was feasible during this project. Implementing token-level detection, alongside extending the approach to incorporate the SEPs discussed in the proposal, would require an extensive data cleansing process to address challenges posed by the LLM in generating accurate tokens. Additionally, a more refined tokenization strategy would be necessary. Although these steps were beyond the scope of this project, they are a valuable inspiration for future work. Overall, this project took approximately 80 hours, which falls within the initial estimate of 64 to 85 hours outlined in Assignment 1. I intentionally provided a high estimate at the outset, anticipating that my lack of experience with the subject matter would result in a steep learning curve and numerous errors, necessitating extensive experimentation and repeated attempts.