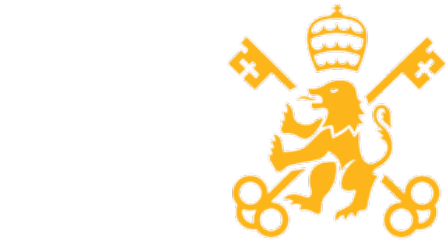


COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

Optimizing Prompt Engineering for Improved Generative AI Content

Master's Thesis in
Machine Learning Engineering

Author: **Pablo Ortolan**

Student ID: 202216408
Advisor: Prof. Federico Muñoz Babiano
Co-advisors: Mario Castro Ponce
Academic Year: 2022-23

English Abstract

Optimizing Prompt Engineering for Improved Generative AI Content,
by Pablo ORTOLAN

Prompt engineering is the process of designing and optimizing prompts for generative artificial intelligence models. The typical use case I want to explore is ordinary people searching for information on generative AI. In this master thesis, I explore techniques to develop new approaches using crafted role and tone prompts to improve the quality of content generated by a specific AI model: ChatGPT-3.5-Turbo. After studying several AI metrics, I choose Rouge-L-Sum and BERTScore to evaluate the efficiency of the newly generated sentences. I use the wiki_qa dataset, a Question-Answering dataset, a collection of questions and corresponding answers in English based on Wikipedia content. The analysis proves that on this specific dataset, ChatGPT 3.5-Turbo works best if the crafted prompt contains no role and an authoritative tone. This thesis is globally conducted over the dataset and doesn't highlight the type of question where each role is more efficient, it could be a future development of this work. Still, it is important to study existing models like ChatGPT-3.5-Turbo in this thesis. It contributes to understand deeper models and stepping back from the dangerous race to ever-larger unpredictable black-box models with emergent capabilities.

Keywords: Artificial Intelligence, Generative AI, Prompt Engineering, ChatGPT-3.5-Turbo, wiki_qa

Spanish Abstract

Optimización de la ingeniería de prompts para mejorar el contenido de la IA generativa, por Pablo ORTOLAN

La ingeniería de prompts es el proceso de diseño y optimización de prompts para modelos de inteligencia artificial generativa. El caso de uso típico que quiero explorar es el de la gente corriente que busca información sobre IA generativa. En esta tesis de máster, exploro técnicas para desarrollar nuevos enfoques utilizando prompts de rol y tono elaborados para mejorar la calidad del contenido generado por un modelo de IA específico: ChatGPT-3.5-Turbo. Tras estudiar varias métricas de IA, elijo Rouge-L-Sum y BERTScore para evaluar la eficacia de las nuevas frases generadas. Utilizo el conjunto de datos wiki_qa, un conjunto de datos de preguntas-respuestas, una colección de preguntas y sus correspondientes respuestas en inglés basadas en el contenido de Wikipedia. El análisis demuestra que, en este conjunto de datos específico, ChatGPT 3.5-Turbo funciona mejor si la pregunta elaborada no contiene ninguna función y tiene un tono autoritario. Esta tesis se realiza de forma global sobre el conjunto de datos y no destaca el tipo de pregunta donde cada rol es más eficiente, podría ser un desarrollo futuro de este trabajo. Aún así, es importante estudiar modelos existentes como ChatGPT-3.5-Turbo en esta tesis. Contribuye a comprender modelos más profundos y a dar un paso atrás en la peligrosa carrera hacia modelos de caja negra impredecibles cada vez más grandes con capacidades emergentes.

Palabras clave: Inteligencia Artificial, Generative AI, Prompt Engineering, ChatGPT-3.5-Turbo, wiki_qa

Contents

Introduction

0.1. Definition and Context

Prompt engineering is the process of designing and optimizing input prompts for generative artificial intelligence models. A prompt, a short piece of text or other input, serves as the starting point for the model to generate new content. In the context of recent breakthroughs in generative AI, such as ChatGPT or Bard, the quality and relevance of prompts are becoming increasingly important.

Many classes exist when referring to AI models. Large Language Models (LLM) refer to one of these classes and are designed to understand and generate human language. They can have different architectures such as a GPT one (Generative Pre-trained Transformer). To analyze this model, I am using Natural Language Processing. That is to say, a set of tools and techniques specialized in human language.

0.2. Objective

The primary objective of this project is to analyze and explore techniques that can be used to improve prompt engineering. Specifically, I aim to gain a deeper understanding of the capabilities of AI models and the sources of errors that arise during prompt generation. The typical use case I want to explore is ordinary people searching for information on generative AI like ChatGPT or Bard. The new search engines era has begun and they are personalised, accurate and scalable. I hope to develop new approaches to optimize prompts and improve the quality of content generated by AI models.

0.3. Scope

In this master thesis, the scope is the investigation of various prompt design strategies, evaluation of their impact on AI model performance, and proposal optimization approaches for achieving better outcomes. It involves experimentation with different prompt formats, lengths, and language styles. The study is conducted using a specific generative

AI model: ChatGPT GPT-3.5, architecture.

AI models can perform various tasks. Some models are unique-task oriented, multi-task oriented or general with a fine-tuning on one specific task. Here is a non-exhaustive list of the tasks that can be performed with AI models, by category:

- Multimodal
  - Feature Extraction
  - Text-to-Image
  - Image-to-Text
  - Text-to-Video
  - Visual Question Answering
  - Document Question Answering
  - Graph Machine Learning
- Computer Vision
  - Depth Estimation
  - Image Classification
  - Object Detection
  - Image Segmentation
  - Image-to-Image
  - Unconditional Image Generation
  - Video Classification
  - Zero-Shot Image Classification
- Natural Language Processing
  - Text Classification
  - Token Classification
  - Table Question Answering
  - Question Answering
  - Zero-Shot Classification

  - Translation
  - Summarization
  - Conversational
  - Text Generation
  - Text2Text Generation
  - Fill-Mask
  - Sentence Similarity
- Audio
  - Text-to-Speech
  - Automatic Speech Recognition
  - Audio-to-Audio
  - Audio Classification
  - Voice Activity Detection
- Tabular
  - Tabular Classification
  - Tabular Regression
- Reinforcement Learning
  - Reinforcement Learning
  - Robotics

This master thesis aims to give tools and advice about prompt engineering on conversational AI such as ChatGPT. In light of this goal, the scope of investigation encompasses the following tasks: Question answering, summarization and conversational.

These tasks have been chosen because they are closer to what an AI is doing today.

0.4. Limitations

While studying prompt engineering for generative AI content optimization, there are certain limitations to consider. These limitations could include:

- Generalizability: The findings and optimizations proposed within the study may be specific to the chosen generative AI models, prompting techniques, and datasets used. The generalizability of the results to other models or domains should be acknowledged.
- Subjectivity of Evaluation: Evaluating the quality and effectiveness of generative AI content is often subjective, as it relies on human judgment. While using generative AI metrics can provide some objectivity, there may still be inherent biases and challenges in assessing the content's relevance and coherence accurately.
- Resource Constraints: The study may be limited by computational resources, access to specialized software or hardware, and time constraints, which could impact the scale and complexity of the experiments and analyses conducted.
- External Factors: The study's scope might not account for external factors such as evolving AI technologies, changes in datasets, or emerging best practices in the field of generative AI, which may impact the long-term applicability of the proposed prompt engineering optimizations.

In this thesis, all these limitations will be considered. When it comes to choosing a dataset, analysing the results or drawing conclusions, I will always moderate my review.

1 | Literature Review

Understanding generative AI metrics is crucial before studying prompt engineering. Metrics provide a systematic framework to assess the quality of AI-generated content. It enables the optimization of prompts to generate desired outputs, enhancing the overall quality and relevance of AI-generated responses. Understanding the ins and outs of each metric will permit us to compare them on every facet they are more oriented with.

Large Language Models, which is abbreviated LLM have emerged as powerful tools in Natural Language Processing (NLP) and have revolutionized various applications, such as language generation, machine translation, question answering, and text classification. As the field of NLP continues to advance rapidly, the need for robust and reliable metrics to evaluate the performance of LLMs becomes increasingly important.

Metrics are critical for assessing the quality and effectiveness of the models. They provide quantitative measures to evaluate the performance on specific tasks and enable comparison of different LLMs and track their progress over time. Moreover, metrics play a crucial role in benchmarking, model selection, and model fine-tuning, as well as in guiding the development and improvement of LLMs.

In this literature review, I provide a comprehensive overview of the metrics used in the evaluation of LLMs. I review the existing literature and highlight the different types of metrics commonly used. I discuss their purpose, strengths and limitations, as well as their applications in various NLP tasks. Furthermore, I examine the challenges and open questions in the field of LLM metrics, such as the need for interpretability and generalization in metric design.

Overall, the literature review aims to provide a comprehensive overview of the current landscape of metrics in the evaluation of LLMs, highlighting the importance of reliable and robust evaluation metrics in advancing the field of NLP and enabling the development of more effective and efficient models.

1.1. Metrics explanation and critical analysis

Different types of metrics are used to measure different aspects of LLMs' quality and effectiveness. In this literature review, I will study extrinsic metrics. The ones that evaluate the performance of a model or algorithm based on its contribution to one or more specific tasks. These metrics assess the model's effectiveness in achieving the intended goal and measure its impact on overall task performance.

I will focus on three main tasks in NLP:

- Text Summarization: Text summarization involves generating a concise and coherent summary of a given text or a collection of texts. It aims to capture the most important information while reducing the length of the original text.
- Text Generation: Text generation refers to the process of automatically generating new textual content. This can include generating sentences, paragraphs, articles, or even dialogue in a conversational setting.
- Text Translation: Text translation involves converting text from one language to another. It accurately conveys the meaning and intent of the source text in the target language.

In this section, metrics will be described in order of publication by following the same pattern.

After explaining the name and acronym, I will begin by providing a concise definition of the metric and its primary purpose and describe how it is calculated or derived.

Then, I specify how the metric's output is interpreted. Every single metric output ranges from 0 to 1 where 1 corresponds to the best output possible. All input requirements are the same, every metric requires a generated text as input, typically in the form of sentences, paragraphs, or documents.
It also requires one or more reference texts, which are human-generated texts or target texts that serve as benchmarks for comparison.

After that, I will expose the typical use cases or applications of the metric and, if relevant, describe how the metric relates to other commonly used metrics in the same domain. Finally, I will highlight any known limitations or caveats associated with the metric.

1.1.1. BLEU

BLEU, standing for Bilingual Evaluation Understudy, was introduced in 2001 by IBM. This metric main task is the automatic evaluation of machine translation and text summarizing. It measures the similarity between the generated text and reference texts, often human-generated.
It provides a numerical score that indicates the degree of overlap or similarity between the generated text and the reference text, focusing on precision.

According to Papineni Papineni et al. ((2002)), BLEU is a string-matching metric. The calculation method of the BLEU metric is based on the evaluation of the n-gram overlapping between generated and reference summaries.

BLEU applies the modified unigram precision. The reference word should be considered exhausted after a matching candidate word is identified.
BLEU also introduces a multiplicative factor to penalize long sentences. Indeed, the sentence brevity penalty equals 1.0 when the candidate length is the same as any reference translation length.
Experimentally, we obtain the best correlation with human judgments using a maximum n-gram order of 4.

The BLEU metric ranges from 0 to 1. Few translations will attain a score of 1 unless they are identical to a reference translation.

BLEU is related to other metrics used for evaluating machine translation and text generation, such as NIST (N-gram-based Evaluation Metric) and METEOR (Metric for Evaluation of Translation with Explicit ORdering). These metrics have similar goals but may differ in their calculation methods and specific considerations.

BLEU is primarily based on n-gram matching and does not capture semantic or contextual aspects of translation quality. It can be sensitive to factors such as tokenization and reference selection. BLEU does not account for fluency, grammar, or other linguistic aspects of the generated text. It is important to consider the limitations of BLEU and use it in conjunction with other evaluation metrics and human judgment for a comprehensive assessment of translation quality.

1.1.2. ROUGE

ROUGE, introduced in 2004, stands for Recall-Oriented Understudy for Gisting Evaluation. ROUGE aims to automatically evaluate text summarising. It measures the overlap

between the generated summary and reference summaries, focusing on recall. According to LinLin ((2004)), ROUGE metric calculation depends on the variants:

- ROUGE-N: Overlap of n-grams between the system and reference summaries.
- ROUGE-1 refers to the overlap of unigrams between the system and reference summaries.
- ROUGE-2 refers to the overlap of bigrams between the system and reference summaries.
- ROUGE-L: Longest Common Subsequence (LCS) based statistics. The longest common subsequence problem takes into account sentence-level structure similarity naturally and automatically identifies the longest co-occurring in sequence n-grams.
- ROUGE-W: Weighted LCS-based statistics that favour consecutive LCSes.
- ROUGE-S: Skip-bigram based co-occurrence statistics. Skip-bigram is any pair of words in their sentence order.
- ROUGE-SU: Skip-bigram plus unigram-based co-occurrence statistics.

The ROUGE scores range from 0 to 1, where a higher score indicates a better match between the generated summary and the reference summary. The specific implementation of every variant may vary depending on the goal to achieve. ROUGE has been extensively used and validated in research and industry. It has been empirically evaluated against human judgments and has shown a reasonable correlation with human assessments of summary quality.

According to this paperLin ((2004)), ROUGE-2, ROUGE-L, ROUGE-W, and ROUGE-S worked well in single document summarization tasks whereas ROUGE-1, ROUGE-L, ROUGE-W, ROUGE-SU4, and ROUGE-SU9 performed great in evaluating very short summaries.

However, how to achieve a high correlation with human judgments in multi-document summarization tasks as ROUGE already did in single document summarization tasks is still an open research topic

1.1.3. METEOR

METEOR, introduced in 2005, stands for Metric for Evaluation of Translation with Explicit ORdering.

METEOR is a heuristic matching metric. It was designed to explicitly address the weak-

nesses in BLEU and not only match identical words but also match words that are simple morphological variants or synonyms of each other.

According to BanerjeeBanerjee and Lavie ((2005)), METEOR calculates several component scores and combines them using a weighted harmonic mean to obtain the final METEOR score:

- Unigram Precision and Recall: Measures the precision and recall of matching unigrams between machine translation and reference translations.
- Alignment Score: Considers the alignment and chunking of matched phrases to capture larger linguistic units.
- Stemming Score: Accounts for stemming variations in words.
- Synonym Score: Incorporates synonyms to capture word-level semantic similarity.
- Paraphrase Score: Considers paraphrases to account for lexical and structural variations.

The METEOR metric described and evaluated in this paperBanerjee and Lavie ((2005)) is still relatively simple and naive. For example, it is not using semantic relatedness to map unigrams.

1.1.4. TER

Translation Edit Rate (TER) ((Snover et al., 2006)), measures the amount of editing that a human would have to perform to change a system output so it exactly matches a reference translation. Snover shows that the single-reference variant of TER correlates as well with human judgments of MT quality as the four-reference variant of BLEU. This paper also defines a human-targeted TER (or HTER) and shows that it yields higher correlations with human judgments than BLEU—even when BLEU is given human-targeted references. Our results indicate that HTER correlates with human judgments better than HMETEOR and that the four-reference variants of TER and HTER correlate with human judgments as well as—or better than—a second human judgment does.

1.1.5. NIST MT

The paper Przybocki et al. ((2009)), discusses the evaluation of automated metrics developed to evaluate machine translation technology. A general discussion of the usefulness of automated metrics is offered. The NIST MetricsMATR evaluation of MT metrology is described, including its objectives, protocols, participants, and test data. The methodol-

ogy employed to evaluate the submitted metrics is reviewed. A summary is provided for the general classes of evaluated metrics. Overall results of this evaluation are presented, primarily by using the means of correlation statistics, showing the degree of agreement between the automated metric scores and the scores of human judgments. Metrics are analyzed at the sentence, document, and system level with results conditioned by various properties of the test data. This paper concludes with some perspective on the improvements that should be incorporated into future evaluations of metrics for MT evaluation.

1.1.6. chrF

In this paper Popović ((2015)), the authors propose the use of the character n-gram F-score for automatic evaluation of machine-translation output. Character n-grams have already been used as a part of more complex metrics, but their individual potential has not been investigated yet. Popovic reports system-level correlations with human rankings for 6-gram F1-score on the WMT12, WMT13 and WMT14 data as well as a segment-level correlation for 6-gram F1 and F3-scores on WMT14 data for all available target languages. The results are very promising, especially for the CHRF3 score – for translation from English, this variant showed the highest segment-level correlations outperforming even the best metrics on the WMT14 shared evaluation task.

1.1.7. GLUE

Natural language understanding models, although versatile, often struggle with out-of-domain data. To address this limitation and promote deeper comprehension, a unified model capable of handling diverse linguistic tasks is essential. The General Language Understanding Evaluation (GLUE) Wang et al. ((2018)), serves as a benchmark for nine NLU tasks, offering an auxiliary dataset and an online platform for model evaluation. GLUE encourages the development of NLU systems that can transfer linguistic knowledge across tasks while maintaining task-specific components. Baseline models like ELMo have been assessed within GLUE, revealing relatively low absolute scores. The diagnostic analysis highlights overall weak performance in tested phenomena, emphasizing the need for improved models. GLUE covers various domains, data quantities, and difficulties, rewarding models that generalize well. Multi-task learning and various pre-trained techniques are employed during the evaluation. Developing generalizable NLU systems and addressing the challenges posed by data-scarce tasks remain key areas of focus within the GLUE benchmark.

1.1.8. BERTScore

BERT, introduced in 2018 by Google, stands for Bidirectional Encoder Representations from Transformers. Devlin et al. ((2018))

Traditionally, models in NLP were trained in a left-to-right or right-to-left manner, making them contextually unaware of the entire sentence during training. BERT introduced a bidirectional training approach that allows the model to capture the context from both directions.

BERTScore is an automatic evaluation metric for text generation, machine translation and image captioning tasks based on BERT model.

According to Tianyi Zhang Zhang et al. ((2019)), BERTScore calculates the similarity between the contextual embeddings of words and phrases in the generated and reference texts. It uses the BERT model to encode the text into contextual embeddings, capturing the meaning and context of words and phrases. The calculation involves computing a similarity score based on the cosine similarity (weighted with inverse document frequency scores) between the embeddings of corresponding words and phrases in the generated and reference texts.

In practice, BERTScore uses greedy matching to maximize the matching similarity score, where each token is matched to the most similar token in the other sentence. BERTScore can be exposed under three variants, recall, precision and F1 measure:
$R_{BERT}, P_{BERT}, F_{BERT}$

BERTScore resolves some limitations of commonly used metrics, especially for challenging adversarial examples. Moreover, it correlates better with human judgments and provides stronger model selection performance than existing metrics. However, BERTScore performance depends on the quality and suitability of the pre-trained BERT model.

1.1.9. MoverScore

In this paper Zhao et al. ((2019)), they investigate strategies to encode system and reference texts to devise a metric that shows a high correlation with human judgment of text quality. It introduces a new metric, namely MoverScore, on several text generation tasks including summarization, machine translation, image captioning, and data-to-text generation, where the outputs are produced by a variety of neural and non-neural sys-

tems. Their findings suggest that metrics combining contextualized representations with a distance measure perform the best. Such metrics also demonstrate strong generalization capability across tasks. For ease of use, they make our metrics available as a web service.

1.1.10. SuperGLUE

The paper Sarlin et al. ((2020)) introduces SuperGlue, a neural network that matches two sets of local features by jointly finding correspondences and rejecting non-matchable points. Assignments are estimated by solving a differentiable optimal transport problem, whose costs are predicted by a graph neural network. We introduce a flexible context aggregation mechanism based on attention, enabling SuperGlue to reason about the underlying 3D scene and feature assignments jointly. Compared to traditional, hand-designed heuristics, our technique learns priors over geometric transformations and regularities of the 3D world through end-to-end training from image pairs. SuperGlue outperforms other learned approaches and achieves state-of-the-art results on the task of pose estimation in challenging real-world indoor and outdoor environments. The proposed method performs matching in real-time on a modern GPU and can be readily integrated into modern SfM or SLAM systems.

1.1.11. BLEURT

BLEURT, introduced in 2020 by Thibault Sellam, stands for Bilingual Evaluation Understudy with Representations from Transformers.
BLEURT is a text generation metric for English based on BERT.
A key ingredient of BLEURT is a novel pre-training scheme that uses random perturbations of Wikipedia sentences augmented with a diverse set of lexical and semantic-level supervision signals.
BLEURT models are trained in three steps: regular BERT pre-training pre-training on synthetic data, and fine-tuning on task-specific ratings (translation and/or data-to-text). According to this paperSellam et al. ((2020)), pre-training makes BLEURT significantly more robust to quality drifts and can quickly adapt to new tasks than BERT.

1.1.12. BARTScore

BARTScore, based on BART model, introduced in 2021, stands for Bidirectional and AutoRegressive Transformers Score. Weizhe Yuan Yuan et al. ((2021)), argues for a formulation of the evaluation of generated text as a text generation problem, directly evaluating text through the lens of its probability of being generated from or generating

other textual inputs and outputs.

The generative formulation of BARTSCORE makes it relatively easy to incorporate these insights here as well; we name this variant BARTSCORE-PROMPT In this paper Yuan et al. ((2021)), BARTSCORE empirically demonstrated its efficacy. Without the supervision of human judgments, BARTSCORE can effectively evaluate texts from 7 perspectives (Informativeness, Relevance, Fluency, Coherence, Factuality, Semantic Coverage, and Adequacy) and achieve a good performance against existing top-scoring metrics.

1.1.13. Bidemensional Leaderbords

Efforts to improve generation models tend to depend on simple n-gram overlap metrics (e.g., BLEU, ROUGE). This paper Kasai et al. ((2021)) argues that new advances in models and metrics should each more directly benefit and inform the other. They, therefore, propose a generalization of leaderboards, bidimensional leaderboards (Billboards), that simultaneously track progress in language generation models and metrics for their evaluation. Unlike conventional unidimensional leaderboards that sort submitted systems by predetermined metrics, a Billboard accepts both generators and evaluation metrics as competing entries. A Billboard automatically creates an ensemble metric that selects and linearly combines a few metrics based on a global analysis across generators. Further, metrics are ranked based on their correlation with human judgments. They release four Billboards for machine translation, summarization, and image captioning. They demonstrate that a linear ensemble of a few diverse metrics sometimes substantially outperforms existing metrics in isolation. Their mixed-effects model analysis shows that most automatic metrics, especially the reference-based ones, overrate machine over a human generation, demonstrating the importance of updating metrics as generation models become stronger (and perhaps more similar to humans) in the future.

1.1.14. MENLI

Recently proposed BERT-based evaluation metrics perform well on standard evaluation benchmarks but are vulnerable to adversarial attacks, e.g., relating to factual errors. This paper argues Chen and Eger ((2022)) that this stems (in part) from the fact that they are models of semantic similarity. In contrast, they develop evaluation metrics based on Natural Language Inference (NLI), which they deem more appropriate modelling. They design a preference-based adversarial attack framework and show that their NLI-based metrics are much more robust to the attacks than the recent BERT-based metrics. On standard benchmarks, their NLI-based metrics outperform existing summarization metrics

but perform below SOTA MT metrics. However, when they combine existing metrics with the NLI metrics, they obtain both higher adversarial robustness (+20% to +30%) and higher quality metrics as measured on standard benchmarks (+5% to +25%).

1.1.15. T5Score

Modern embedding-based metrics for evaluation of generated text generally fall into one of two paradigms: discriminative metrics that are trained to directly predict which outputs are of higher quality according to supervised human annotations, and generative metrics that are trained to evaluate text based on the probabilities of a generative model. Both have their advantages; discriminative metrics can directly optimize for the problem of distinguishing between good and bad outputs, while generative metrics can be trained using abundant raw text. In this paper Qin et al. ((2022)), they present a framework that combines the best of both worlds, using both supervised and unsupervised signals from whatever data they have available. They operationalize this idea by training T5S CORE, a metric that uses these training signals. They perform an extensive empirical comparison with other existing metrics on 5 datasets, 19 languages and 280 systems, demonstrating the utility of our method. 2 Experimental results show that: T5S CORE achieves the best performance on all datasets against existing top-scoring metrics at the segment level.

1.1.16. DiscoScore

Recently, there has been a growing interest in designing text generation systems from a discourse coherence perspective, e.g., modelling the interdependence between sentences. Still, recent BERT-based evaluation metrics are weak in recognizing coherence and thus are not reliable in a way to spot the discourse-level improvements of those text generation systems. DiscoScore was introduced in 2022, a parametrized discourse metric, which uses BERT to model discourse coherence from different perspectives, driven by Centering theory. Their experiments encompass 16 non-discourse and discourse metrics, including DiscoScore and popular coherence models, evaluated on summarization and document-level machine translation. They find that:

- the majority of BERT-based metrics correlate much worse with human-rated coherence than early discourse metrics, invented a decade ago.
- The recent state-of-the-art BARTScore is weak when operated at the system level – which is particularly problematic as systems are typically compared in this manner.

DiscoScore, in contrast, achieves strong system-level correlation with human ratings,

not only in coherence but also in factual consistency and other aspects, and surpasses BARTScore by over 10 correlation points on average. Further, aiming to understand DiscoScore, they provide justifications for the importance of discourse coherence for evaluation metrics and explain the superiority of one variant over another.

1.1.17. GPTScore

Generative Artificial Intelligence (AI) has enabled the development of sophisticated models that are capable of producing high-calibre text, images, and other outputs through the utilization of large pre-trained models. Nevertheless, assessing the quality of the generation is an even more arduous task than the generation itself, and this issue has not been given adequate consideration recently. This paper Fu et al. ((2023)), proposes a novel evaluation framework, GPTScore, which utilizes the emergent abilities (e.g., zero-shot instruction) of generative pre-trained models to score generated texts. There are 19 pre-trained models explored in this paper, ranging in size from 80M (e.g., FLAN-T5-small) to 175B (e.g., GPT3). Experimental results on four text generation tasks, 22 evaluation aspects, and corresponding 37 datasets demonstrate that this approach can effectively allow us to achieve what one desires to evaluate for texts simply by natural language instructions. This nature helps us overcome several long-standing challenges in text evaluation–how to achieve customized, multi-faceted evaluation without the need for annotated samples.

1.2. Metrics Comparison

In this section, I compare the previously exposed metrics to highlight their different strengths, weaknesses and their differences. They can be classified into 3 categories depending on the technique used for calculation:

- Basic algorithm
- Advanced algorithm
- Benchmark

1.2.1. Similarity Measure

To observe the similarity between the generated and reference text, various techniques can be used. Throughout history, they evolved in more and more complex ways.

BLEU focuses on n-gram precision, considering the overlap of n-gram sequences. In sum, BLEU captures the adequacy and fluency of translations, while ROUGE was designed to match the semantic coverage metric.
In addition to them, METEOR, TER, NIST MT, chrF, and MoverScore typically compare the generated output against one or more reference texts using various algorithms that analyze overlapping n-grams, lexical similarities, or alignment-based measures.

To get into details, BLEU measures the similarity by computing the precision of matching n-grams and then tends to focus on precision and penalizes overly short translations. Whereas the NIST MT metric is similar to BLEU, it places more emphasis on longer n-grams using a brevity penalty to account for length differences.
ROUGE calculates the overlap between the texts using various measures, including n-gram co-occurrence statistics (ROUGE-N), longest common subsequence (ROUGE-L), and skip-bigram statistics (ROUGE-S). ROUGE focuses on the recall of important content in the texts.
METEOR considers both precision and recall by comparing n-grams. The measure incorporates stemming, synonymy, and paraphrase matching to capture semantic similarities between the candidate and references.
TER evaluates the quality by quantifying the number of edits required to transform the output into the reference, considering additions, deletions, substitutions, and word order changes. TER aims to capture fluency and adequacy.
chrF is a metric that operates at the character level instead of the word level. It computes the F-score of matching character n-grams so it is robust to morphological differences, word order variations for example.
MoverScore evaluates the quality of generated sentences by measuring their structural differences. It calculates the earth mover's distance, which quantifies the minimum effort required to transform one distribution into another based on the word mover's distance algorithm.

On the other hand, metrics like BERTScore, BLEURT, BARTScore, T5Score, DiscoScore, and GPTScore leverage advanced similarity measures based on contextual embeddings, pre-trained language models, or semantic matching techniques.
These metrics go beyond exact lexical matches and consider the semantic and contextual understanding of the generated text, enabling a more nuanced and comprehensive evaluation of language generation and understanding tasks.
To get into details, BERTScore is a metric that measures the similarity by computing contextual embeddings using BERT (Bidirectional Encoder Representations from Trans-

formers). It considers both precision and recall of the embeddings and aligns them to calculate a similarity score (semantic and syntactic similarities).
BLEURT (Bilingual Evaluation Understudy with Representations from Transformers) is a metric that combines the strengths of pre-trained language models with traditional evaluation methods. BLEURT is trained to predict human-assigned scores for sentence pairs using a ranking-based approach.
BARTScore is a metric that calculates the similarity by employing the BART model to generate paraphrases and computes the probability of the reference being generated by the model. It focuses on semantic similarity and fluency.
T5Score: T5Score is a metric that evaluates the quality of text generation models using the T5 model, which is a text-to-text transformer model, to estimate the conditional probabilities. T5Score measures the fluency and adequacy of the generated text.
DiscoScore is a metric designed to evaluate dialogue generation systems. It measures the quality of generated dialogues based on their relevance, coherence, and informativeness. DiscoScore combines both automatic evaluation and human evaluation to provide a comprehensive assessment of dialogue systems.
GPTScore is a metric specifically developed for evaluating text generation models based on the GPT (Generative Pre-trained Transformer) architecture. It measures the quality of the generated output by comparing it to reference texts and capturing the likelihood of the reference given the output. GPTScore takes into account both fluency and coherence in the generated text.

Finally, unlike the others, some metrics like GLUE, SuperGLUE, Bidimensional Leaderboards, and MENLI provide standardized evaluation frameworks and tasks to assess performance. In the context of natural language processing (NLP) or machine learning, a benchmark typically consists of a set of tasks, datasets, and evaluation metrics.
While GLUE and its evolution SuperGLUE focus more on sentiment analysis, question answering, and textual entailment, MENLI determines the logical relationship between pairs of sentences in multiple languages, including tasks such as contradiction, entailment, and neutral. Bidimensional Leaderboards are evaluation frameworks that incorporate both task-specific metrics and language modelling scores.

1.2.2. Reference Sensitivity

Reference sensitivity refers to the degree to which a metric or evaluation method for natural language processing tasks is influenced by the choice of reference or gold standard. In the case of BLEU, ROUGE, METEOR, TER (Translation Edit Rate), NIST MT (NIST

Machine Translation), chrF (character n-gram F-score), and MoverScore, these metrics are generally considered to be reference sensitive. This means that their results can vary depending on the specific reference or gold standard used for comparison.
On the other hand, metrics like BERTScore, GLUE, SuperGLUE, BLEURT, BARTScore, Bidimensional Leaderboards, MENLI (Multi-perspective Natural Language Inference), T5Score, DiscoScore, and GPTScore are designed to be less sensitive to the reference choice. These metrics aim to capture the quality or similarity of generated text based on other factors beyond direct reference matching. By considering various linguistic aspects or leveraging pre-trained language models, these metrics provide a more comprehensive evaluation of language understanding and generation tasks, taking into account contextual and semantic information.

1.2.3. Model-dependence

Model dependency, i.e., reliance on a metric on specific pre-trained models or resources, is important because it reflects the Contextual Understanding and Adaptability of a metric. BLEU, METEOR, and ROUGE are generally model-agnostic and do not rely on specific pre-trained models. They can be applied to evaluate the text generated by various models without being tightly coupled to a particular architecture.
In contrast, BERTScore, BLEURT, and BARTScore are more dependent on transformer-based models. This model dependence allows them to capture more fine-grained semantic information and context-aware similarity.
However, it also means that the availability and quality of the pre-trained models can impact the performance and applicability of these metrics. The choice between model-agnostic and model-dependent metrics should be made based on the specific needs and resources available for evaluation.

Model dependence refers to the extent to which the performance of evaluation metrics in natural language processing tasks relies on the specific models used.
Metrics such as BLEU, ROUGE, METEOR, TER, NIST MT, chrF, and MoverScore are generally considered to be model-dependent. These metrics heavily rely on the quality and characteristics of the translation or generation models being evaluated. Different models may produce varying outputs, resulting in different scores from these metrics.
On the other hand, metrics like BERTScore, GLUE, SuperGLUE, BLEURT, BARTScore, Bidimensional Leaderboards, MENLI (Multi-perspective Natural Language Inference), T5Score, DiscoScore, and GPTScore aim to reduce model dependence. They leverage pre-trained language models, contextual embeddings, or multi-task learning approaches to provide more robust and generalizable evaluations. By considering richer linguistic

representations and broader contextual information, these metrics strive to be less reliant on specific models and capture the overall quality and understanding of the generated text more effectively.

1.2.4. Human assessment correlation

While all these metrics aim to provide automated evaluation measures, their correlation with human evaluations can vary.
Since BLEU has been found to have a limited correlation with human judgments, as it primarily focuses on n-gram precision and may not capture the overall quality and coherence of generated text, researchers tried to improve this correlation.
All previous metrics have been extensively studied and have shown varying degrees of correlation with human judgments. These metrics aim to capture different aspects of translation quality, summary coherence, or language understanding, and their scores are often compared to human assessments to measure their effectiveness.
Newer metrics strive to improve the correlation with human assessments by leveraging pre-trained language models, semantic matching, or incorporating multiple perspectives, aiming to provide a more reliable and accurate estimation of human judgment. However, human assessors' expertise remains crucial in evaluating and improving natural language processing systems.

1.3. Conclusion

In conclusion, the field of Natural Language Processing offers several metrics for evaluating various aspects of text generation, summarization, translation, and other NLP tasks. Metrics have evolved and diversified through time.
BLEU, ROUGE, METEOR, TER, NIST MT, chrF, GLUE, BERTScore, MoverScore, SuperGLUE, BLEURT, BARTScore, Bidimensional Leaderboards, MENLI, T5Score, DiscoScore and GPTScore provide valuable tools for automated evaluation, each with its strengths and considerations.

These metrics differ in their underlying approaches and the dimensions they capture. When comparing these metrics, it is important to consider factors such as similarity measure, reference sensitivity, model dependence, and human correlation. These considerations help in selecting the appropriate metric based on the specific evaluation requirements and characteristics of the NLP task at hand.

It is worth noting that no single metric can fully capture all aspects of text quality, and

a combination of metrics or careful consideration of their strengths and limitations is often recommended. Moreover, all metrics have variants that are more specific catching the essence of a task. Checking fine-tuned metrics is an important step when it comes to choosing one. The choice of metric should align with the task objectives, available resources, and the desired evaluation granularity. Thanks to this preliminary study and to capture the best insight into our measures, two metrics are selected for the thesis. Both metrics are complementary and add a different aspect and meaning to the analysis.

- Basic metric: **ROUGE-Lsum**. ROUGE is one of the basic metrics with the best result in human correlation, way better than BLEU. ROUGE-LSUM is considered to be a more accurate measure than ROUGE-1, ROUGE-2 or ROUGE-L. It is less sensitive to changes in the order of the words in the summary.
- Advanced algorithm: **BERTScore**. It is a more robust measure of semantic similarity than ROUGE. BERTScore has also been shown to be more robust to changes in the length of the summary and the number of references. I also have the option of using the precision, recall or f1 score. Precision measures how accurate the model is when it predicts that an instance is positive. Recall measures how complete the model is when it predicts that an instance is positive. I choose the F1 score because it is a harmonic mean of precision and recall. It will capture more globally the result than the two others. Moreover, I stated before that the main limits of the best score metric are especially for challenging adversarial examples and the quality of the BERT pre-train model. In my case, the dataset I use is not constructed to be highly adversarial. That is to say, there are no carefully crafted questions to cause the machine learning model to make a wrong prediction; there are no high perturbations. Secondly, the BERT model I choose is well-documented and polyvalent. I can adapt the complexity by choosing a Roberta-large model or a distilbert-base-uncased model.

It is important to keep an eye on a more basic metric because BERTScore is sensitive to the quality of the reference summaries. If the reference summaries are not of high quality, then BERTScore may not be able to accurately measure the quality of the generated summary.

As the field of NLP continues to evolve, new metrics may emerge, and existing ones may be refined. Researchers and practitioners should stay updated on the latest developments in NLP evaluation metrics to make informed decisions and ensure reliable assessments of text generation and other NLP applications.

2 | Methodology

In this section, I describe the way I organised my master thesis. That is to say, my work management, communications with my supervisor but also my research design and the different phases of my work.

2.1. Methodology tools

During this Master Thesis (TFM), I use some methodology tools such as Trello for work management and Microsoft Teams for team communication.

2.1.1. Microsoft Teams

Microsoft Teams allows me to share files, collaborate, and stay connected with my supervisor.

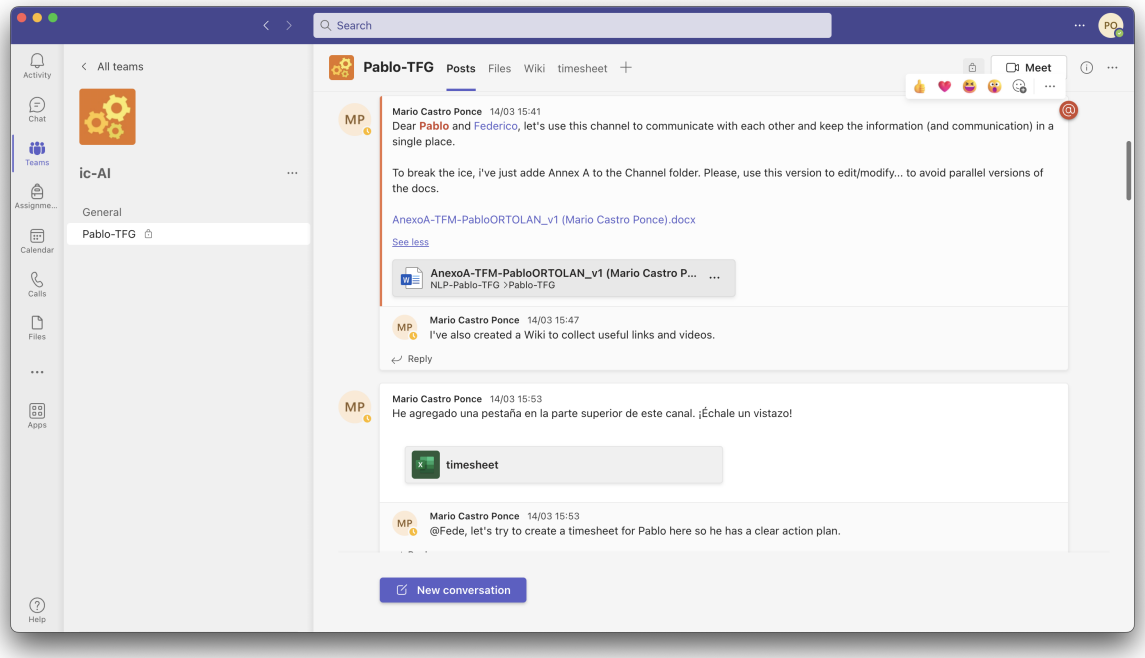


Figure 2.1: Screenshot of Microsoft Teams: posts with supervisors.

2.1.2. Kanban

Kanban is a popular project management and workflow management system that originated in Japan and was initially introduced by Toyota in the 1940s. The term "Kanban" is derived from two Japanese words: "kan," which means "visual," and "ban," which means "card" or "board." It is a visual method used to manage work and improve efficiency in various processes.

The Kanban system uses visual cues, often represented by cards or sticky notes, to represent tasks or units of work. These cards are placed on a Kanban board, which is a visual representation of the workflow. The board typically consists of columns that represent different stages of the process, such as "To Do," "In Progress," and "Done." As work progresses, the cards move from one column to the next, indicating their status.

The main principles of Kanban include:

- Visualizing the workflow: All work items are visualized on the Kanban board, providing a clear and transparent view of the entire process.
- Limiting work in progress (WIP): Kanban emphasizes setting WIP limits for each stage of the process to prevent overloading the team and maintain a smooth flow of work.
- Pull-based system: Work is pulled into the next stage only when the previous stage has the capacity, as indicated by available WIP slots.
- Continuous improvement: Kanban encourages teams to regularly review and improve their processes to optimize efficiency and effectiveness.

Kanban is widely used in software development, manufacturing, project management, and various other industries. It promotes a flexible and adaptive approach to managing work, making it easier to accommodate changes and focus on continuous delivery and improvement. As a result, it has become a valuable tool for students seeking to increase productivity during their master's thesis. For example, T. Boronat and his team dedicated a whole paper to this application. Boronat et al. ((2017))

Trello is a Kanban methodology tool that provides a visual platform for organizing tasks, and projects, making it easier to track progress and manage work efficiently. I use it by dividing my tasks into three categories such as "to do", "doing" and "done". All tasks of my projects are available here: `https://trello.com/`

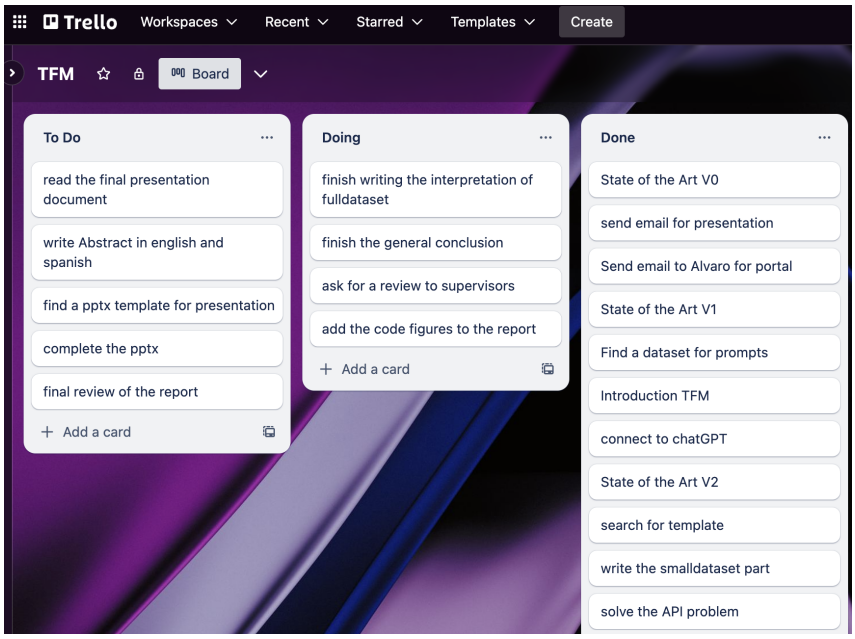


Figure 2.2: Screenshot of an example of Kanban tools: Trello.

As you can see above, this tool is very useful to organise tasks and keep a project organised. This tool is even more efficient when assigning tasks to colleagues and working in a team.

2.2. Schedule

Concerning my Master Thesis schedule, the process was as follows:

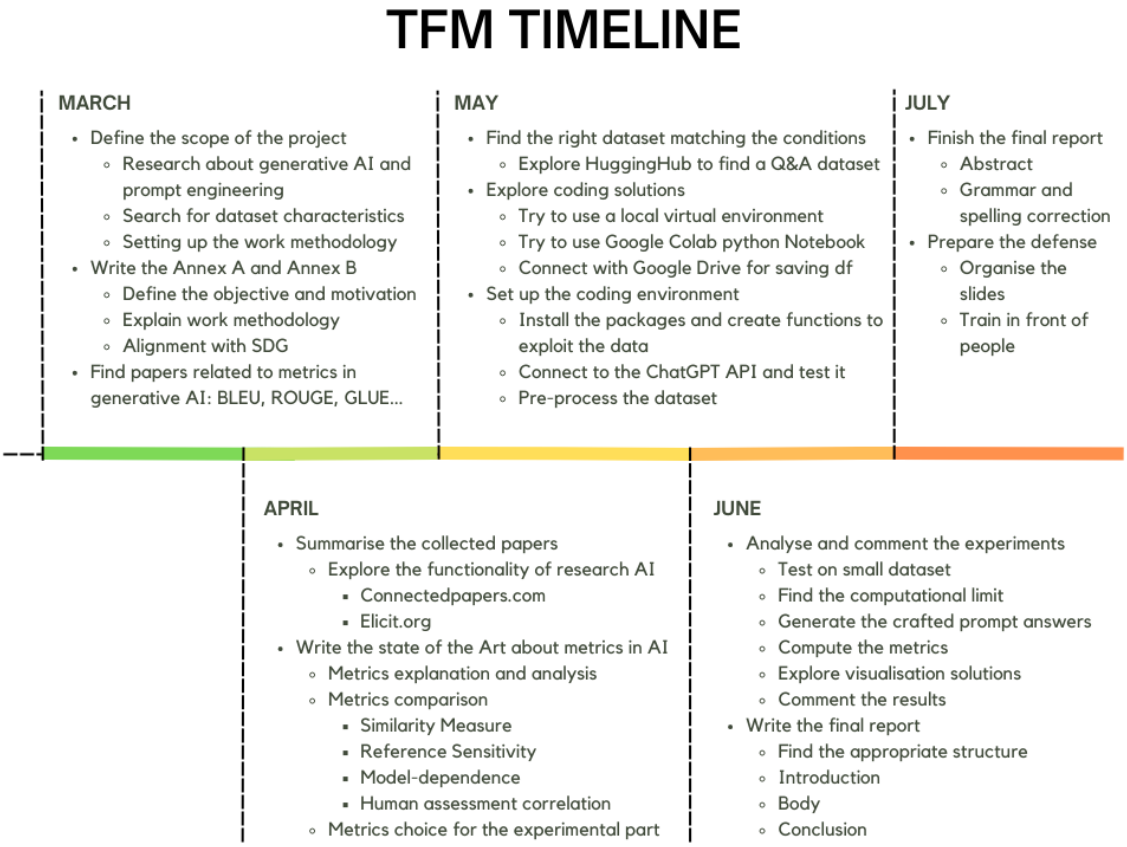


Figure 2.3: TFM timeline throughout the semester

My supervisor and I are communicating during weekly meetings for major topics and by teams for minor ones.

2.3. Coding

During my Master Thesis and for all the experiments, I use Google Colab. Google Colab is a cloud-based Jupyter Notebook environment that runs in my web browser. It allows me to write in Markdown and execute code in Python, as well as other programming languages, and share my work with others, like my supervisors.
Moreover, Colab is a great way to experiment with machine learning because it provides access to free GPUs, which can speed up machine learning operations and analysis.

Of course, there are also some downsides to using Google Colab. For example, I am limited to the amount of storage space that I have in my Google Drive account. Additionally, if I am working on a sensitive project, I may not want to store my code and data in the cloud.

Here is a screenshot of the coding environment:

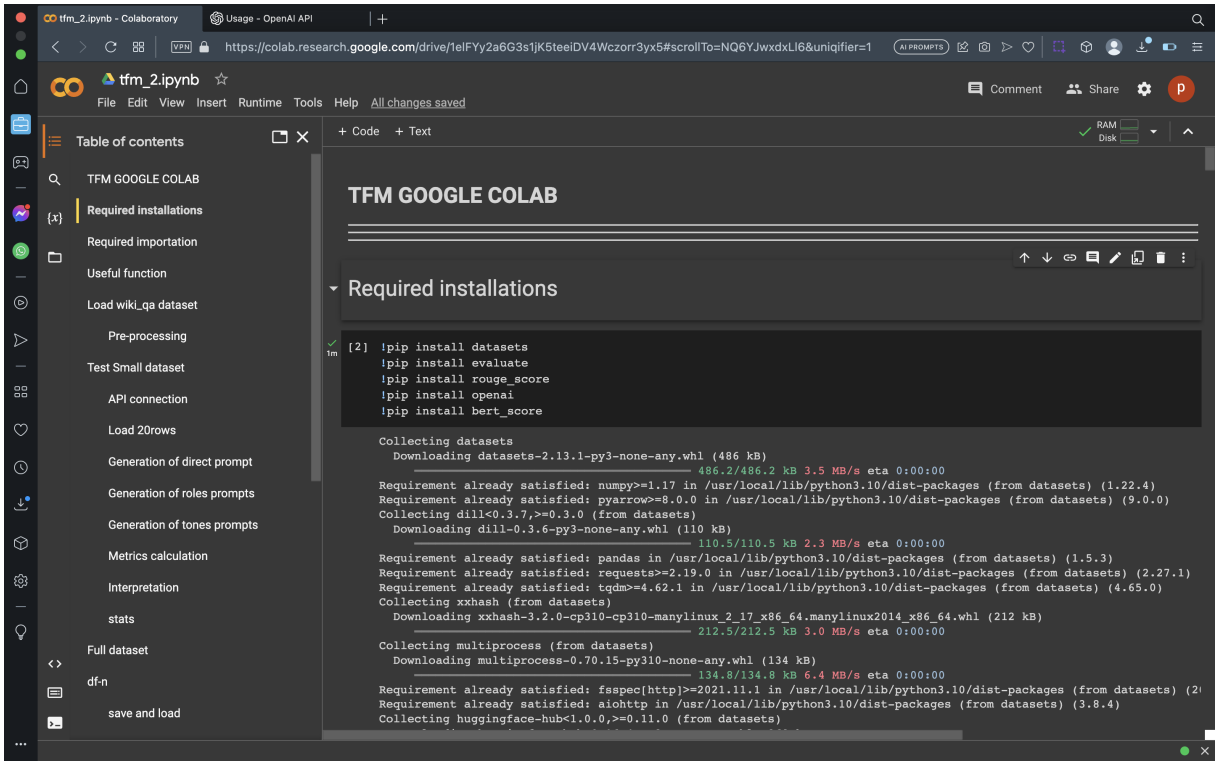


Figure 2.4: Screenshot of Google Colab

I use Google Colab for several reasons. First, for the Ease of use. I have a MacBook Pro M1 8GB and its architecture is very different from other PCs. Google Colab is a web-based platform, which means that I can access it from any OS and any device with an internet connection. This makes it easy to collaborate with others on projects and to share my work. Moreover, everybody loads the notebook simply to run the required packages in the first cell to have the same results I have. There are no local installations to make, so it is easier for my supervisors.

Then, Google Colab provides free access to GPUs, which can significantly speed up the training and execution of machine learning models. For me, this is especially important for computing the metrics BERTScore which is based on the generation of a large model: RoBERTa-large. Thanks to that I can significantly reduce the computational time compared to a CPU.

I also found some limitations to the utilisation of Google Colab. Every session is limited in time. After 12 hours, the session close and all the variables are crushed. Fortunately, Google Colab offers the possibility to save data in Google Drive. I use this functionality to save datasets in CSV. However, the CSV format converts all data as string objects.

When loading again the dataset some columns have to be converted before usage.

You can find the final code I used in the annex, at the end of this paper.

3 | AI Model Capabilities

3.1. AI Model Capabilities Overview

AI generative models have a wide range of capabilities. They can be used to create realistic-looking images, generate text that is indistinguishable from human-written text, and even compose music that is similar to the work of famous composers.

AI generative models are still under development, but they have the potential to revolutionize the way we create and interact with data. They could be used to create new forms of art, generate new ideas, and even create new products and services.

In the next sections, I will talk about the specific capabilities of AI generative models.

Image generation

AI generative models can be used to generate realistic-looking images. This is done by training the model on a dataset of existing images. The model then learns to identify the patterns that are common in these images, and it can use this knowledge to generate new images that are similar to the images it was trained on.

Many fields benefit from this technology, some of them are described in the next paragraph.

DALL-E 2 DALL-E 2 is a large language model developed by OpenAI that can create realistic images from text descriptions. It is a successor to the original DALL-E model, which was released in 2021.

To use DALL-E 2, you simply need to provide a text description of the image you want to generate. For example, you could say "A photorealistic painting of a cat riding a unicycle on a rainbow." DALL-E 2 will then generate an image that matches your description. The image is so realistic that it is difficult to tell that it is not a real photograph. It could be used to create realistic images for marketing materials, to generate art, or to help people with visual impairments.

G. Marcus wrote a paper Marcus et al. ((2022)) about DALL-E 2 system. They intentionally prompted more challenging sentences than the typical ones. Only a few images fully satisfied their requests. This model is still under development, and it is, for sure, an AI to follow in the future.

GLIDE for medicine The model GLIDE (Guided Language to Image Diffusion for Generation and Editing) is a text-to-image generative artificial intelligence. GLIDE has rich representations, but medical applications of this model have not been systematically explored. J. Kather, Narmin Ghaffari Laleh, S. Foersch, D. Truhn Kather et al. ((2022)) used this model for reasonably strong representations of key topics in cancer research and oncology, in particular the general style of histopathology images and multiple facets of diseases, pathological processes and laboratory assays. They found that domain-agnostic generative AI models can learn relevant medical concepts without explicit training and it can be better with additional domain-specific fine-tuning.

Text generation

AI generative models can be used to generate text that is indistinguishable from human-written text. This is done by training the model on a dataset of existing text. The model then learns to identify the patterns that are common in this text, and it can use this knowledge to generate new text that is similar to the text it was trained on.

For example, Deep Poetry is a text generation field described in this paper Xie et al. ((2017)). Text generation is a foundational task in natural language processing. However, most of this work has focused on generating prose. We investigate whether deep learning systems can be used to synthesize poetry, in particular Shakespearean-styled works. The model developed in this paper produces sonnets that are generated by complex models that have a coherent meaning and relatively correct meter without blatantly copying Shakespeare's original works. Here is an example: *For that deep wound it gives I loved me not; That I might see me, than now are thief; That which it might me, and words old tell That I are one with my friend?*

Music Generation

AI generative models can be used to compose music that is similar to the work of famous composers. This is done by training the model on a dataset of existing music. The model then learns to identify the patterns that are common in this music, and it can use this knowledge to generate new music that is similar to the music it was trained on.

AI generative models are a powerful tool that has the potential to revolutionize the way we create and interact with data. They are still under development, but they have already made significant progress. As they continue to develop, they will become even more powerful, versatile and help humans to compose.

For example, the paper Huang et al. ((2020)) presents findings on what 13 musician and developer teams needed when co-creating a song with AI. These findings reflect a need to design machine learning-powered music interfaces that are more decomposable, steerable, interpretable, and adaptive, which in return will enable artists to more effectively explore how AI can extend their expression.

3.2. AI model limits

Although Generative AI can provide accurate and helpful responses, there are still some common errors that can occur. In the next section, I described the main limitations concerning generative AI.

3.2.1. Misunderstanding the Context

AI models, including text-generative AI, often suffer from the limitation of context misunderstanding, resulting in responses that are irrelevant or inaccurate. This is because these models lack true comprehension and semantic understanding of language. While they can generate plausible sentences based on patterns learned from the training data, they might fail to capture the broader context of a question.

For example, in conversational AI, context plays a crucial role in maintaining coherent and meaningful dialogue. When a model lacks context awareness, it may respond incorrectly or inappropriately, frustrating users and limiting the usability of the AI system.

Research is ongoing to improve context understanding in AI models. Techniques such as memory-augmented neural networks and attention mechanisms have shown promising results in capturing long-range dependencies and contextual information. The papers by Graves Graves et al. ((2014)) and by Vaswani Vaswani et al. ((2017)) are notable examples of work addressing context understanding in AI models.

3.2.2. Bias in Language and Data

AI models, including text-generative AI, are trained on vast amounts of data collected from various sources, which may contain inherent biases. Consequently, the AI models

tend to inherit these biases and may generate biased outputs in their responses. These biases can manifest in various forms, including gender, race, and cultural biases.

Addressing bias in AI is a challenging and crucial research area. Various approaches have been proposed to reduce bias in language models. For instance, the paper by Zhang Zhang et al. ((2018)) explores adversarial learning techniques to reduce bias in text generation. Moreover, the paper Sap et al. ((2019)) highlights the importance of tackling biases in AI models and the potential consequences of biased language generation.

To mitigate bias, it is essential to curate diverse and representative training data, implement fairness-aware training techniques, and conduct thorough evaluations to identify and rectify biases in AI-generated content.

3.2.3. Inaccurate Information

AI models, like text generative AI, generate responses based on patterns learned from their training data. Consequently, if the training data contains errors or inaccuracies, the AI model may produce incorrect information.

To address this limitation, researchers and practitioners emphasize the importance of high-quality training data. Bender and Friedman with Bender and Friedman ((2018)) introduce the concept of data statements to describe the data used for training NLP models, helping researchers and users understand potential limitations and biases in AI models.

In addition, approaches like knowledge distillation, where a more accurate model transfers knowledge to a less accurate one, can be employed to improve the accuracy of AI-generated information. The paper by Hinton Hinton et al. ((2015)) presents the knowledge distillation technique and its potential applications.

3.2.4. Uncertainty in Responses

AI models, including text-generative ones, might not always provide definitive answers. Instead, they may express uncertainty or provide a range of possible responses, which can be challenging for users seeking precise information.

One way to address uncertainty in AI responses is to explore approaches that improve the confidence estimation of AI models. Techniques like Bayesian neural networks, presented in the paper Pearce et al. ((2018)), can be employed to quantify the uncertainty in AI-generated responses.

Additionally, providing more context and refining questions can help AI models generate more accurate and confident responses. User studies, such as Al-Rfou et al. ((2016)), explore the impact of context in improving the quality of AI responses.

3.2.5. Technical Limitations

AI models, including text generative AI, have technical limitations based on their architecture, training data, and the scope of the problem they are designed to solve.

To address technical limitations, researchers continually explore advancements in AI architectures. For instance, Transformer-based models have shown significant improvements in various NLP tasks, including text generation. The original Vaswani et al. ((2017)) introduced the Transformer model, which has become the foundation for many state-of-the-art language models.

Additionally, research in multimodal learning, which combines information from multiple modalities such as text and images, can enhance the capabilities of AI models. The paper Lu et al. ((2019)) demonstrates the potential of multimodal pretraining for improving AI models' performance in understanding and generating both text and images.

However, users need to understand the limitations of AI models and ensure that the questions or prompts align with the model's technical capabilities. If a particular task exceeds the scope of the current AI model, users might need to explore other models or employ a combination of AI systems to achieve their desired results.

3.2.6. Conclusion

Acknowledging and addressing the limitations of AI models, particularly in text generative AI, are essential for responsible and effective use of AI technology. Ongoing research in the field focuses on enhancing context understanding, reducing biases, improving accuracy, and addressing uncertainty in AI responses. Researchers and developers play a crucial role in continuously advancing AI models to address these limitations and provide more reliable and useful AI-generated content.

By referencing academic papers and staying up-to-date with the latest research, users and practitioners can make informed decisions when interacting with AI models and contribute to the improvement of AI technology overall. Responsible use, coupled with advancements in AI research, can lead to AI systems that are more accurate, unbiased, and beneficial for various applications.

4 | Techniques for Prompt Optimization

Knowing how to write optimized AI prompts is a valuable and useful skill allowing one to get the expected result. There are several common techniques used in the field of prompt engineering to influence the behaviour and output of AI models.

In this chapter, I explore different techniques for improving prompt quality and analyse various modification strategies.

4.1. Instructional Prompts

Adding explicit instructions to the prompt can guide the AI model towards the desired behaviour or output. Instructions can specify the format, style, or specific information to be included in the generated response. For example, instructing the model to provide a step-by-step solution, adopt a formal tone, or consider a particular perspective.

4.2. Constraint-based Prompts

Introducing constraints in the prompt can restrict the model's response space, guiding it to generate outputs that adhere to specific requirements. These constraints can include word count limits, required keywords, or instructions to focus on a particular topic. Constraint-based prompts have been utilized in various AI applications to control the model's output.

4.3. Exemplar-based Prompts

Including example outputs in the prompt can provide the model with specific instances of the desired response. By presenting well-crafted exemplars, the model can be encouraged to generate responses that match the given examples in terms of structure, style, or content.

Exemplar-based prompts have been studied in text generation tasks, particularly in the context of neural language models.

4.4. Contextual Prompts

Incorporating relevant contextual information in the prompt can help the model generate responses that are sensitive to the specific context or situation. Contextual prompts provide the model with the necessary background information to produce more coherent and contextually appropriate outputs.

Contextual prompts have been widely explored in various natural language generation tasks. The paper by Tom B. Brown Brown et al. ((2020)) introduces GPT-3, a massive language model capable of adapting to different tasks with minimal context provided.

4.5. Priming Prompts

Priming involves providing a preliminary portion of the desired response as part of the prompt. By seeding the model with partial information, it can be guided to continue generating outputs that align with the given prompt context and initial response fragment.

4.6. Reformulation and Rewriting

Modifying the phrasing or structure of the prompt can influence the model's understanding and generate more accurate responses. By refining the wording, making it more explicit, or rephrasing ambiguous parts, prompt engineering can improve the model's comprehension and generation capabilities.

Reformulation and rewriting of prompts have been explored in various works related to natural language processing and text generation. The paper by Alec Radford Radford et al. ((2018)) introduces GPT, a seminal language model, which has inspired numerous subsequent works in prompt engineering and modification.

4.7. Conclusion

By understanding and applying these techniques for prompt optimization, users can effectively guide AI models to produce desired outputs. Each technique has its strengths and applications, and the choice of the most suitable technique depends on the specific task and desired outcomes. Researchers continue to explore new methods for prompt engineer-

ing to advance the capabilities of AI models and ensure more controlled, accurate, and contextually-aware responses. Leveraging prompt optimization techniques in conjunction with responsible AI practices can lead to more efficient and beneficial interactions with AI systems.

4.8. Thesis techniques

For the rest of the thesis, not all techniques are investigated. The objective is to help ordinary people to make more efficient requests. It means, that it has to be something easy and quick to do every time someone has to search. The more relevant technique is then the instructional prompt.
I will focus on assigning a role and a tone to the model to get the desired response.

The main two goals of the following experiments are to :

- Quantify the efficiency to add a role and a tone in a prompt.
- Identify the best roles and tones to add to a specific dataset.

5 | Experimental Results and Analysis

In this chapter, I present my experiments, their purpose and how I conducted them.

5.1. Presentation of the experiments

5.1.1. Purpose

The goal of this experiment is to find the best way to write a prompt to get the best answer. First, we have to define what is the best answer. In this thesis, the best answer is the closest generated answer to the human reference. I measure the distance between them using different metrics. Each one gives a different interpretation of the result: some emphasize the human correlation, while others can emphasise the similarity of the words used.

Then, the questions I am trying to optimise the answers are specific to one dataset. They are close to typical questions one another wants to ask a generative AI.

5.1.2. Hypothesis

Before doing any experiments here are the assumptions I am making.

- The human answers are considered the best answer possible to each prompt. I will not try to write again some of them that can be improved.
- The dataset used in this thesis is a good representative of the Q&A I want to optimize.

5.1.3. Methodology and technique

Dataset presentation

In this thesis, I needed a dataset that completed my requirements, that is to say:

- Prompts: the prompts have to be diverse and short, less than 30 words. Indeed, to cover the variety of prompts that a model can face in a real-life context, the prompts have to be non-subject specific. Moreover, the length is important for the computation, several answers have to be generated for each prompt.
- Human reference answers: the human-generated answers have to be high quality. Many datasets contain only a few words as answers. To measure the intrinsic capability to generate good answers, answers have to be developed. More than the keywords, the quality is in the details.

According to the above requirements I choose the wiki_qa dataset Yang et al. ((2015)): `https://huggingface.co/datasets/wiki_qa`

This dataset is a Question-Answering dataset, a collection of questions and corresponding answers in English based on Wikipedia content. It is composed of 5 fields and contains 20360 rows in the train part, 2733 rows in validation and 6165 rows in the test. For each question, they are multiple answers related and each one is classified as relevant or not. Here is a description of the fields:

- question_id (string): The ID of the question that was asked.
- question (string): The question that was asked.
- document_title (string): The title of the Wikipedia article that the question was asked about
- answer (string): The answer to the question.
- label (class label): Whether or not the answer is relevant to the question.

For this thesis, only relevant answers are kept. This dataset fulfils the requirements:

- Prompts: They are human-generated and diverse because they are extracted from Wikipedia, one of the largest and most popular general reference works available on the internet that covers a wide range of topics, including history, science, geography, arts, culture, technology, and more.
- Human reference answers: They are also human-generated because Wikipedia is collaborative. In this dataset, answers are not single words but sentences response which are relevant to our study.

Prompts variation

In this subsection, I present the different modifications the prompts undergo. These variants aim to measure, thanks to metrics, the effect of specifying a role and a tone thanks to a piece of a sentence.

Specifying the role At first, specifying the role in the prompt for ChatGPT helps set the context, improve the quality of responses, and make the conversation more engaging and tailored to the desired outcome. That is why, it is important to define what can be the best roles. Here is a list of the chosen ones with their explanation:

- Student: Taking on the role of a student allows one to ask questions, seek clarification, or request explanations on various topics. It's useful for educational purposes or to learn new information.
- Teacher: Assuming the role of a teacher enables one to seek guidance, instructional strategies, or educational resources. ChatGPT can provide insights from an educator's perspective and offer advice on teaching methods or classroom management.
- Expert: Using a professional role, such as a doctor, lawyer, engineer, or accountant, allows one to explore domain-specific knowledge and seek specialized advice. ChatGPT can provide information and insights relevant to the chosen profession.
- Child: By specifying the role of a child, the conversation is more engaging, educational, and relatable to younger audiences. It allows ChatGPT to provide responses that align with the developmental stage and understanding of a child, fostering a more meaningful and age-appropriate interaction.
- Writer: When looking for help with writing or brainstorming ideas, adopting the role of a creative writer can be beneficial. ChatGPT can assist with generating story plots, character development, or providing writing tips.

Specifying the tone At first, specifying the tone helps optimize the quality and style of the conversation, ensuring that ChatGPT's responses align with the intended communication goals and desired emotional impact. That is why, it is important to define what can be the best tones. Here is a list of the chosen ones with their explanation:

- Formal: A formal tone is suitable for professional settings, academic discussions, or responses that are polite, professional, and respectful. It is commonly used in business communications, official correspondence, or when seeking authoritative information.

- Friendly: A friendly tone creates a warm and approachable atmosphere. It is useful when engaging in casual conversations, seeking informal advice, or for responses that are conversational and personable. This tone is often used in social interactions, customer service, or when seeking a more relaxed conversation.
- Informative: An informative tone focuses on providing clear and concise information. It is helpful when seeking factual answers, and explanations, or for responses that are straightforward and knowledge-oriented. This tone is commonly used in educational contexts, Q&A sessions, or when you prioritize accuracy and clarity.
- Supportive: A supportive tone conveys empathy, understanding, and encouragement. It is beneficial when discussing personal or sensitive topics, seeking emotional support, or for responses that are compassionate and comforting. This tone is often used in counselling, mental health discussions, or when you need emotional reassurance.
- Humorous: A humorous tone adds wit, playfulness, and lightheartedness to the conversation. It is useful when engaging in creative writing, light-hearted discussions, or for responses that are entertaining and funny. This tone can make the conversation more enjoyable and engaging.
- Authoritative: An authoritative tone conveys expertise, confidence, and a sense of expertise. It is suitable when seeking guidance, instructions, or responses that are assertive and credible. This tone is often used in instructional contexts, technical discussions, or when you need a confident and knowledgeable response.

5.1.4. Dataset pre-processing

For the practical experiment, I need first to pre-process the dataset. The dataset is divided into three parts, 20360 rows in the train part, 2733 rows in validation and 6165 rows in the test part. Among these parts, I only need the ones where the answer is relevant to the question. Then after dropping the rows where the label is equal to 0, I have:

- Train: 1040 rows
- Test: 293 rows
- Validation: 140 rows
- Total: 1473 rows

I observe that there are multiple questions with different relevant answers. To have a unique reference for my calculation, I remove the duplicates and keep the first answer to

each question. Now, I have a total of 1242 rows.

At first, I don't know if I choose all rows or only part of them, it will depend on the computational cost, the time consumed for each generation, and the API availability.

To experiment with that, I begin with a small dataset, called *dfsmall*, with 20 rows.

Then, I will try with one part of the full dataset. To prepare that I unite the Train, Test and Validation dataset and will take the first n-lines.

5.1.5. API set up

To generate all the different answers, I need an API from ChatGPT. Thanks to that, I will be able to use the prompts and modify them with a piece of code.

After, getting back my API key from the open AI website (`https://platform.openai.com/account/api-keys`), I used it in my code but got an error *"RateLimitError: You exceeded your current quota, please check your plan and billing details."*

I can not continue my thesis freely. I have to upgrade my account to use this functionality. Sadly, no other platform has a similar free API. Here is the piece of code that connects to the API:

```
import openai
openai.api_key = "ab-**************"

completion = openai.ChatCompletion.create(model="gpt-3.5-turbo",
   messages=[{"role": "user", "content": "Give me 3 ideas for apps I
   could build with openai APIs "}])
print(completion.choices[0].message.content)
```

Then I construct a function that allows me to generate an answer through the API. The function takes three arguments: question, role, and tone. The question argument is the question that the user wants to be answered. The role argument is the role that the user wants to take on when answering the question. The tone argument is the tone that the user wants to use when answering the question.

The function creates a prompt that includes the role and tone arguments. For example, if the role argument is "teacher" and the tone argument is "informative", then the prompt would be "Answer as a teacher and with an informative tone: " + question.

The function then uses the OpenAI API to generate a response to the prompt. The response is returned as a string. Here is the code of the function:

```
def generate(question,role,tone):
```

```
  if role=='' and tone=='':
    prompt=question
  if role=='' and len(tone)>0:
    prompt = 'Answer with a '+tone+' tone: ' + question
  if tone=='' and len(role)>0:
    prompt = 'Answer as a '+role+': ' + question
  else:
    prompt = 'Answer as a '+role+' and with a '+tone+' tone: ' +
   question

  completion = openai.ChatCompletion.create(model="gpt-3.5-turbo",
   messages=[{"role": "user", "content": prompt}])
  return completion.choices[0].message.content

roles = ['direct','writer','student','teacher','expert','child']
tones = ['direct','formal','friendly','informative','supportive','
   humorous','authoritative']
```

As described below, a prompt example could be: "Answer as a teacher, how to cook vegetables?" or "Answer with a formal tone, how to cook vegetables".

In the next section, the experimental part will be presented.

5.2. Experimental manipulation

5.2.1. Test on small dataset

There are a few reasons why I use a small piece of the dataset for analysis before the full one.

- To identify any errors or problems with your data.
- To choose the right analytical tools.
- To understand the data better. When I analyze a small dataset, I can take more time and understand the data.

In the following sections, I will use the 20 rows.

Required installations

Here are the installations I need to run my program: The implementation in Python is shown below:

```
    !pip install datasets
```

```
!pip install evaluate
!pip install rouge\_score
!pip install openai
!pip install pandas
```

Here is a description of their function:

- datasets: This package provides a collection of datasets for natural language processing tasks. It includes datasets for text classification, text summarization, question answering, and more.
- evaluate: This package provides a collection of evaluation metrics for natural language processing tasks like rouge, bleu...
- rouge_score: This package provides an implementation of the ROUGE score, which is a metric for evaluating the quality of text summarization.
- openai: This package provides access to the OpenAI API, which allows you to use OpenAI's large language models, such as GPT-3.
- pandas: Pandas DataFrames are a powerful data structure for representing and manipulating tabular data. They are similar to spreadsheets, but they are more efficient and flexible.

Dataset preparation

First, I take the united dataset described in the Dataset pre-processing part. It is containing only 4 columns:

id	question_id	question	answer
12	Q11	how big is bmc ...	Employing over 6,000...
15	Q48	how long was...	The black-and-white...
...	...	...	...

Table 5.1: Initial Dataset

Here is their description:

- id: this integer corresponds to the initial position of the row in the dataset
- question_id: this string is unique and attached to a question.

- question: this string contains the question.
- answer: this string contains the human answer.

Generation of the answers

From this, I can generate the answers I need for the metrics calculation. I will generate 12 answers, stored in strings. One for the direct answer, the API response to the brute question. Then 5 answers for the 5 added roles: 'student', 'teacher', 'expert', 'child', 'writer' And 6 answers for the 6 tones added: 'formal', 'friendly', 'informative', 'supportive', 'humorous', 'authoritative'.

My dataset now has 16 columns: question_id, question, answer, direct_answer, writer_answer, student_answer, teacher_answer, expert_answer, child_answer, formal_answer, friendly_answer, informative_answer, supportive_answer, humorous_answer, authoritative_answer

Metrics calculation

From this, I can compute the metrics. I use the evaluate-metric functionality of huggingface. Snippets of code has to be implemented in Python to load the metric and use it. The website (`https://huggingface.co/evaluate-metric` also provides documentation on how to use these metrics.

As said before, I used the metric rouge-Lsum and the metric BERTScore. One will capture more of the words precision used and the other will highlight semantically-correct phrases.

I calculate each rouge-Lsum score and bertscore for each row of my 12 answers columns. I store this in 4 columns: rouge_role, bert_role, rouge_tone, bert_tone. Each column is a float list in which the first float is corresponding to the direct_answer score, the second one to writer_answer score etc.

First interpretation

The objective of this subsection is not to have strong results and conclusions. I want to confirm that the data is workable and explore one or two possible visualisations.

So, now that I have all the metrics calculated, I can tell for each row what role and tone is giving the highest rouge score and bertscore.

For each row, in the 4 metric columns and using the max function, I identify the index where the score is the highest. If Rouge and Bert score agree on the best role or tone.

Then the counter of this role/tone is 1 up. If they disagree, I will note them but take into account the bertscore.

I create 4 new columns: rouge_index_role, bert_index_role, rouge_index_tone, bert_index_tone where the integer corresponds to the index of the highest score in the column. That is to say, for the role part:

- 0: direct_answer
- 1: writer_answer
- 2: student_answer
- 3: teacher_answer
- 4: expert_answer
- 5: child_answer

And for the tone part:

- 0: direct_answer
- 1: formal_answer
- 2: friendly_answer
- 3: informative_answer
- 4: supportive_answer
- 5: humorous_answer
- 6: authoritative_answer

Quick analysis Visually, for the small dataset of 20 rows, here is the result. In abscissa, I have the row number in the dataset and the best score index in ordinate, for both metrics.

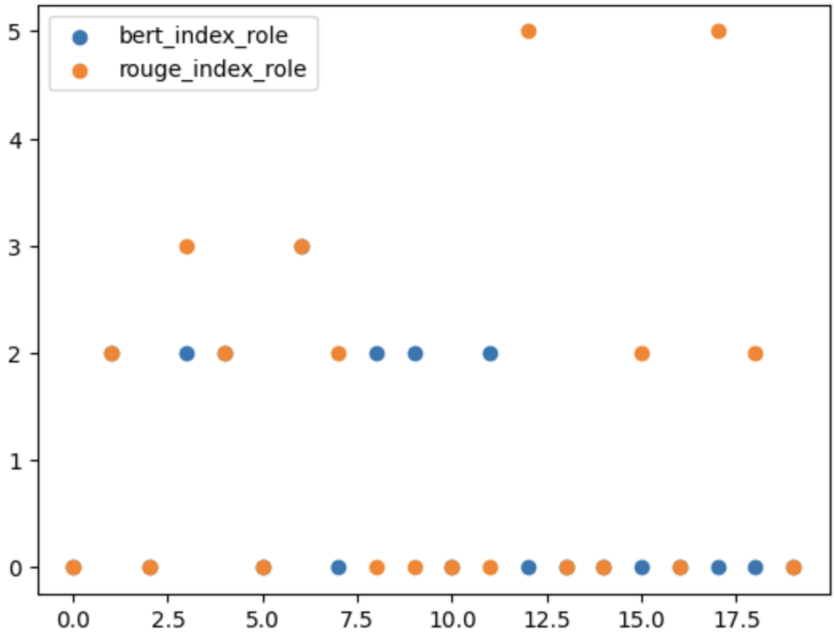


Figure 5.1: Index of the best role answers for each row.

As shown above, when considering only the role, 9 times out of 20, rouge score and bertscore agree on the best answers.

When they agree, the best role to adopt is the numbers 0, 2 and 3. These numbers are associated with the direct answer, the student's answer and the teacher's answer.

When they disagree, it is between rougescore first and bertscore, 2-3, 0-2 (x3), 2-0 (x3), 0-5. I can observe that the index 0 is often involved.

At first sight, I can tell, not adding any role is often a good thing. Nevertheless, for some questions, better results come from role-added prompts. For the 20 first questions of the wiki_qa dataset, specifying the role of a teacher or a student is providing good results, but globally not as much as not crafting the prompt. I will investigate in the next part how better it can be.

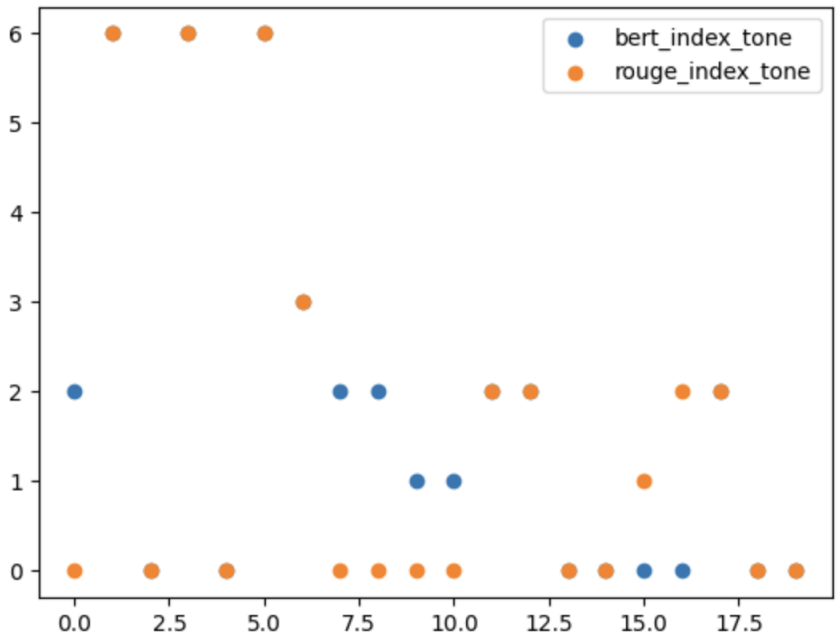


Figure 5.2: Index of the best tone answers for each row.

Considering the tone, with only 20 rows, we can observe that 0,2 and 6 are more often the best index. They are associated with the direct answer, friendly tone and the authoritative one. The data looks workable so I will not search very deeply for now. A more detailed analysis will be done with more data in the next section.

Detailed analysis This first analysis is not deep enough because I can't compare the different gaps and measure the profit of using a crafted prompt. Let's see another visualisation: the heatmap.

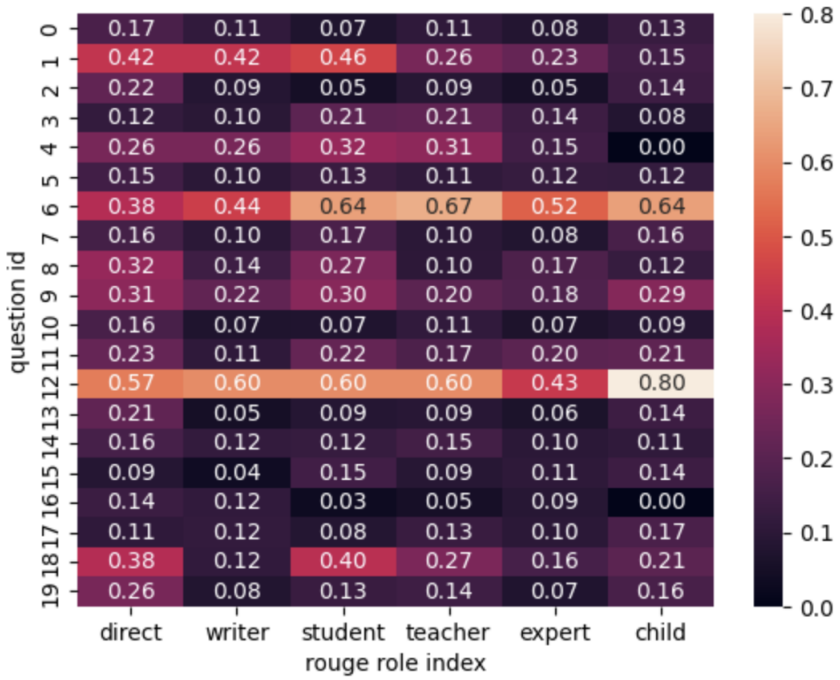


Figure 5.3: Role heatmap with rouge metric.

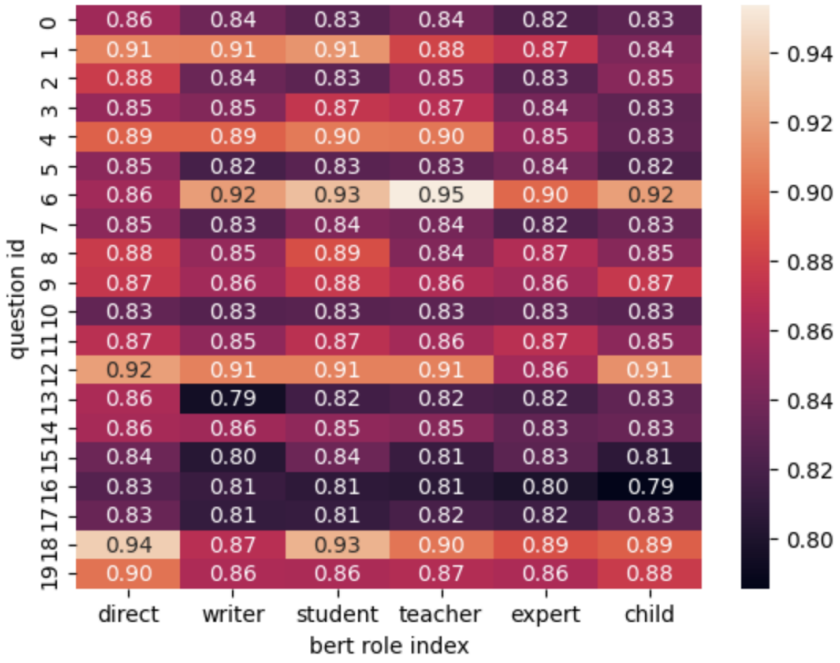


Figure 5.4: Role heatmap with Bert metric.

In this first two figures, there are some interesting things to say. The rows where the colour seems to be identical means that there are no significant improvements with the

craft prompts. Nevertheless, the score in the heatmap helps us to tell which role is better in each row.

The interesting rows are:

- row 0: the rouge score is very low, the direct answer is a bit higher but the Bert score is greater than 0.83. It means that the model did not succeed to use the same words as in the human answer. Nonetheless, the meaning is preserved because the Bert score is high. In this case, not specifying a role is slightly better.
- row 1: rouge says that the student role is the best by 0.04 and Bert says it is equal for direct, writer and student. In this case, the first three roles are a real improvement, student one in particular, in comparison with the last 3 roles.
- row 6: rouge and Bert agree that the teacher role is better than the rest.
- row 12: rouge says that child role is far better with 0.8 (0.6 for the seconds). But Bert says that all roles are similar except the expert one. This means that the child answer correlated well with the reference using basic 'child' words. Bertscore is altered because in that case, the reference is not high quality.

Overall, the direct answer is performing well and with no insight about the question it is better to not ask for any role.

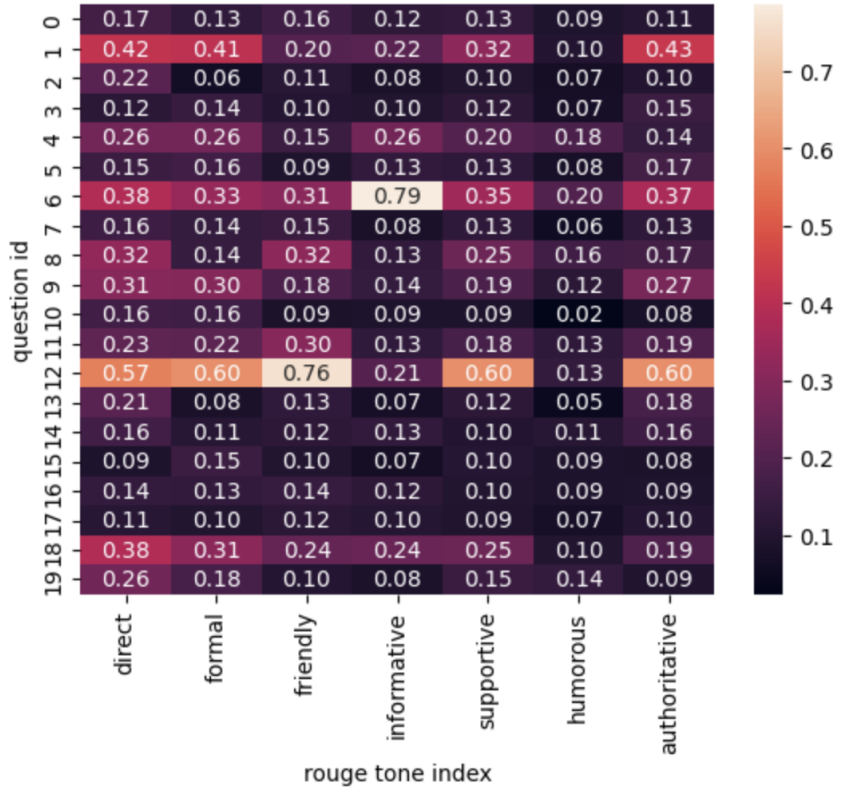


Figure 5.5: Tone heatmap with rouge metric.

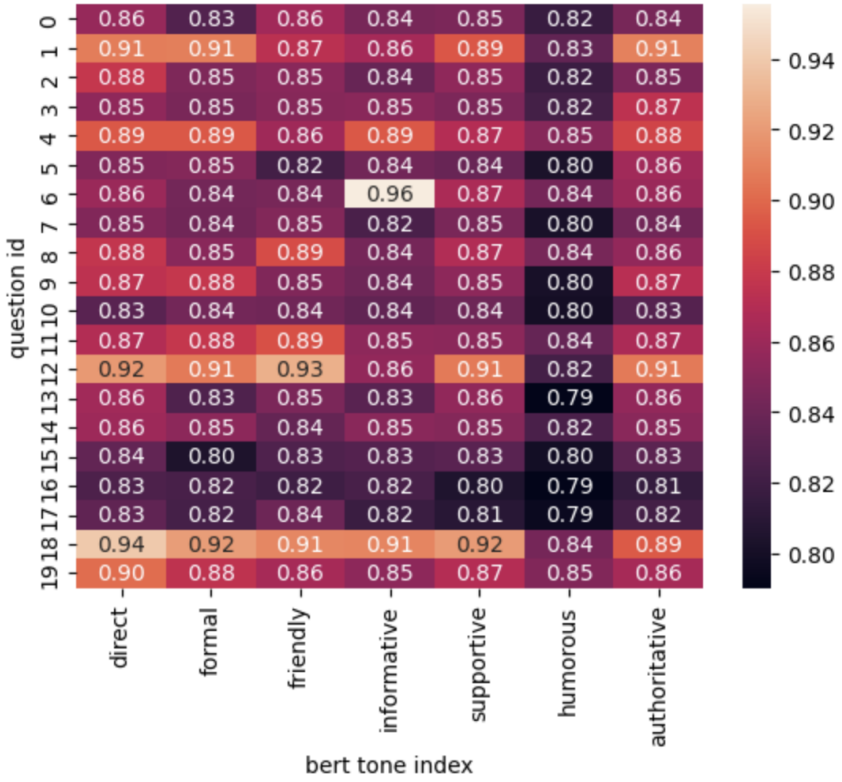


Figure 5.6: Tone heatmap with Bert metric.

The same interesting rows are visible for the tone analysis. This suggests that some questions can be more manoeuvrable than others.

After seeing the details, let's see the average:

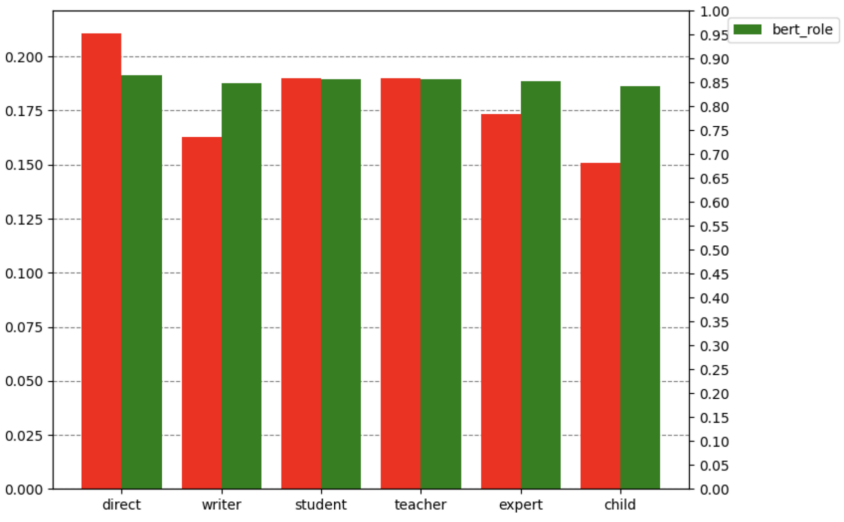


Figure 5.7: Role bar chart.

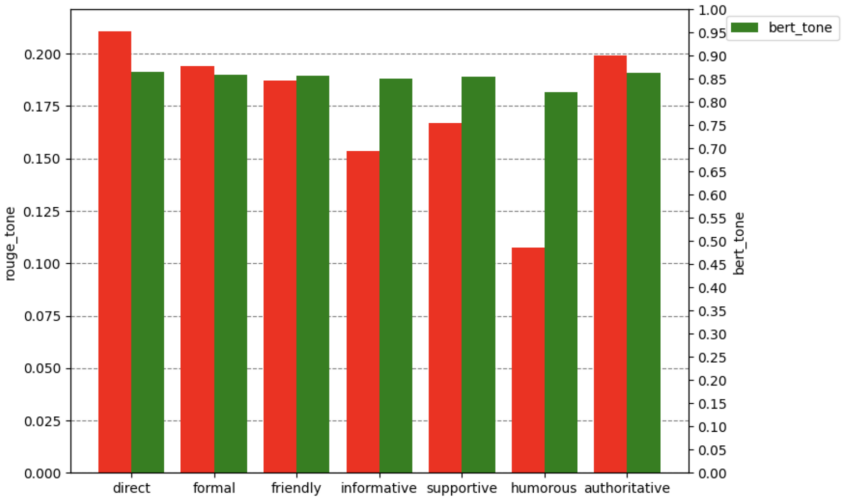


Figure 5.8: Tone bar chart.

For the rouge metric, the higher score is low with 0.24 for the role and 0.23 for tone. For the Bert metric, the highest score is high with 0.87 for both role and tone. As we can see above, on average the direct answer is better than the crafted prompts answers.

Moreover, these tools and visualisations are not very complex and relevant with this amount of rows. They are not explaining that much but they had served their purpose for this part. That is to say, they show that the preparation, generation and analysis of the small dataset are possible. The results are partial because it is just a piece of the dataset.

Nevertheless, at first sight, it is not so surprising that the direct answer is the best on average. Each question is different and for now, I can just say that no role or tone is efficient for this entire wiki_qa dataset.

To conclude, the manipulation with the small dataset helped to understand better the data and the Python mechanics to manipulate it. The data looks workable so now, the real analysis will take place.

5.2.2. Full dataset

In this section, I will try to reproduce the same process I did in the small dataset experiments but with more data. The amount of rows is not defined in advance because it depends on the computational cost of the operations.

Preparation of the dataset

The dataset I will use is the same as before, the wiki_qa one. As explained before I united the training, testing and validating dataset. I will use the number of rows in the order of the dataset.

Computational limit and saving data

Among the 1242 rows, I take a subset of 500 questions and try to generate the direct answer for each one. I use this command:

```
df = dflarge.copy()
df = df[:500]
```

In this section, I will cover the problem I had with the computation limit of the API.

After taking a subset of 500 rows, I try to generate the 500 direct answers with the API.

Unfortunately, after a couple of dozen minutes, I faced this error:

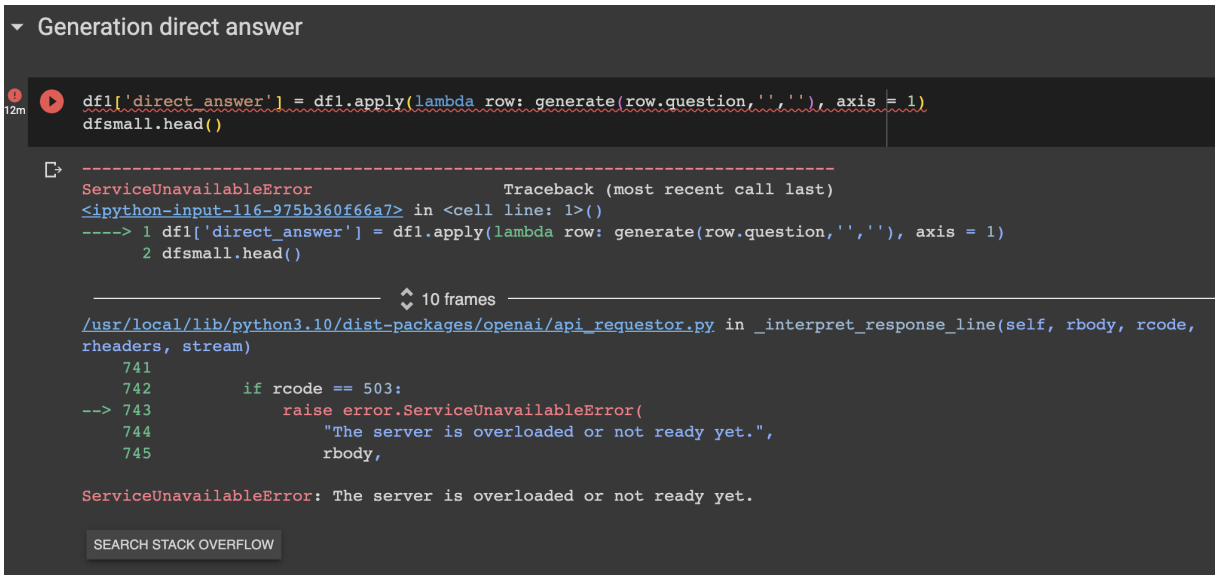


Figure 5.9: Screenshot of overload failure of ChatGPT API.

After investigating on forums, this problem is widespread and can occur while submitting too much data at a time when the servers are busy. So I waited and tried at different times of the day but each time I faced the same message.

I understood that the amount of data might be too big for the API. So I decided to take a smaller subset of the dataset.

I tried with 300 rows, 200 rows and I faced the same message again and again.

Finally, with 100 rows and 10 minutes of waiting, in the morning only, I succeeded to generate 100 direct answers. I figured out that in the morning Americans are sleeping and then OpenAI servers are less busy.

This test concludes that I will generate my answers with a subset of 100 rows. I did this technique twice. Both df1 and df2 are distinct subsets of 100 rows. After constructing both datasets with the next part process, I concatenate these two to have a 200 rows dataset and analyse it.

After each generation, I save my new dataset on a CSV to not lose the content after disconnecting the Google Colab session.

Google Colab offers a package that allows me to save my numpy dataframe from my notebook in a CSV stored in Google Drive. Then I can load it again into Google Colab. One point of attention, when loading a CSV into Python, every column are string object, so you may need to convert it to use it.

Here is the piece of code I used to save and load my dataset.

```
from google.colab import drive
def save_drive(df, namefile):
  drive.mount('/content/drive')
  path = '/content/drive/My Drive/TFM/'+namefile+'.csv'

  with open(path, 'w', encoding = 'utf-8-sig') as f:
    df.to_csv(f)
  return 'success'

save_drive(df,'dataframe_v1')

url = 'https://drive.google.com/uc?id={}'.format('dataframe_v1_&652!')
df = pd.read_csv(url)
```

Thanks to that, every generation is permanently stored in Google Drive.

Generating the answers

I know the amount of data I need to generate my answers and not cause failure. As said before, the process I am going to explain has been realised twice. I create the first dataframe, called df1, with rows number 20 to 120 and df2, with 120 to 220. In that way, I am not computing again the small dataset.

For the rest of the section, I will only explain and show examples with df1, as they are identical.

Direct generation The direct generation corresponds to the generation of the answer with no modification in the prompt, that is to say, submitting to ChatGPT the questions as it stands in the dataset and getting back the answer.

The piece of code that does this is:

```
df1['direct_answer'] = df1.apply(lambda row: generate(row.question
    ,'',''), axis = 1)
```

A new column is created in the DataFrame called direct_answer. The apply() method iterates over each row in the DataFrame and calls the generate() function on each row. It returns the direct answer to the question in a string object. The code takes 10 minutes to run.

Roles generation Then, the crafted prompt responses are generated. The structure for generating is the same adding the role of writer, student, teacher, expert and child.

```
df1['writer_answer'] = df1.apply(lambda row: generate(row.question,'
    Writer',''), axis = 1)
df1['student_answer'] = df1.apply(lambda row: generate(row.question,'
    student',''), axis = 1)
df1['teacher_answer'] = df1.apply(lambda row: generate(row.question,'
    teacher',''), axis = 1)
df1['expert_answer'] = df1.apply(lambda row: generate(row.question,'
    expert',''), axis = 1)
df1['child_answer'] = df1.apply(lambda row: generate(row.question,'child
    ',''), axis = 1)
```

Each of the generations takes at least 10 minutes and the dataframe is saved between each one.

Tones generation The structure for generating is the same as the tone: formal, friendly, informative, supportive, humorous and authoritative.

```
df1['formal_answer'] = df1.apply(lambda row: generate(row.question,'','
    formal'), axis = 1)
df1['friendly_answer'] = df1.apply(lambda row: generate(row.question
    ,'','friendly'), axis = 1)
df1['informative_answer'] = df1.apply(lambda row: generate(row.question
    ,'','informative'), axis = 1)
```

```
df1['supportive_answer'] = df1.apply(lambda row: generate(row.question
    ,'','supportive'), axis = 1)
df1['humorous_answer'] = df1.apply(lambda row: generate(row.question
    ,'','humorous'), axis = 1)
df1['authoritative_answer'] = df1.apply(lambda row: generate(row.
    question,'','authoritative'), axis = 1)
```

Finally, after each generation, df1 has 15 string object columns.

question_id	question	answer	direct_answer	writer_answer	student_answer	teacher_answer	expert_answer	child_answer	formal_answer	friendly_answer	informative_answer	supportive_answer	humorous_answer	authoritative_answer
Q11	how big is bm	Employing ov	BMC Software is	BMC Software is	As a student, I do no	As a teacher, I don't l	As an AI langua	BMC Software in	BMC Software, h	BMC Software is a	BMC Software	BMC Softwar	Well, let me t	BMC Software i
Q48	how long was	The black-anc	I Love Lucy was	I Love Lucy was	I Love Lucy was on t	I Love Lucy was on t	As an AI langua	I'm sorry, I don't	I Love Lucy aire	I Love Lucy was on	"I Love Lucy" is	I Love Lucy w	Oh, buckle up	I Love Lucy was
Q112	what bird fam	Owls are a gr	The owl belongs	The owl belongs	The owl belongs to t	The owl belongs to t	As a language m	Oh, the owl is pa	The bird family t	The owl belongs to	The owl belong	The owl belor	Ah, the majes	The owl belong
Q167	how many pe	The Oklahom	The Oklahoma C	The Oklahoma C	The Oklahoma City t	The Oklahoma City t	As an AI langua	I'm really sorry, t	The Oklahoma C	I'm sorry to hear thi	The Oklahoma	The Oklahom	Well, apologi	168 people wer
Q197	how many xb	There are cur	There are a vast	As an AI langua	As a student, I do no	As a teacher, I don't l	As an AI langua	Oh boy, there ar	There is no exac	Hey there! I'm glad	As of Septemb	There are ap	Oh boy, let m	As of now, there
Q253	how many pla	Jupiter is the	Jupiter is approx	Jupiter is the fifth	Jupiter is the fifth pla	Jupiter is the fifth pla	As an expert, I c	Jupiter is the fifth	Jupiter is locatec	Jupiter is around 4	Jupiter is the fif	Jupiter is app	Well, let me p	Jupiter is locate

Figure 5.10: Screenshot of df1 csv after generation.

Computational time In total, for a dataframe of 100 rows, the generation computational time is reaching 3 hours.

Metrics calculation

After generating all the answers needed, metrics will allow us to measure the efficiency of the response and compare them.

As a reminder, for this thesis, Bertscore and Rouge-Lsum metrics are used.

Calculation process For both role and tone analysis, the same technique is applied. As a reminder, I choose to use the Rouge-Lsum variant and the F1-Score for Bertscore.

I compute the score for every different answer, including the direct one, and store it in a temporary column. Each calculation needs two parameters, the prediction is a ChatGPT answer and the reference is the human-written answer.

Then I create a new column where each row aggregates the score in a list. Therefore, four new columns are created permanently: rouge_role, bert_role, rouge_tone, bert_tone.

For example, for every row of the rouge_role column, there is a float list of 6 elements corresponding to the direct answer and the five other roles. Here is the code associated:

```
df1['rouge1'] = df1.apply(lambda row: rouge.compute(predictions=[row.
    direct_answer],references=[row.answer])['rougeLsum'], axis = 1)
df1['rouge2'] = df1.apply(lambda row: rouge.compute(predictions=[row.
    writer_answer],references=[row.answer])['rougeLsum'], axis = 1)
df1['rouge3'] = df1.apply(lambda row: rouge.compute(predictions=[row.
    student_answer],references=[row.answer])['rougeLsum'], axis = 1)
```

```
df1['rouge4'] = df1.apply(lambda row: rouge.compute(predictions=[row.
    teacher_answer],references=[row.answer])['rougeLsum'], axis = 1)
df1['rouge5'] = df1.apply(lambda row: rouge.compute(predictions=[row.
    expert_answer],references=[row.answer])['rougeLsum'], axis = 1)
df1['rouge6'] = df1.apply(lambda row: rouge.compute(predictions=[row.
    child_answer],references=[row.answer])['rougeLsum'], axis = 1)

# Create a new column 'rouge_role' containing the merged values as lists
df2['rouge_role'] = df2.apply(lambda row: [row['rouge1'], row['rouge2'],
     row['rouge3'], row['rouge4'], row['rouge5'], row['rouge6']], axis=1)

# Drop the individual 'rouge' columns if needed
df2.drop(columns=['rouge1', 'rouge2', 'rouge3', 'rouge4', 'rouge5', '
    rouge6'], inplace=True)
```

At the end, the first row of the column 'rouge_role' is containing: [0.16, 0.11, 0.06, 0.10, 0.07, 0.12] associated with the direct rouge answer, writer answer, student answer, teacher answer, expert answer and child answer.

The same thing is done with a list of 7 elements for the tones.

Computational time For both role and tone, Bertscore uses the roberta-large model from huggingface. Every column calculation is taking 10 minutes.

Rouge score computing time is lower, around 4 minutes for each column.

In total, for a dataframe of 100 rows, the metric computational time is reaching 3 hours.

In the next section, the metrics will be analysed to compare the different answers.

5.3. Interpretation of experimental findings

The interpretation of experimental findings is a critical step in the thesis process. I will try to draw conclusions and make sense of the results of the underlying phenomenon.

As the roles and tones generation are done separately, their interpretation can be done separately. It is not the goal of this thesis to cross the roles and tones to determine the best combination. I assume that the best combination is the best role combined with the best tone.

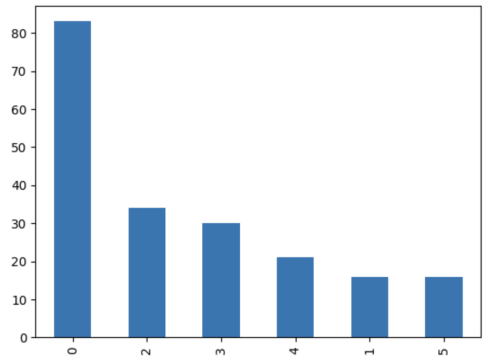


Figure 5.11: Best Rouge role index occurrence bar chart.

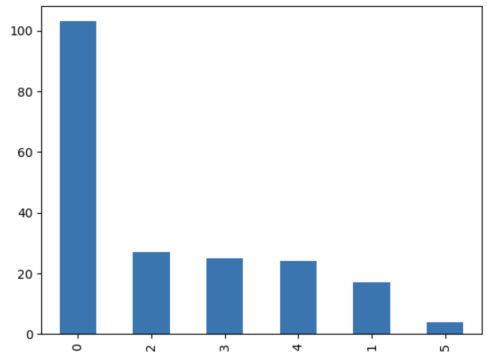


Figure 5.12: Best Bert role index occurrence bar chart.

5.3.1. Roles interpretation

In this section, I will present the findings about the crafted role prompts. Different analyses are conducted and their conclusions are leading to the next analysis.

Best answer choice

Like in the small dataset analysis, I begin with the macro view. That is to say, according to each metric, finding which role is the best. I create two new columns, 'rouge_index_role' and 'bert_index_role', containing the index with the highest bert and rouge scores. (integer associated with the role - see Test on a small dataset, Interpretation). Here is the code associated:

```
df1['rouge_index_role'] = df1.apply(lambda row: row.rouge_role.index(max
    (row.rouge_role)), axis = 1)
df1['bert_index_role']  = df1.apply(lambda row: row.bert_role.index(max(
    row.bert_role)), axis = 1)
```

Then, for Rouge and BertScore, I count the occurrence of each index to see which one appears the most. Here is the result:

For Rouge, 41.5% of the time, the direct answer is the best one. Followed by student and teacher with 17% and 15%. Expert, writer and child are at 10.5% and 8% each.

For BertScore, 51.5% of the time, the direct answer is the best one. Followed by student and teacher with 13.5% and 12.5%. Expert, writer and child are at 12%, 8.5% and 2%.

From these observations, I can deduce that when choosing only the highest score and erasing the rest, globally, the direct answer is the best.

Nevertheless, the student and teacher are well positioned and on some specific questions,

they are better than the direct one. Moreover, for now, I don't know how far the crafted prompt scores are when the direct score is the best. Choosing the maximum is erasing many data and information.

Average analysis

In this part, I will investigate the data in its entirety. I will take into account the score even if they are not the best.

The first idea I had was to print the heatmap with detailed scores like in the small dataset. But now, with 200 rows, visually it is not convenient to read and does not provide any global information.

So, I decided to compute the mean of each score. Here is the result, Rouge and Bert are on two different y scales:

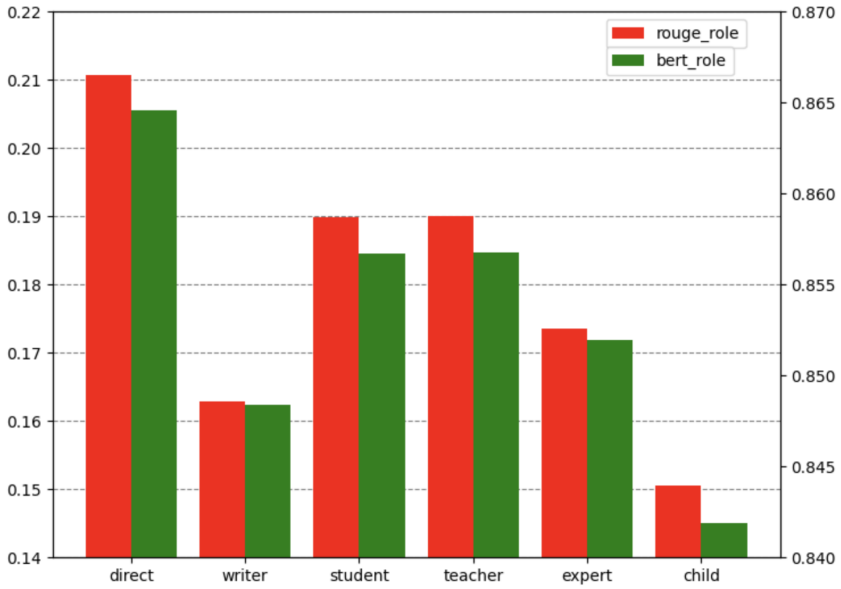


Figure 5.13: Rouge and Bert role score on average.

On average, for both Rouge and Bert scores, the direct score is higher. This bar plot emphasises the fact that teacher and student roles are the two more interesting for this dataset. Expert is in between and the writer and child are clearly below the others.

This visualisation also shows that the rouge score is way lower than the bertscore. The score range is from 0.15 to 0.21 for rouge and from 0.84 to 0.87 for bertscore. The two metrics are very different and not based on the same techniques: rouge is based on n-grams and bertscore on the Bert model. It makes no sense to compare them directly. Only

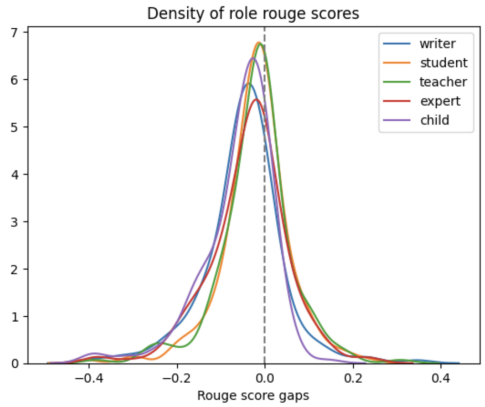


Figure 5.14: Density of rouge role scores.

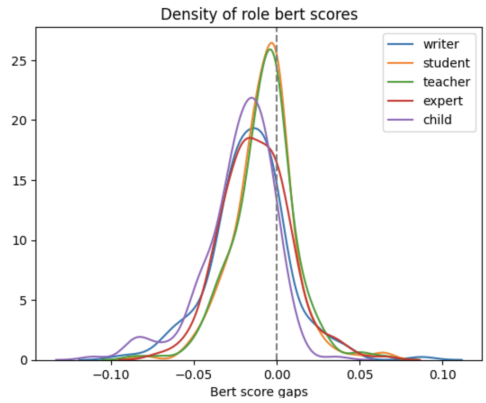


Figure 5.15: Density of Bert role scores.

their different evolution and ratio are relevant for this study to draw proper conclusions.

Finally, this analysis confirmed the previous one, teacher and student roles provide good results on average but not as good as the direct one.

For the next analysis, I will investigate the gap between metrics. That is to say, when the direct answer is not the best one, does another role provide a significant advantage? Does the direct answer provide a significant advantage every time it is the higher score or is it similar to adding a role?

Gap analysis

Firstly, I will analyse the gap between the direct score and each other column. Here is a density chart of the score difference to direct score:

As shown above, on average roles are less efficient than the direct answer: the maxima of the density curves are below 0. Nevertheless, it exists some questions where each role is better than the direct answer. In the BertScore graphic, it is clear that student and teacher roles have a bigger area under the curb than the others.

On another side, the graphics are showing the positive and negative contributions to add a role. We see on the two graphics that the advantage of adding a role is not worth it. For example, in the BertScore graphic that there are more cases of losing -0.05 than gaining 0.05. Moreover, there are more chances of losing and very few chances of gaining a large score.

This analysis tells us that, it is not worth it to add a role because we can lose more points than gain them and the ones we can gain are not big.

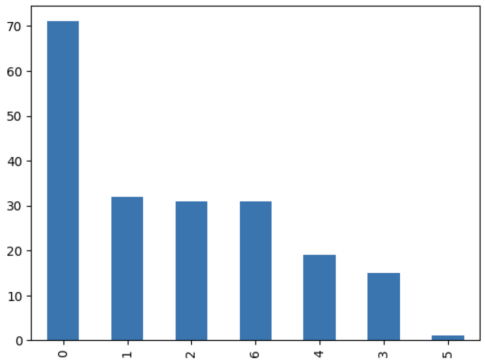


Figure 5.16: Best Rouge tone index occurrence bar chart.

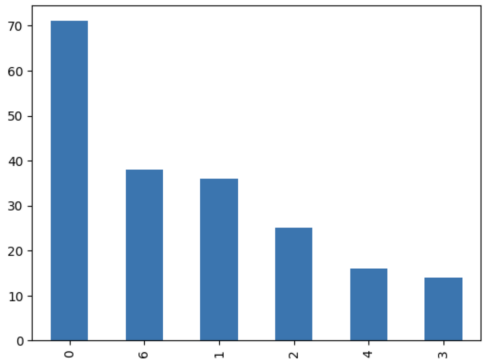


Figure 5.17: Best Bert tone index occurrence bar chart.

To conclude, on this wiki_qa dataset, there are no specific roles that outperform the direct answer. It depends on the question.

5.3.2. Tones interpretation

In this section, I will present the findings about the crafted tones prompts.

Best answer choice

Like in the small dataset and role analysis, I begin with the macro view. I create two new columns, 'rouge_index_tone' and 'bert_index_tone', containing the index with the highest bert and rouge scores. (integer associated with the role - see Test on a small dataset, Interpretation). Then, for Rouge and BertScore, I count the occurrence of each index to see which one appears the most. Here is the result:

For Rouge, 35.5% of the time, direct answer is the best one. Followed by formal, friendly and authoritative at 16%, and twice at 15.5%. Expert, writer and child are at 10.5% and 8% each.

For BertScore, 51.5% of the time, the direct answer is the best one. Followed by student and teacher with 13.5% and 12.5%. Informative, supportive and humorous are at 9.5%, 7.5% and 0.5%.

From these observations, I can deduce that when choosing only the highest score and erasing the rest, globally, the direct answer is the best.

Nevertheless, formal, friendly and authoritative tones are well positioned and on some specific questions, they are better than the direct one. Moreover, for now, I don't know how far the crafted prompt scores are when the direct score is the best. Choosing the

maximum is erasing many data and information.

Average analysis

In this part, I will investigate the data in its entirety. I will take into account the score even if they are not the best.

Same to a role analysis, I decided to compute the mean of each score. Here is the result, Rouge and Bert are on two different y scales:

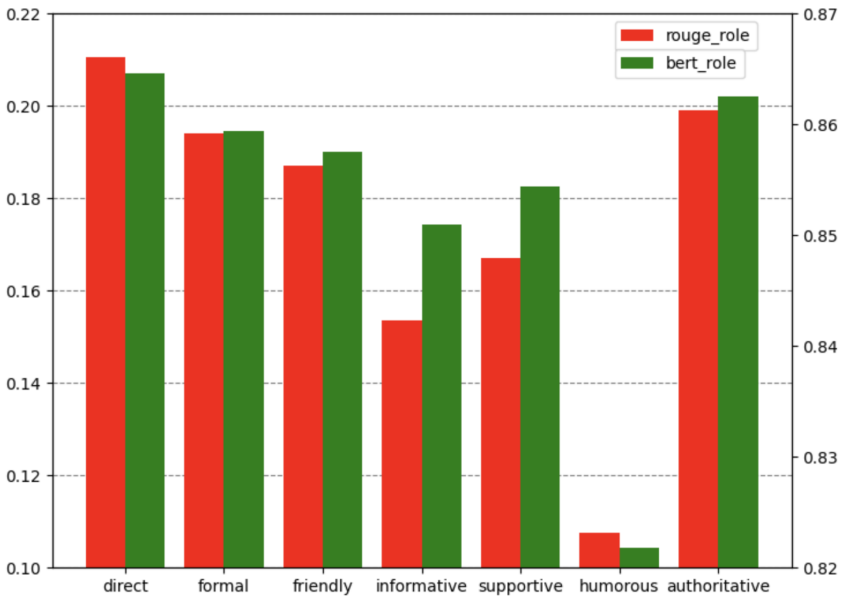


Figure 5.18: Rouge and Bert role score on average.

On average, for both Rouge and Bert scores, the direct score is again higher. Nevertheless, authoritative seems very interesting. For both Rouge and Bert scores, its value is closer than what we could observe with the roles. Formal and friendly tones are also interesting for this dataset. Informative and supportive are in between and humorous is clearly below the others.

For the next analysis, I will investigate the gap between metrics. That is to say if the advantage of adding a tone, especially the authoritative one provides a significant advantage.

Gap analysis

Same as for the roles, I will analyse the gap between the direct score and each other column. Here is a density chart of the score difference to direct score:

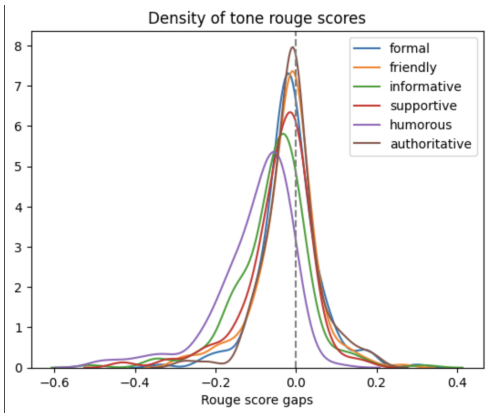


Figure 5.19: Density of rouge role scores.

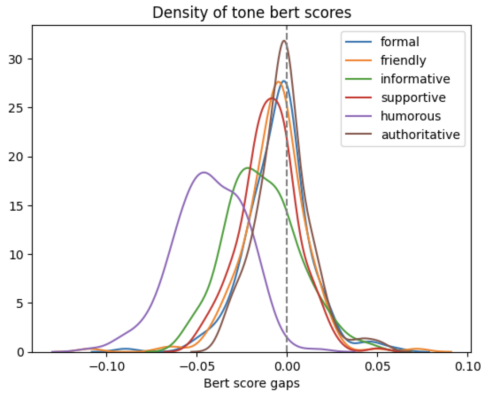


Figure 5.20: Density of Bert role scores.

As shown above, on average tones are less efficient than the direct answer: the maxima of the density curves are below 0. Nevertheless, Nevertheless, it exists some questions where each tone is better than the direct answer.

The same comments from role gap analysis are accurate here also. Besides that, the authoritative tone is very interesting. On average it is almost as good as the direct answer. For BertScore, the maximum penalty it could cause is -0.05, but the maximum significant gain it could obtain is more than +0.06. The Rouge is not so clear because there is still a residue below -0.2. As it is the same for all tones, it can be due to the intrinsic technique of the metric. As Rouge is based on n-grams it is less prone to detect the difference of meanings when the same words are used.

To conclude, on this wiki_qa dataset, it is interesting to add the authoritative role to generate better answers. It can be explained because the authoritative tone tends to generate more concise answers. The wiki_qa dataset includes diverse themes but all answers are quite short when compared to the generation ability of the today models.

5.4. Experimental conclusion

The experimental part is an important section of this thesis. It represents the achievement of all the previous research and analysis.

5.4.1. Role prompts

The analysis of this dataset leads to a point where the role benefit is not established. The different results prove that the direct answer, that is to say, the answer to the question

with no modification is the best one.

At first sight, it is a surprise. The dataset wiki_qa is an online encyclopedia. Then, they are mostly consulted by students. So the written style could have been more teacher-oriented.

Here are the conclusions I draw from the role analysis:

- ChatGPT 3.5-Turbo perform well if the question is well-written with no role-crafted prompt.
- Still, it is beneficial to add a role, such as teacher or student on some questions because they can provide better answers. However, this thesis is globally conducted over the dataset and doesn't highlight the type of question where each role is efficient.

5.4.2. Tone prompts

The analysis of this dataset leads to a point where the tone benefit is clear. The different results prove that the direct answer is efficient but not the best. Using a crafted prompt beginning with "Answer with an authoritative tone" is more efficient.

Here are the conclusions I draw from the role analysis:

- The wiki_qa dataset tends to favour concise answers. The authoritative tone doesn't bother itself with unnecessary words and ChatGPT 3.5-Turbo succeed to choose well the words used.
- Still, ChatGPT 3.5-Turbo perform well if the question is well-written with no crafted prompt.

6 | Conclusions and future developments

Prompt engineering is a complex subject. In this thesis, I designed and optimised input prompts for generative artificial intelligence models in a very specific way. I used ChatGPT Turbo-3.5 API and the wiki_qa dataset.

This dataset, a Question-Answering dataset, is reflecting a specific usage of Generative AI. Today, many people are using this technology like they are their search engine. The wiki_qa dataset corresponds exactly to what ordinary people could search for. Habits are changing and people want a more personalized search experience while having a clear answer to their questions. To help them to write the better prompt possible, I conducted this thesis.

After searching for papers, videos, and Twitter accounts explaining the Generative AI technology, I search for methods to measure the accuracy of a generation. After exploring dozens of metrics I chose 2 of them to help me understand the data.

Prompt engineering is a new trend with few resources available online. I search for different techniques and chose two of them: crafting prompts by specifying roles and tones. The identification of them will allow us to get better answers to this type of question.

Then, I chose the dataset that would correspond to the needs I want to answer: the wiki_qa. After determining the right working method with my supervisors, I began to realise tests: connecting to the ChatGPT API, processing the dataset, generating answers, computing metrics and analysing the results.

The results are mixed. For the role part, none of the chosen ones are performing well on the whole dataset. Teacher and student roles are interesting in some questions. For the tone part, the authoritative tone is performing well on the entire dataset by generating more concise answers. The score are proving that the risk is worth it.

Finally, for a general question concerning Wikipedia information, the best answer you get is by adding a crafted piece prompt before your question: 'Answer with an authoritative

tone: ...'

This thesis answered some questions but raised other ones. First, the data studied here was not categorised but it could be interesting to identify what role and tone are performing better on a specific group of questions. For example, students and teachers for science questions, experts for medicine etc.

In parallel with this work, it could be interesting to try other roles and other tones to be sure the best one is analysed. Moreover, it would be interesting to do the same exercise on another dataset with different source questions like coding generation to observe the ability of a model to create code.

To conclude, the generative AI models are spreading everywhere and it is becoming evident that English may become the new programming language.

However, it is important to study existing models like ChatGPT-3.5-Turbo in this thesis. It contributes to understand deeper models and stepping back from the dangerous race to ever-larger unpredictable black-box models with emergent capabilities.

Bibliography

R. Al-Rfou, M. Pickett, J. Snaider, Y.-H. Sung, B. Strope, and R. Kurzweil. Conversational contextual cues: The case of personalization and history for response ranking. *arXiv preprint arXiv:1606.00372*, 2016.

S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

E. M. Bender and B. Friedman. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 2018.

T. Boronat, N. Montañés Muñoz, D. Garcia-Sanoguera, O. Fenollar, and V. Fombuena. Utilización de técnicas kanban para la gestión de tesis doctorales. In *In-Red 2017. III Congreso Nacional de innovación educativa y de docencia en red.*, pages 110–116. Editorial Universitat Politècnica de València, 2017.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Y. Chen and S. Eger. Menli: Robust evaluation metrics from natural language inference. *arXiv preprint arXiv:2208.07316*, 2022.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

J. Fu, S.-K. Ng, Z. Jiang, and P. Liu. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*, 2023.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai. Ai song contest: Human-ai co-creation in songwriting. *arXiv preprint arXiv:2010.05388*, 2020.

J. Kasai, K. Sakaguchi, R. L. Bras, L. Dunagan, J. Morrison, A. R. Fabbri, Y. Choi, and N. A. Smith. Bidimensional leaderboards: Generate and evaluate language hand in hand. *arXiv preprint arXiv:2112.04139*, 2021.

J. N. Kather, N. Ghaffari Laleh, S. Foersch, and D. Truhn. Medical domain knowledge in domain-agnostic generative ai. *NPJ digital medicine*, 5(1):90, 2022.

C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

G. Marcus, E. Davis, and S. Aaronson. A very preliminary analysis of dall-e 2. *arXiv preprint arXiv:2204.13807*, 2022.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

T. Pearce, M. Zaki, A. Brintrup, N. Anastassacos, and A. Neely. Uncertainty in neural networks: Bayesian ensembling. *stat*, 1050:12, 2018.

M. Popović. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395, 2015.

M. Przybocki, K. Peterson, S. Bronsart, and G. Sanders. The nist 2008 metrics for machine translation challenge—overview, methodology, metrics, and results. *Machine Translation*, 23:71–103, 2009.

Y. Qin, W. Yuan, G. Neubig, and P. Liu. T5score: Discriminative fine-tuning of generative evaluation metrics. *arXiv preprint arXiv:2212.05726*, 2022.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. *Improving language understanding by generative pre-training*, 2018.

M. Sap, D. Card, S. Gabriel, Y. Choi, and N. A. Smith. The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1668–1678, 2019.

P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

T. Sellam, D. Das, and A. P. Parikh. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*, 2020.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, 2006.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

S. Xie, R. Rastogi, and M. Chang. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation. *Natural Lang. Process. Deep Learn*, 2017.

Y. Yang, W.-t. Yih, and C. Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1237. URL `https://aclanthology.org/D15-1237`.

W. Yuan, G. Neubig, and P. Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277, 2021.

B. H. Zhang, B. Lemoine, and M. Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340, 2018.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*, 2019.

List of Figures

7 | Appendix A: Notebook

TFM GOOGLE COLAB

Required installations

```
!pip install datasets
!pip install evaluate
!pip install rouge_score
!pip install openai
!pip install bert_score
```

```
Collecting datasets
  Downloading datasets-2.13.1-py3-none-any.whl (486 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 486.2/486.2 kB 4.1 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.22.4
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (9.0
Collecting dill<0.3.7,>=0.3.0 (from datasets)
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 7.4 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.65.
Collecting xxhash (from datasets)
  Downloading xxhash-3.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (212 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.5/212.5 kB 5.1 MB/s eta 0:00:00
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.15-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 7.2 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]>=2021.11.1 in /usr/local/lib/python3.10/dist-packages (from datas
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.8.4)
Collecting huggingface-hub<1.0.0,>=0.11.0 (from datasets)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 7.7 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datase
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from a
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datas
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->da
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->dat
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from hu
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0
INFO: pip is looking at multiple versions of multiprocess to determine which version is compatible with other
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.14-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.3/134.3 kB 5.4 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8
Installing collected packages: xxhash, dill, multiprocess, huggingface-hub, datasets
Successfully installed datasets-2.13.1 dill-0.3.6 huggingface-hub-0.16.4 multiprocess-0.70.14 xxhash-3.2.0
Collecting evaluate
  Downloading evaluate-0.4.0-py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.4/81.4 kB 1.3 MB/s eta 0:00:00
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (2.
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from evaluate) (1.22.4
Requirement already satisfied: dill in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.3.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from evaluate) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (2
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from evaluate) (4.65.
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from evaluate) (3.2.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.70.
Requirement already satisfied: fsspec[http]>=2021.05.0 in /usr/local/lib/python3.10/dist-packages (from evalu
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from evalua
```

Required importation

```
from google.colab import drive
from datasets import load_dataset
import pandas as pd
import evaluate
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

rouge = evaluate.load('rouge')
bert = evaluate.load('bertscore')

import openai
# https://www.youtube.com/watch?v=pGOyw_M1mNE&t=175s
openai.api_key = "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"
```

```
Downloading builder script: 100%  6.27k/6.27k [00:00<00:00, 128kB/s]
Downloading builder script: 100%  7.95k/7.95k [00:00<00:00, 124kB/s]
```

Useful function

```
from google.colab import drive
def save_drive(df, namefile):
  drive.mount('/content/drive')
  path = '/content/drive/My Drive/TFM/'+namefile+'.csv'

  with open(path, 'w', encoding = 'utf-8-sig') as f:
    df.to_csv(f)
  return 'success'


def generate(question,role,tone):
  if role=='' and tone=='':
    prompt=question
  if role=='' and len(tone)>0:
    prompt = 'Answer with a '+tone+' tone: ' + question
  if tone=='' and len(role)>0:
    prompt = 'Answer as a '+role+': ' + question
  else:
    prompt = 'Answer as a '+role+' and with a '+tone+' tone: ' + question

  completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=[{"role": "user", "content": prompt}])
  return completion.choices[0].message.content

roles = ['direct','writer','student','teacher','expert','child']
tones = ['direct','formal','friendly','informative','supportive','humorous','authoritative']



def convert_string_to_list(string):
  """Converts a string of numbers in a list format to a list.

  Args:
    string: A string of numbers in a list format.

  Returns:
    A list of numbers.
  """

  list_of_numbers = []
  for number in string[1:-1].split(','):
    list_of_numbers.append(float(number))
  return list_of_numbers
```

Load wiki_qa dataset

```
from datasets import load_dataset
```

```
# https://huggingface.co/datasets/wiki_qa

# get entire dataset
dataset = load_dataset("wiki_qa", "raw")

# sample from the test split
print("Sample from test dataset split")
test_sample = dataset["test"][0]
print("Fields in the sample: ", [key for key in test_sample.keys()])
print("question: ", test_sample["question"])
print("document_title: ", test_sample["document_title"])
print("answer: ", test_sample["answer"])
print("label: ", test_sample["label"])
print("\n-----------\n")
```

```
Downloading builder script: 100%  3.79k/3.79k [00:00<00:00, 206kB/s]
Downloading metadata: 100%  1.77k/1.77k [00:00<00:00, 45.3kB/s]
Downloading readme: 100%  13.6k/13.6k [00:00<00:00, 513kB/s]
Downloading and preparing dataset wiki_qa/raw to /root/.cache/huggingface/datasets/wiki_qa/raw/0.1.0/d2d236b5
Downloading data: 100%  7.09M/7.09M [00:00<00:00, 8.39MB/s]
Dataset wiki_qa downloaded and prepared to /root/.cache/huggingface/datasets/wiki_qa/raw/0.1.0/d2d236b5cbdc6f
100%  3/3 [00:00<00:00, 65.35it/s]
Sample from test dataset split
Fields in the sample:  ['question_id', 'question', 'document_title', 'answer', 'label']
question:  HOW AFRICAN AMERICANS WERE IMMIGRATED TO THE US
document_title:  African immigration to the United States
answer:  African immigration to the United States refers to immigrants to the United States who are or were r
label:  0

-----------
```

Pre-processing

```
import pandas as pd
dfv = pd.DataFrame(dataset['validation'])
print(len(dfv))
print(len(dfv[dfv['label']==1]))
```

```
2733
140
```

```
import pandas as pd
dft = pd.DataFrame(dataset['test'])
print(len(dft))
print(len(dft[dft['label']==1]))
```

```
6165
293
```

```
import pandas as pd
dftr = pd.DataFrame(dataset['train'])
print(len(dftr))
print(len(dftr[dftr['label']==1]))
```

```
20360
1040
```

```
import pandas as pd
df = pd.concat([dfv[dfv['label']==1], dft[dft['label']==1],dftr[dftr['label']==1]])
df = df.drop(['label','document_title'],axis=1)
```

```
df = df.drop_duplicates(subset='question_id')
df
```

	question_id	question	answer
12	Q11	how big is bmc software in houston, tx	Employing over 6,000, BMC is often credited wi...
15	Q48	how long was i love lucy on the air	The black-and-white series originally ran from...
23	Q66	how did armando christian perez become famous	Armando Pérez (born January 15, 1981), better ...
91	Q112	what bird family is the owl	Owls are a group of birds that belong to the o...
154	Q167	how many people were killed in the oklahoma ci...	The Oklahoma blast claimed 168 lives, includin...
...	...	...	...
20281	Q3034	what is melissa and joey about	The series follows local politician Mel Burke ...
20303	Q3037	What is an economic feature?	Other broad distinctions include those between...
20320	Q3039	what is the average american income	U.S. median household income fell from $51,144...
20338	Q3042	When was Apple Computer founded	The company was founded on April 1, 1976, and ...
20348	Q3043	what is section eight housing	Section 8 of the Housing Act of 1937 (), often...

1242 rows × 3 columns

Test Small dataset

API connection

```
import openai

# https://www.youtube.com/watch?v=pGOyw_M1mNE&t=175s

openai.api_key = "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"

completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=[{"role": "user", "content": "Give me 3
print(completion.choices[0].message.content)
```

```
1. Language Learning Assistant: Build an app that utilizes OpenAI's language models to help users learn new l

2. Resume/CV Writer: Develop an app that assists users in creating professional resumes and CVs. The app can

3. Virtual Personal Stylist: Create an app that uses OpenAI's language models to provide personalized fashior
```

Load 20rows

```
dfsmall = df.copy()
dfsmall = dfsmall[:20]
dfsmall
```

	question_id	question	answer
12	Q11	how big is bmc software in houston, tx	Employing over 6,000, BMC is often credited wi...
15	Q48	how long was i love lucy on the air	The black-and-white series originally ran from...
23	Q66	how did armando christian perez become famous	Armando Pérez (born January 15, 1981), better ...
91	Q112	what bird family is the owl	Owls are a group of birds that belong to the o...
154	Q167	how many people were killed in the oklahoma ci...	The Oklahoma blast claimed 168 lives, includin...
220	Q197	how many xbox 360 games are there	There are currently 952 games (multiplatform: ...
249	Q253	how many planets is jupiter away from the sun?	Jupiter is the fifth planet from the Sun and t...
304	Q305	what happened on the moon during the period of...	During this event a very large number of the i...
326	Q335	what does the federal reserve do	Its duties have expanded over the years, and t...
395	Q385	how manv nature oceans are on earth	In the context of Earth . it refers to one or ...

```
def generate(question,role,tone):
  if role=='' and tone=='':
    prompt=question
  if role=='' and len(tone)>0:
    prompt = 'Answer with a '+tone+' tone: ' + question
  if tone=='' and len(role)>0:
    prompt = 'Answer as a '+role+': ' + question
  else:
    prompt = 'Answer as a '+role+' and with a '+tone+' tone: ' + question

  completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=[{"role": "user", "content": prompt}])
  return completion.choices[0].message.content

roles = ['direct','writer','student','teacher','expert','child']
tones = ['direct','formal','friendly','informative','supportive','humorous','authoritative']
```

Generation of direct prompt

```
dfsmall['direct_answer'] = dfsmall.apply(lambda row: generate(row.question,'',''), axis = 1)
# for a 20 rows dataset, it took 1m50s to generate 20 answers with ChatGPT API
#G1 stands for "generation 1" that is to say the prompt is the question with no addings.
dfsmall.head()
```

	question_id	question	answer	
12	Q11	how big is bmc software in houston, tx	Employing over 6,000, BMC is often credited wi...	BMC Software is
15	Q48	how long was i love lucy on the air	The black-and-white series originally ran from...	I Love Lucy
23	Q66	how did armando christian perez become famous	Armando Pérez (born January 15, 1981), better ...	Armando Christi
91	Q112	what bird family is the owl	Owls are a group of birds that belong to the o...	The owl belon
154	Q167	how many people were killed in the oklahoma ci...	The Oklahoma blast claimed 168 lives, includin...	The Oklahoma City

Generation of roles prompts

```
dfsmall['writer_answer'] = dfsmall.apply(lambda row: generate(row.question,'Writer',''), axis = 1)
```

```
dfsmall['student_answer'] = dfsmall.apply(lambda row: generate(row.question,'student',''), axis = 1)
```

```
dfsmall['teacher_answer'] = dfsmall.apply(lambda row: generate(row.question,'teacher',''), axis = 1)
```

```
dfsmall['expert_answer'] = dfsmall.apply(lambda row: generate(row.question,'expert',''), axis = 1)
```

```
dfsmall['child_answer'] = dfsmall.apply(lambda row: generate(row.question,'child',''), axis = 1)
```

```
dfsmall.head()
```

	question_id	question	answer	direct_answer	writer_answer	student_answer	teacher_
0	Q11	how big is bmc software in houston, tx	Employing over 6,000, BMC is often credited wi...	BMC Software is a substantial presence in Hous...	BMC Software is a prominent technology company...	As a student, I do not have access to current ...	As a teache have access
1	Q48	how long was i love lucy on the air	The black-and-white series originally ran from...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for six seasons, fr...	I Love Lucy wa air for a total
2	Q66	how did armando christian perez become famous	Armando Pérez (born January 15, 1981), better ...	Armando Christian Perez, better known by his s...	Armando Christian Perez, more famously known a...	Armando Christian Perez, also known as Pitbull...	Armando C Perez, better kr
3	Q112	what bird family is the owl	Owls are a group of birds that belong to the o...	The owl belongs to the Strigidae family, chara...	The owl belongs to the family Strigidae, which...	The owl belongs to the bird family known as St...	The owl belong bird family calle
4	Q167	how many people were killed in the oklahoma ci...	The Oklahoma blast claimed 168 lives, includin...	The Oklahoma City bombing, which occurred on A...	The Oklahoma City bombing, which occurred on A...	The Oklahoma City bombing occurred on April 19...	The Oklaho bombing occurred

Generation of tones prompts

```
dfsmall['formal_answer'] = dfsmall.apply(lambda row: generate(row.question,'','formal'), axis = 1)
```

```
dfsmall['friendly_answer'] = dfsmall.apply(lambda row: generate(row.question,'','friendly'), axis = 1)
```

```
dfsmall['informative_answer'] = dfsmall.apply(lambda row: generate(row.question,'','informative'), axis = 1)
```

```
dfsmall['supportive_answer'] = dfsmall.apply(lambda row: generate(row.question,'','supportive'), axis = 1)
```

```
dfsmall['humorous_answer'] = dfsmall.apply(lambda row: generate(row.question,'','humorous'), axis = 1)
```

```
dfsmall['authoritative_answer'] = dfsmall.apply(lambda row: generate(row.question,'','authoritative'), axis = 1)
```

```
dfsmall.head()
```

	question_id	question	answer	direct_answer	writer_answer	student_answer	teacher_answer	expert_answer
0	Q11	how big is bmc software in houston, tx	Employing over 6,000, BMC is often credited wi...	BMC Software is a substantial presence in Hous...	BMC Software is a prominent technology company...	As a student, I do not have access to current ...	As a teacher, I don't have access to real-time...	As an A language model, I don't have access t..
1	Q48	how long was i love lucy on the air	The black-and-white series originally ran from...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for six seasons, fr...	I Love Lucy was on the air for a total of six ...	As an A language model, I can provide you wit..
2	Q66	how did armando christian perez become famous	Armando Pérez (born January 15, 1981), better ...	Armando Christian Perez, better known by his s...	Armando Christian Perez, more famously known a...	Armando Christian Perez, also known as Pitbull...	Armando Christian Perez, better known as Pitbu...	Armando Christian Perez, known by his stage na..
			Owls are the o...					

Metrics calculation

```
dfsmall['rouge1']= dfsmall.apply(lambda row: rouge.compute(predictions=[row.direct_answer],references=[row.answer]
dfsmall['rouge2']= dfsmall.apply(lambda row: rouge.compute(predictions=[row.writer_answer],references=[row.answer]
dfsmall['rouge3']= dfsmall.apply(lambda row: rouge.compute(predictions=[row.student_answer],references=[row.answer
dfsmall['rouge4']= dfsmall.apply(lambda row: rouge.compute(predictions=[row.teacher_answer],references=[row.answer
dfsmall['rouge5']= dfsmall.apply(lambda row: rouge.compute(predictions=[row.expert_answer],references=[row.answer]
dfsmall['rouge6']= dfsmall.apply(lambda row: rouge.compute(predictions=[row.child_answer],references=[row.answer])

# Create a new column 'merged_rouge' containing the merged values as lists
dfsmall['rouge_role'] = dfsmall.apply(lambda row: [row['rouge1'], row['rouge2'], row['rouge3'], row['rouge4'], row

# Drop the individual 'rouge' columns if needed
dfsmall.drop(columns=['rouge1', 'rouge2', 'rouge3', 'rouge4', 'rouge5', 'rouge6'], inplace=True)
```

```
dfsmall['rouge0'] = dfsmall.apply(lambda row: rouge.compute(predictions=[row.direct_answer],references=[row.answer
dfsmall['rouge1'] = dfsmall.apply(lambda row: rouge.compute(predictions=[row.formal_answer],references=[row.answer
dfsmall['rouge2'] = dfsmall.apply(lambda row: rouge.compute(predictions=[row.friendly_answer],references=[row.answ
dfsmall['rouge3'] = dfsmall.apply(lambda row: rouge.compute(predictions=[row.informative_answer],references=[row.a
dfsmall['rouge4'] = dfsmall.apply(lambda row: rouge.compute(predictions=[row.supportive_answer],references=[row.an
dfsmall['rouge5'] = dfsmall.apply(lambda row: rouge.compute(predictions=[row.humorous_answer],references=[row.answ
dfsmall['rouge6'] = dfsmall.apply(lambda row: rouge.compute(predictions=[row.authoritative_answer],references=[row

# Create a new column 'merged_rouge' containing the merged values as lists
dfsmall['rouge_tone'] = dfsmall.apply(lambda row: [row['rouge0'],row['rouge1'], row['rouge2'], row['rouge3'], row[

# Drop the individual 'rouge' columns if needed
dfsmall.drop(columns=['rouge0','rouge1', 'rouge2', 'rouge3', 'rouge4', 'rouge5', 'rouge6'], inplace=True)
```

```
dfsmall['bert1']= dfsmall.apply(lambda row: bert.compute(predictions=[row.direct_answer],references=[row.answer],l
dfsmall['bert2']= dfsmall.apply(lambda row: bert.compute(predictions=[row.writer_answer],references=[row.answer],l
dfsmall['bert3']= dfsmall.apply(lambda row: bert.compute(predictions=[row.student_answer],references=[row.answer],
dfsmall['bert4']= dfsmall.apply(lambda row: bert.compute(predictions=[row.teacher_answer],references=[row.answer],
dfsmall['bert5']= dfsmall.apply(lambda row: bert.compute(predictions=[row.expert_answer],references=[row.answer],l
dfsmall['bert6']= dfsmall.apply(lambda row: bert.compute(predictions=[row.child_answer],references=[row.answer],la

# Create a new column 'merged_rouge' containing the merged values as lists
dfsmall['bert_role'] = dfsmall.apply(lambda row: [row['bert1'], row['bert2'], row['bert3'], row['bert4'], row['ber

# Drop the individual 'rouge' columns if needed
dfsmall.drop(columns=['bert1', 'bert2', 'bert3', 'bert4', 'bert5', 'bert6'], inplace=True)
```

```
dfsmall['bert0']= dfsmall.apply(lambda row: bert.compute(predictions=[row.direct_answer],references=[row.answer],l
dfsmall['bert1']= dfsmall.apply(lambda row: bert.compute(predictions=[row.formal_answer],references=[row.answer],l
dfsmall['bert2']= dfsmall.apply(lambda row: bert.compute(predictions=[row.friendly_answer],references=[row.answer]
dfsmall['bert3']= dfsmall.apply(lambda row: bert.compute(predictions=[row.informative_answer],references=[row.answ
dfsmall['bert4']= dfsmall.apply(lambda row: bert.compute(predictions=[row.supportive_answer],references=[row.answe
dfsmall['bert5']= dfsmall.apply(lambda row: bert.compute(predictions=[row.humorous_answer],references=[row.answer]
dfsmall['bert6']= dfsmall.apply(lambda row: bert.compute(predictions=[row.authoritative_answer],references=[row.an
```

```
# Create a new column 'merged_rouge' containing the merged values as lists
dfsmall['bert_tone'] = dfsmall.apply(lambda row: [row['bert0'],row['bert1'], row['bert2'], row['bert3'], row['bert
```

```
# Drop the individual 'rouge' columns if needed
dfsmall.drop(columns=['bert0','bert1', 'bert2', 'bert3', 'bert4', 'bert5', 'bert6'], inplace=True)
```

```
Some weights of RobertaModel were not initialized from the model checkpoint at roberta-large and are newly in
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference
```

```
dfsmall.head()
```

	question_id	question	answer	direct_answer	writer_answer	student_answer	teacher_answer	expert_answer
0	Q11	how big is bmc software in houston, tx	Employing over 6,000, BMC is often credited wi...	BMC Software is a substantial presence in Hous...	BMC Software is a prominent technology company...	As a student, I do not have access to current ...	As a teacher, I don't have access to real-time...	As an A language model, I don't have access t..
1	Q48	how long was i love lucy on the air	The black-and-white series originally ran from...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for six seasons, fr...	I Love Lucy was on the air for a total of six ...	As an A language model, I can provide you wit..
2	Q66	how did armando christian perez become famous	Armando Pérez (born January 15, 1981), better ...	Armando Christian Perez, better known by his s...	Armando Christian Perez, more famously known a...	Armando Christian Perez, also known as Pitbull...	Armando Christian Perez, better known as Pitbu...	Armando Christian Perez, known by his stage na..
3	Q112	what bird family is the owl	Owls are a group of birds that belong to the o...	The owl belongs to the Strigidae family, chara...	The owl belongs to the family Strigidae, which...	The owl belongs to the bird family known as St...	The owl belongs to the bird family called Stri...	As a language model AI, I can provide expert i..
4	Q167	how many people were killed in the oklahoma ci...	The Oklahoma blast claimed 168 lives, includin...	The Oklahoma City bombing, which occurred on A...	The Oklahoma City bombing, which occurred on A...	The Oklahoma City bombing occurred on April 19...	The Oklahoma City bombing, which occurred on A...	As an A language model, I can provide you wit..

5 rows × 23 columns

If needed because the dataset was store on csv, here is a command that convert the string into list.

```
dfsmall['rouge_role'] = dfsmall.apply(lambda row: convert_string_to_list(row.rouge_role), axis = 1)
dfsmall['bert_role'] = dfsmall.apply(lambda row: convert_string_to_list(row.bert_role), axis = 1)
dfsmall['rouge_tone'] = dfsmall.apply(lambda row: convert_string_to_list(row.rouge_role), axis = 1)
dfsmall['bert_tone'] = dfsmall.apply(lambda row: convert_string_to_list(row.bert_role), axis = 1)
```

Otherwise, continue below:

```
dfsmall['rouge_index_role'] = dfsmall.apply(lambda row: row.rouge_role.index(max(row.rouge_role)), axis = 1)
dfsmall['bert_index_role'] = dfsmall.apply(lambda row: row.bert_role.index(max(row.bert_role)), axis = 1)
```

```
dfsmall['rouge_index_tone'] = dfsmall.apply(lambda row: row.rouge_tone.index(max(row.rouge_tone)), axis = 1)
dfsmall['bert_index_tone'] = dfsmall.apply(lambda row: row.bert_tone.index(max(row.bert_tone)), axis = 1)
```

```
dfsmall.head()
```

	question_id	question	answer	direct_answer	writer_answer	student_answer	teacher_answer	expert_answer
0	Q11	how big is bmc software in houston, tx	Employing over 6,000, BMC is often credited wi...	BMC Software is a substantial presence in Hous...	BMC Software is a prominent technology company...	As a student, I do not have access to current ...	As a teacher, I don't have access to real-time...	As an A language model, I don't have access t..
1	Q48	how long was i love lucy on the air	The black-and-white series originally ran from...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for a total of six ...	I Love Lucy was on the air for six seasons, fr...	I Love Lucy was on the air for a total of six ...	As an A language model, I can provide you wit..
2	Q66	how did armando christian perez become famous	Armando Pérez (born January 15, 1981), better ...	Armando Christian Perez, better known by his s...	Armando Christian Perez, more famously known a...	Armando Christian Perez, also known as Pitbull...	Armando Christian Perez, better known as Pitbu...	Armando Christian Perez, known by his stage na..
3	Q112	what bird family is the owl	Owls are a group of birds that belong to the o...	The owl belongs to the Strigidae family, chara...	The owl belongs to the family Strigidae, which...	The owl belongs to the bird family known as St...	The owl belongs to the bird family called Stri...	As a language model AI, I can provide expert i..
4	Q167	how many people were killed in the oklahoma	The Oklahoma blast claimed 168 lives,	The Oklahoma City bombing, which occurred on A...	The Oklahoma City bombing, which occurred on A...	The Oklahoma City bombing occurred on April 19...	The Oklahoma City bombing, which occurred on A...	As an A language model, I can provide you wit..

```
save_drive(dfsmall,'dfsmall_vf')
#https://drive.google.com/file/d/1-4gq-yyh0xlUpNZ_MhCmdQ2d5hQ7lEXx/view?usp=sharing
#1-4gq-yyh0xlUpNZ_MhCmdQ2d5hQ7lEXx
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", f
'success'
```

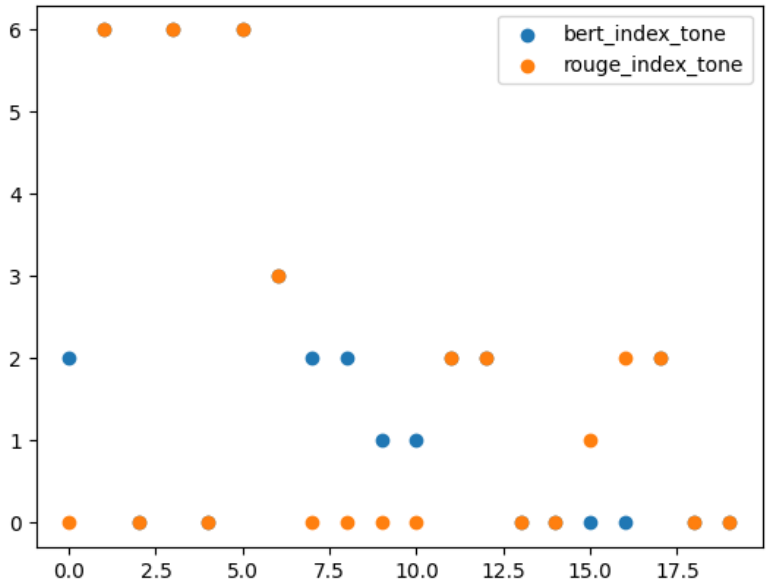
```
url = 'https://drive.google.com/uc?id={}'.format('1-4gq-yyh0xlUpNZ_MhCmdQ2d5hQ7lEXx')
dfsmall = pd.read_csv(url)
```

```
dfsmall['rouge_role'] = dfsmall.apply(lambda row: convert_string_to_list(row.rouge_role), axis = 1)
dfsmall['bert_role'] = dfsmall.apply(lambda row: convert_string_to_list(row.bert_role), axis = 1)
dfsmall['rouge_tone'] = dfsmall.apply(lambda row: convert_string_to_list(row.rouge_role), axis = 1)
dfsmall['bert_tone'] = dfsmall.apply(lambda row: convert_string_to_list(row.bert_role), axis = 1)
```

Interpretation

```
plt.scatter(range(len(dfsmall.bert_index_role)),dfsmall.bert_index_role,label='bert_index_role')
plt.scatter(range(len(dfsmall.rouge_index_role)),dfsmall.rouge_index_role,label='rouge_index_role')
plt.legend()
plt.show()
```


```
plt.scatter(range(len(dfsmall.bert_index_tone)),dfsmall.bert_index_tone,label='bert_index_tone')
plt.scatter(range(len(dfsmall.rouge_index_tone)),dfsmall.rouge_index_tone,label='rouge_index_tone')
plt.legend()
plt.show()
```

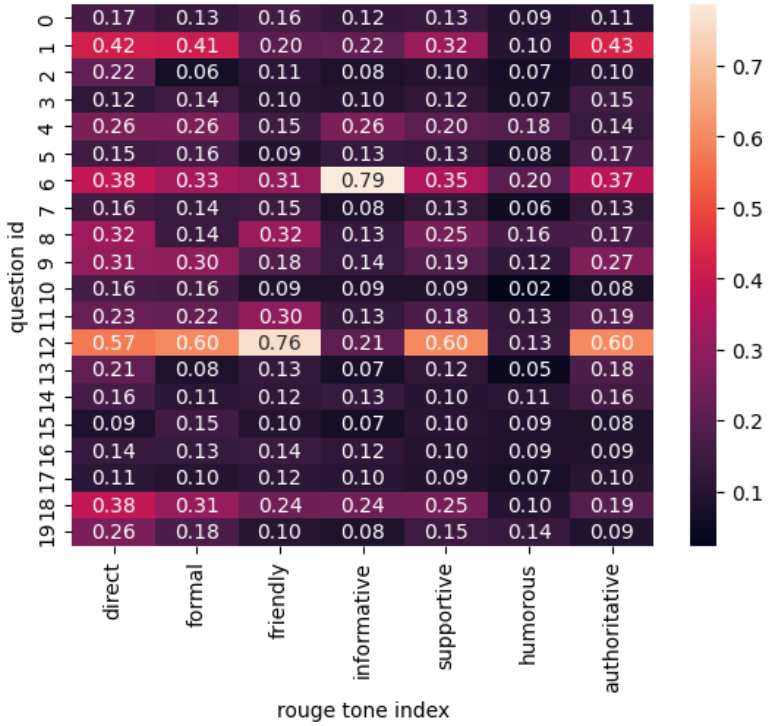



```
roles = ['direct','writer','student','teacher','expert','child']
tones = ['direct','formal','friendly','informative','supportive','humorous','authoritative']


rouge_role_s = np.array([np.array(xi) for xi in dfsmall.rouge_role])
rouge_tone_s = np.array([np.array(xi) for xi in dfsmall.rouge_tone])
bert_role_s = np.array([np.array(xi) for xi in dfsmall.bert_role])
bert_tone_s = np.array([np.array(xi) for xi in dfsmall.bert_tone])


rouge_tone_s[0]
```

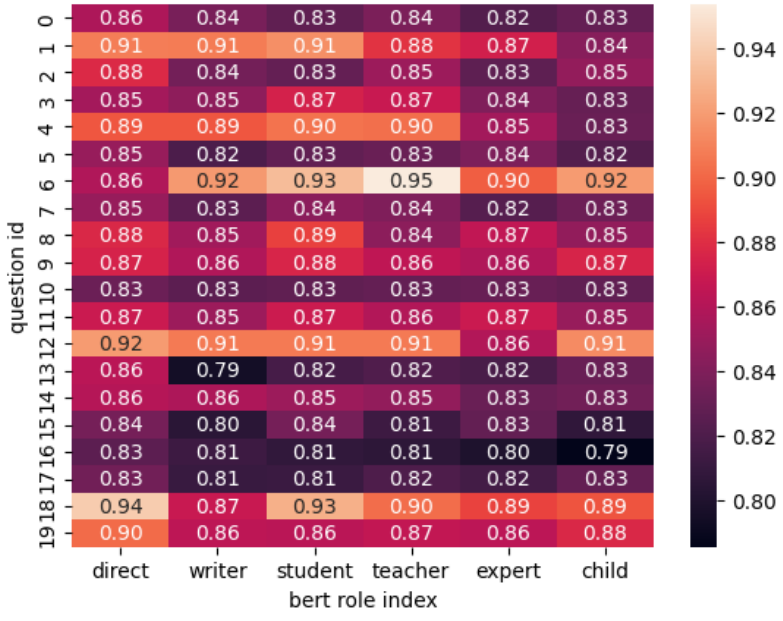
```
array([0.16666667, 0.128     , 0.16      , 0.12307692, 0.13186813,
       0.09022556, 0.10638298])
```

```
ax = sns.heatmap(rouge_role_s,annot=True,fmt=".2f",xticklabels=roles)
ax.set(xlabel="rouge role index", ylabel="question id")
plt.show()
```

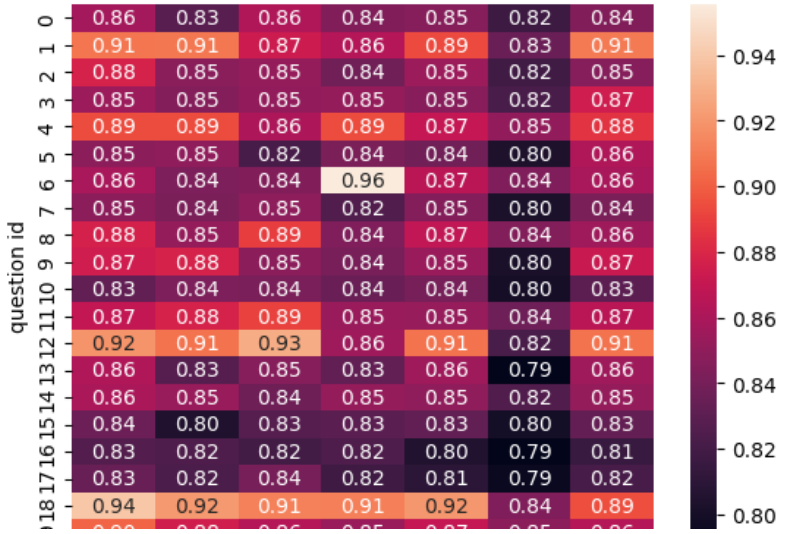

```
ax = sns.heatmap(rouge_tone_s,annot=True,fmt=".2f",xticklabels=tones)
ax.set(xlabel="rouge tone index", ylabel="question id")
plt.show()
```




```
ax = sns.heatmap(bert_role_s,annot=True,fmt=".2f",xticklabels=roles)
ax.set(xlabel="bert role index", ylabel="question id")
plt.show()
```




```
ax = sns.heatmap(bert_tone_s,annot=True,fmt=".2f",xticklabels=tones)
ax.set(xlabel="bert tone index", ylabel="question id")
plt.show()
```



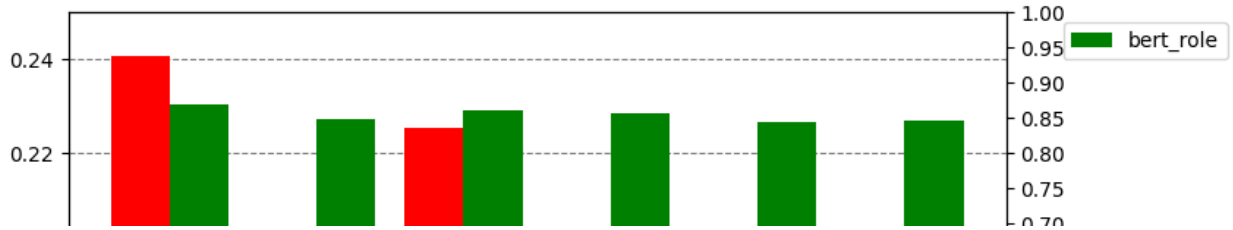
stats

```
x1,y1=[i for i in range(10)],[i for i in range(10,0,-1)]
x2,y2=[i for i in range(10)],[i for i in range(10)]
x3,y3=[i for i in range(10)],[i**2 for i in range(10)]
```

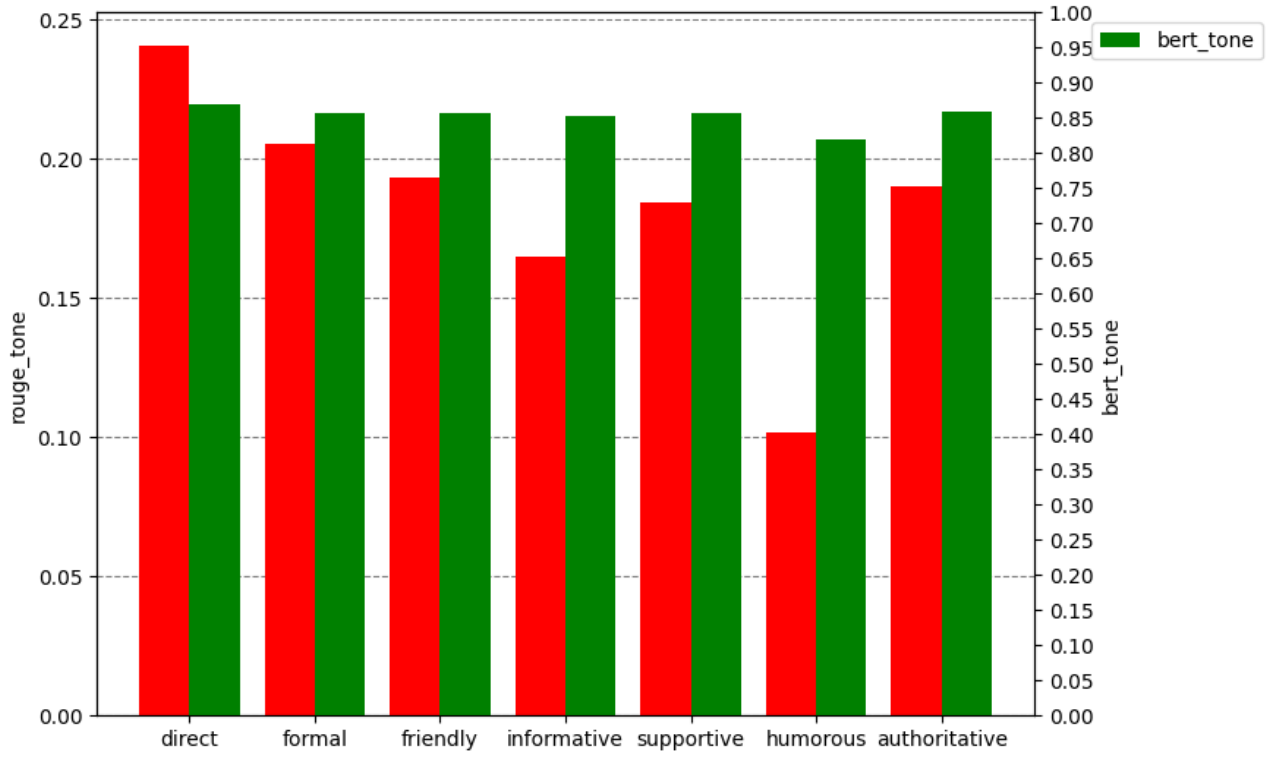
```
np.mean(bert_tone_s,axis=0)
```

```
array([0.86894216, 0.85713278, 0.85774731, 0.85301905, 0.85616336,
       0.81934012, 0.85932439])
```

```
data = [np.mean(rouge_role_s,axis=0),
np.mean(bert_role_s,axis=0)]
X = np.arange(6)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax2 = ax.twinx()
ax.bar(X - 0.2, data[0], color = 'r', width = 0.4,label='rouge_role')
ax2.bar(X + 0.2, data[1], color = 'g', width = 0.4,label='bert_role')

ax.set_ylim([0.1, 0.25])
ax2.set_ylim([0.8, 0.9])

ax.set_axisbelow(True)
ax.yaxis.grid(color='gray', linestyle='dashed')
ax.set_xticks(range(6))
ax.set_xticklabels(roles)
plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.yticks(np.linspace(0,1,21))

plt.show()
```




```
data = [np.mean(rouge_tone_s,axis=0),
np.mean(bert_tone_s,axis=0)]
X = np.arange(7)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax2 = ax.twinx()
ax.bar(X - 0.2, data[0], color = 'r', width = 0.4,label='rouge_tone')
ax2.bar(X + 0.2, data[1], color = 'g', width = 0.4,label='bert_tone')
ax.set_axisbelow(True)
ax.yaxis.grid(color='gray', linestyle='dashed')
ax.set_xticks(range(7))
ax.set_xticklabels(tones)
ax.set_ylabel('rouge_tone')
ax2.set_ylabel('bert_tone')
plt.legend(bbox_to_anchor=(1.05, 1.0),loc='upper left')
plt.yticks(np.linspace(0,1,21))
plt.show()
```



Full dataset

df-n

```
df2 = df.copy()
df2 = df2[120:220]
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 2638 to 2511
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   question_id  100 non-null    object
```

```
 1   question     100 non-null    object
 2   answer       100 non-null    object
dtypes: object(3)
memory usage: 3.1+ KB
```

```
df2.head()
```

	question_id	question	answer
2638	Q2958	Who is the highest scoring NBA player	Wilt Chamberlain holds the all-time records fo...
2649	Q2961	what is direct marketing channel	Direct marketing is a channel-agnostic form of...
2662	Q2969	what is disney's magic kingdom	Magic Kingdom Park, also known as Magic Kingdo...
2669	Q2974	what is the name of the family who own the bil...	Still owned by one of Vanderbilt's descendants...
2672	Q2978	who was the congressman who was caught with an...	On March 10, 2008, The New York Times reported...

save and load

save df1

```
save_drive(df1,'df1_v1')
#https://drive.google.com/file/d/1-761raLbXAhYjNNJrSScgu3ixN-raMek/view?usp=sharing
#1-761raLbXAhYjNNJrSScgu3ixN-raMek
```

```
Mounted at /content/drive
'success'
```

save df2

```
save_drive(df2,'df2_v1')
#https://drive.google.com/file/d/1IPgdBGH6N5kB-JgfXQCLO9wwx373Ye3e/view?usp=sharing
#1IPgdBGH6N5kB-JgfXQCLO9wwx373Ye3e
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", f
'success'
```

save df (1 et 2 union)

```
save_drive(df,'df')
#https://drive.google.com/file/d/1_YIHujAz-SIXL2iCUEV8V5LWj6jOFPG-/view?usp=sharing
#1_YIHujAz-SIXL2iCUEV8V5LWj6jOFPG-
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", f
'success'
```

load df1

```
url = 'https://drive.google.com/uc?id={}'.format('1-761raLbXAhYjNNJrSScgu3ixN-raMek')
df1 = pd.read_csv(url)
```

```
df1.head()
```

load df2

```
url = 'https://drive.google.com/uc?id={}'.format('1IPgdBGH6N5kB-JgfXQCLO9wwx373Ye3e')
df2 = pd.read_csv(url)
```

```
df2.head()
```

load df union df1 df2

```
url = 'https://drive.google.com/uc?id={}'.format('1_YIHujAz-SIXL2iCUEV8V5LWj6jOFPG-')
df = pd.read_csv(url)
```

```
df.head()
```

	question_id	question	answer	direct_answer	writer_answer	student_answer	teacher_answer	expert_answe
0	Q813	How Works Diaphragm Pump	A diaphragm pump (also known as a Membrane pum...	A diaphragm pump is a type of positive displac...	A diaphragm pump, also known as a membrane pum...	As a student, a diaphragm pump works by using ...	A diaphragm pump is a type of positive displac...	As an expert can explain ho a diaphragm pu
1	Q879	What happened to "The Glades" tv series	The show has been renewed for a fourth season.	"The Glades" TV series was unfortunately cance...	"The Glades" was a television crime drama seri...	"The Glades" TV series was a crime drama that ...	"The Glades" TV series was an American crime d...	As an language mode I can provic informa
2	Q931	what kind of company is Microsoft?	Microsoft Corporation is an American multinati...	Microsoft is a multinational technology compan...	Microsoft is a multinational technology compan...	As a student, Microsoft is a multinational tec...	As a teacher, Microsoft is primarily known as ...	As an language mode I can provic informa
3	Q985	what area code is 949	Area code 949 is an area code in California th...	The area code 949 is primarily associated with...	Area code 949 is associated with the state of ...	As a student, I can answer that the area code ...	As a teacher, I can tell you that the area cod...	As an expert can confirm th the area code
4	Q1050	who sings the song never ending story	The English version was performed by Limahl an...	The song "Never Ending Story" is sung by Limahl.	The song "Never Ending Story" was originally p...	The song "Never Ending Story" is originally pe...	The song "Never Ending Story" was originally p...	The song "Nev Ending Stor was originally p

5 rows × 23 columns

```
df.drop(columns=['Unnamed: 0'], inplace=True)
```

Convert

```
df['rouge_role'] = df.apply(lambda row: convert_string_to_list(row.rouge_role), axis = 1)
df['bert_role'] = df.apply(lambda row: convert_string_to_list(row.bert_role), axis = 1)
df['rouge_tone'] = df.apply(lambda row: convert_string_to_list(row.rouge_tone), axis = 1)
df['bert_tone'] = df.apply(lambda row: convert_string_to_list(row.bert_tone), axis = 1)
```

```
df['rouge_index_role'] = df.apply(lambda row: int(row.rouge_index_role), axis = 1)
```

Combine df

```
df = pd.concat([df1, df2])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 0 to 99
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
```

```
 ---  ------                 --------------  -----
  0   question_id            200 non-null    object
  1   question               200 non-null    object
  2   answer                 200 non-null    object
  3   direct_answer          200 non-null    object
  4   writer_answer          200 non-null    object
  5   student_answer         200 non-null    object
  6   teacher_answer         200 non-null    object
  7   expert_answer          200 non-null    object
  8   child_answer           200 non-null    object
  9   formal_answer          200 non-null    object
  10  friendly_answer        200 non-null    object
  11  informative_answer     200 non-null    object
  12  supportive_answer      200 non-null    object
  13  humorous_answer        200 non-null    object
  14  authoritative_answer   200 non-null    object
  15  rouge_role             200 non-null    object
  16  rouge_tone             200 non-null    object
  17  bert_role              200 non-null    object
  18  bert_tone              200 non-null    object
  19  rouge_index_role       200 non-null    int64
  20  bert_index_role        200 non-null    int64
  21  rouge_index_tone       200 non-null    int64
  22  bert_index_tone        200 non-null    int64
dtypes: int64(4), object(19)
memory usage: 37.5+ KB
```

Generation

Generation direct answer

```
df2['direct_answer'] = df2.apply(lambda row: generate(row.question,'',''), axis = 1)
```

Generation role answers

```
df1['writer_answer'] = df1.apply(lambda row: generate(row.question,'Writer',''), axis = 1)
```

```
df1['student_answer'] = df1.apply(lambda row: generate(row.question,'student',''), axis = 1)
```

```
df1['teacher_answer'] = df1.apply(lambda row: generate(row.question,'teacher',''), axis = 1)
```

```
df1['expert_answer'] = df1.apply(lambda row: generate(row.question,'expert',''), axis = 1)
```

```
df1['child_answer'] = df1.apply(lambda row: generate(row.question,'child',''), axis = 1)
```

Generation tone answers

```
df1['formal_answer'] = df1.apply(lambda row: generate(row.question,'','formal'), axis = 1)
```

```
df1['friendly_answer'] = df1.apply(lambda row: generate(row.question,'','friendly'), axis = 1)
```

```
df1['informative_answer'] = df1.apply(lambda row: generate(row.question,'','informative'), axis = 1)
```

```
df1['supportive_answer'] = df1.apply(lambda row: generate(row.question,'','supportive'), axis = 1)
```

```
df1['humorous_answer'] = df1.apply(lambda row: generate(row.question,'','humorous'), axis = 1)
```

```
df1['authoritative_answer'] = df1.apply(lambda row: generate(row.question,'','authoritative'), axis = 1)
```

Metric calculation

Rouge role

```
df1['rouge1'] = df1.apply(lambda row: rouge.compute(predictions=[row.direct_answer],references=[row.answer])['roug

df1['rouge2'] = df1.apply(lambda row: rouge.compute(predictions=[row.writer_answer],references=[row.answer])['roug

df1['rouge3'] = df1.apply(lambda row: rouge.compute(predictions=[row.student_answer],references=[row.answer])['rou

df1['rouge4'] = df1.apply(lambda row: rouge.compute(predictions=[row.teacher_answer],references=[row.answer])['rou

df1['rouge5'] = df1.apply(lambda row: rouge.compute(predictions=[row.expert_answer],references=[row.answer])['roug

df1['rouge6'] = df1.apply(lambda row: rouge.compute(predictions=[row.child_answer],references=[row.answer])['rouge

# Create a new column 'merged_rouge' containing the merged values as lists
df2['rouge_role'] = df2.apply(lambda row: [row['rouge1'], row['rouge2'], row['rouge3'], row['rouge4'], row['rouge5

# Drop the individual 'rouge' columns if needed
df2.drop(columns=['rouge1', 'rouge2', 'rouge3', 'rouge4', 'rouge5', 'rouge6'], inplace=True)
```

Rouge tone

```
df1['rouge0'] = df1.apply(lambda row: rouge.compute(predictions=[row.direct_answer],references=[row.answer])['roug

df1['rouge1'] = df1.apply(lambda row: rouge.compute(predictions=[row.formal_answer],references=[row.answer])['roug

df1['rouge2'] = df1.apply(lambda row: rouge.compute(predictions=[row.friendly_answer],references=[row.answer])['ro

df1['rouge3'] = df1.apply(lambda row: rouge.compute(predictions=[row.informative_answer],references=[row.answer])[

df1['rouge4'] = df1.apply(lambda row: rouge.compute(predictions=[row.supportive_answer],references=[row.answer])['

df1['rouge5'] = df1.apply(lambda row: rouge.compute(predictions=[row.humorous_answer],references=[row.answer])['ro

df1['rouge6'] = df1.apply(lambda row: rouge.compute(predictions=[row.authoritative_answer],references=[row.answer]

# Create a new column 'merged_rouge' containing the merged values as lists
df1['rouge_tone'] = df1.apply(lambda row: [row['rouge0'],row['rouge1'], row['rouge2'], row['rouge3'], row['rouge4'

# Drop the individual 'rouge' columns if needed
df1.drop(columns=['rouge0','rouge1', 'rouge2', 'rouge3', 'rouge4', 'rouge5', 'rouge6'], inplace=True)
```

Bert role

```
df1['bert1'] = df1.apply(lambda row: bert.compute(predictions=[row.direct_answer],references=[row.answer],lang='en
```

```
Downloading (…)lve/main/config.json: 100%  482/482 [00:00<00:00, 18.1kB/s]
Downloading (…)olve/main/vocab.json: 100%  899k/899k [00:00<00:00, 1.33MB/s]
Downloading (…)olve/main/merges.txt: 100%  456k/456k [00:00<00:00, 20.3MB/s]
Downloading model.safetensors: 100%  1.42G/1.42G [00:23<00:00, 61.5MB/s]
Some weights of RobertaModel were not initialized from the model checkpoint at roberta-large and are newly ir
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference
```

```
df1['bert2'] = df1.apply(lambda row: bert.compute(predictions=[row.writer_answer],references=[row.answer],lang='en
```

```
df1['bert3'] = df1.apply(lambda row: bert.compute(predictions=[row.student_answer],references=[row.answer],lang='e

df1['bert4'] = df1.apply(lambda row: bert.compute(predictions=[row.teacher_answer],references=[row.answer],lang='e

df1['bert5'] = df1.apply(lambda row: bert.compute(predictions=[row.expert_answer],references=[row.answer],lang='en

df1['bert6'] = df1.apply(lambda row: bert.compute(predictions=[row.child_answer],references=[row.answer],lang='en'

# Create a new column 'merged_rouge' containing the merged values as lists
df1['bert_role'] = df1.apply(lambda row: [row['bert1'], row['bert2'], row['bert3'], row['bert4'], row['bert5'], ro

# Drop the individual 'rouge' columns if needed
df1.drop(columns=['bert1', 'bert2', 'bert3', 'bert4', 'bert5', 'bert6'], inplace=True)
```

Bert tone

```
df1['bert0'] = df1.apply(lambda row: bert.compute(predictions=[row.direct_answer],references=[row.answer],lang='en

df1['bert1'] = df1.apply(lambda row: bert.compute(predictions=[row.formal_answer],references=[row.answer],lang='en

df1['bert2'] = df1.apply(lambda row: bert.compute(predictions=[row.friendly_answer],references=[row.answer],lang='

df1['bert3'] = df1.apply(lambda row: bert.compute(predictions=[row.informative_answer],references=[row.answer],lan

df1['bert4'] = df1.apply(lambda row: bert.compute(predictions=[row.supportive_answer],references=[row.answer],lang

df1['bert5'] = df1.apply(lambda row: bert.compute(predictions=[row.humorous_answer],references=[row.answer],lang='

df1['bert6'] = df1.apply(lambda row: bert.compute(predictions=[row.authoritative_answer],references=[row.answer],l

# Create a new column 'merged_rouge' containing the merged values as lists
df1['bert_tone'] = df1.apply(lambda row: [row['bert0'],row['bert1'], row['bert2'], row['bert3'], row['bert4'], row

# Drop the individual 'rouge' columns if needed
df1.drop(columns=['bert0','bert1', 'bert2', 'bert3', 'bert4', 'bert5', 'bert6'], inplace=True)
```

Plotting

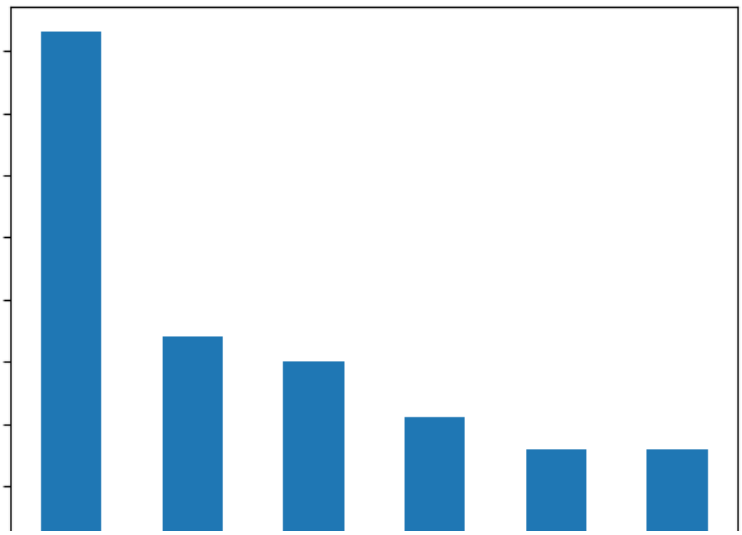
```
rouge_role = np.array([np.array(xi) for xi in df.rouge_role])
rouge_tone = np.array([np.array(xi) for xi in df.rouge_tone])
bert_role = np.array([np.array(xi) for xi in df.bert_role])
bert_tone = np.array([np.array(xi) for xi in df.bert_tone])
```

Role

Best index choice

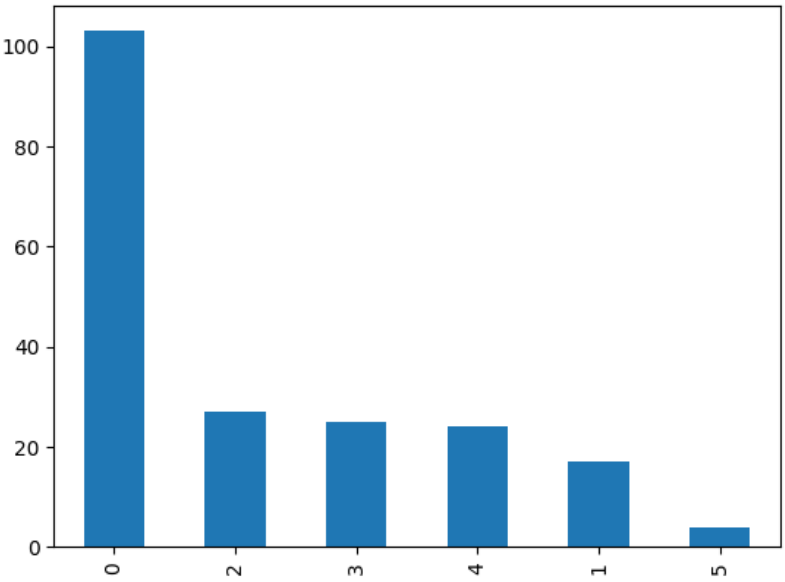
```
df['rouge_index_role'] = df.apply(lambda row: row.rouge_role.index(max(row.rouge_role)), axis = 1)
df['bert_index_role']  = df.apply(lambda row: row.bert_role.index(max(row.bert_role)), axis = 1)


print(df.rouge_index_role.value_counts())
print(df.rouge_index_role.value_counts(normalize=True))
df.rouge_index_role.value_counts().plot(kind='bar')
```

```
0    83
2    34
3    30
4    21
1    16
5    16
Name: rouge_index_role, dtype: int64
0    0.415
2    0.170
3    0.150
4    0.105
1    0.080
5    0.080
Name: rouge_index_role, dtype: float64
<Axes: >
```



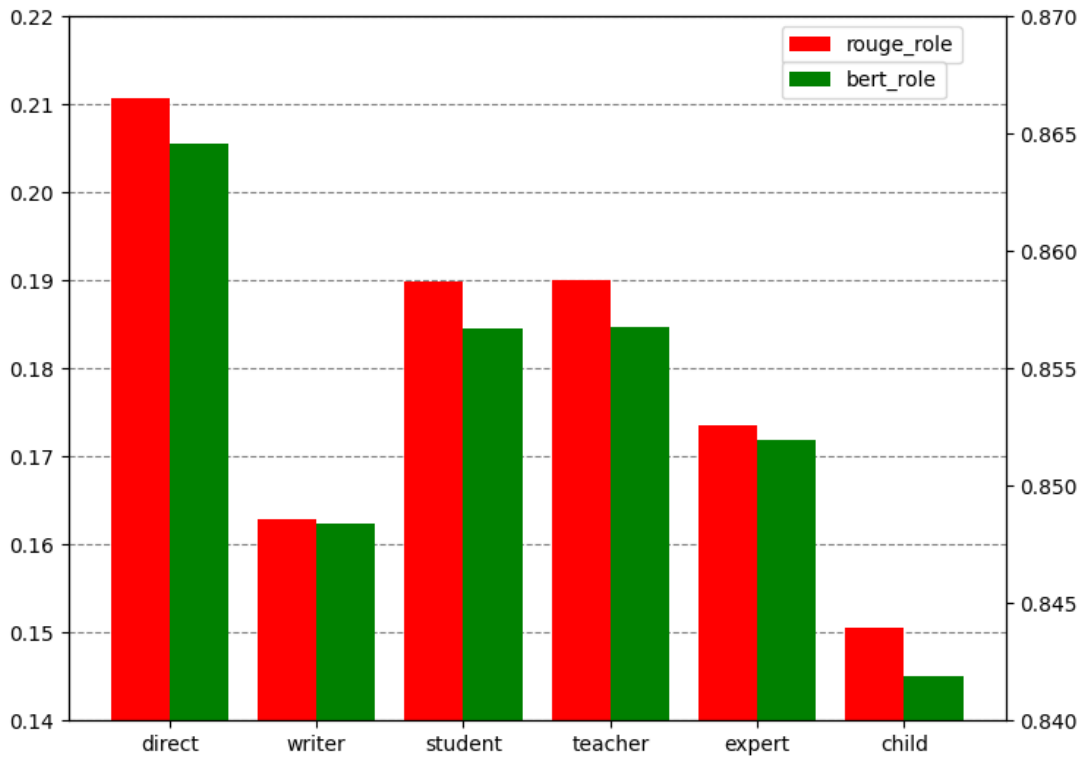

```
print(df.bert_index_role.value_counts())
print(df.bert_index_role.value_counts(normalize=True))
df.bert_index_role.value_counts().plot(kind='bar')
```

```
0    103
2     27
3     25
4     24
1     17
5      4
Name: bert_index_role, dtype: int64
0    0.515
2    0.135
3    0.125
4    0.120
1    0.085
5    0.020
Name: bert_index_role, dtype: float64
<Axes: >
```



Average analysis

```
data = [np.mean(rouge_role,axis=0),
np.mean(bert_role,axis=0)]
X = np.arange(6)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax2 = ax.twinx()

ax.bar(X - 0.2, data[0], color = 'r', width = 0.4,label='rouge_role')
ax2.bar(X + 0.2, data[1], color = 'g', width = 0.4,label='bert_role')

ax.set_ylim([0.14, 0.22])
ax2.set_ylim([0.84, 0.87])

ax.set_axisbelow(True)

ax.yaxis.grid(color='gray', linestyle='dashed')

ax.set_xticks(range(6))
ax.set_xticklabels(roles)

#plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
ax.legend(bbox_to_anchor=(0.75, 1.0), loc='upper left')
ax2.legend(bbox_to_anchor=(0.75, 0.95), loc='upper left')

plt.show()
```

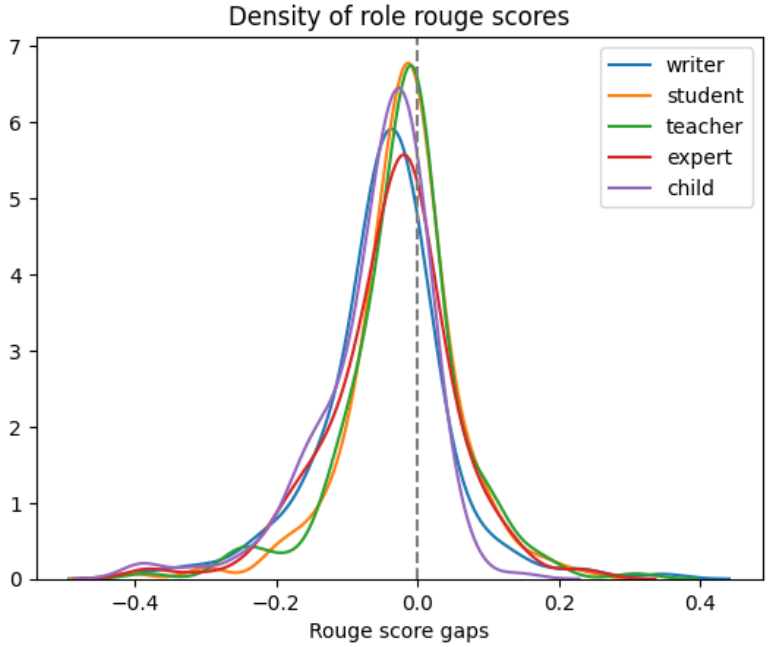


Gap analysis

```
temp = rouge_role.transpose()
gap_rouge_role = [temp[i]-temp[0] for i in range(len(temp))]


temp = bert_role.transpose()
gap_bert_role = [temp[i]-temp[0] for i in range(len(temp))]


for i in range(1,len(gap_rouge_role)):
  # seaborn histogram
  sns.kdeplot(gap_rouge_role[i],label=roles[i])
```

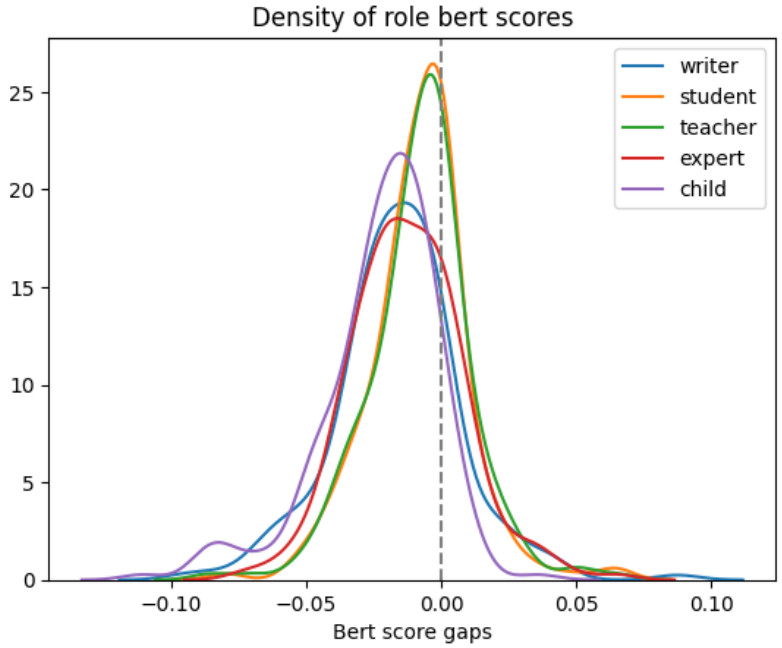
```
  # Add labels
plt.title('Density of role rouge scores')
plt.xlabel('Rouge score gaps')
plt.ylabel('')
plt.axvline(x = 0, color = 'grey', linestyle='dashed')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7aebbc90dc30>
```




```
for i in range(1,len(gap_bert_role)):
  # seaborn histogram
  sns.kdeplot(gap_bert_role[i],label=roles[i])

  # Add labels
plt.title('Density of role bert scores')
plt.xlabel('Bert score gaps')
plt.ylabel('')
plt.axvline(x = 0, color = 'grey', linestyle='dashed')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7aebbc7efb80>
```



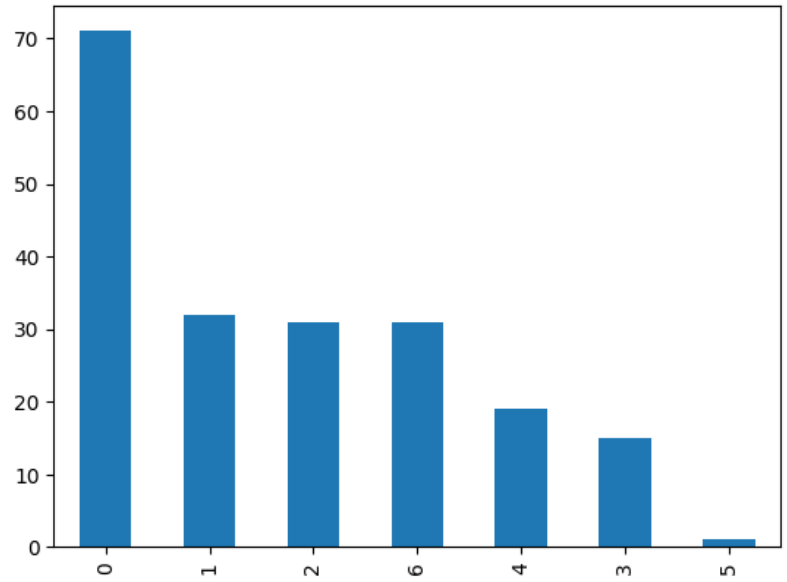
Tones

Best index choice

```
df['rouge_index_tone'] = df.apply(lambda row: row.rouge_tone.index(max(row.rouge_tone)), axis = 1)
df['bert_index_tone']  = df.apply(lambda row: row.bert_tone.index(max(row.bert_tone)), axis = 1)
```

```
print(df.rouge_index_tone.value_counts())
print(df.rouge_index_tone.value_counts(normalize=True))
df.rouge_index_tone.value_counts().plot(kind='bar')
```

```
0    71
1    32
2    31
6    31
4    19
3    15
5     1
Name: rouge_index_tone, dtype: int64
0    0.355
1    0.160
2    0.155
6    0.155
4    0.095
3    0.075
5    0.005
Name: rouge_index_tone, dtype: float64
<Axes: >
```




```
print(df.bert_index_tone.value_counts())
print(df.bert_index_tone.value_counts(normalize=True))
df.bert_index_tone.value_counts().plot(kind='bar')
```

```
0    71
6    38
1    36
2    25
4    16
3    14
Name: bert_index_tone, dtype: int64
0    0.355
6    0.190
1    0.180
2    0.125
4    0.080
3    0.070
Name: bert_index_tone, dtype: float64
<Axes: >
```

Average analysis

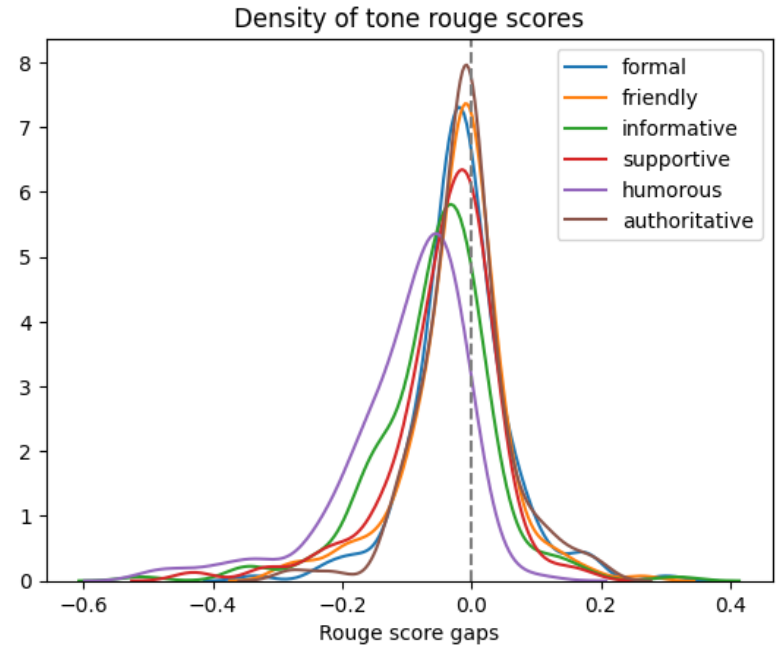
```
data = [np.mean(rouge_tone,axis=0),
np.mean(bert_tone,axis=0)]
X = np.arange(7)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax2 = ax.twinx()

ax.bar(X - 0.2, data[0], color = 'r', width = 0.4,label='rouge_role')
ax2.bar(X + 0.2, data[1], color = 'g', width = 0.4,label='bert_role')

ax.set_ylim([0.1, 0.22])
ax2.set_ylim([0.82, 0.87])

ax.set_axisbelow(True)

ax.yaxis.grid(color='gray', linestyle='dashed')

ax.set_xticks(range(7))
ax.set_xticklabels(tones)

#plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
ax.legend(bbox_to_anchor=(0.75, 1.0), loc='upper left')
ax2.legend(bbox_to_anchor=(0.75, 0.95), loc='upper left')

plt.show()
```

Gap analysis

```
temp = rouge_tone.transpose()
gap_rouge_tone = [temp[i]-temp[0] for i in range(len(temp))]
temp = bert_tone.transpose()
gap_bert_tone = [temp[i]-temp[0] for i in range(len(temp))]
```

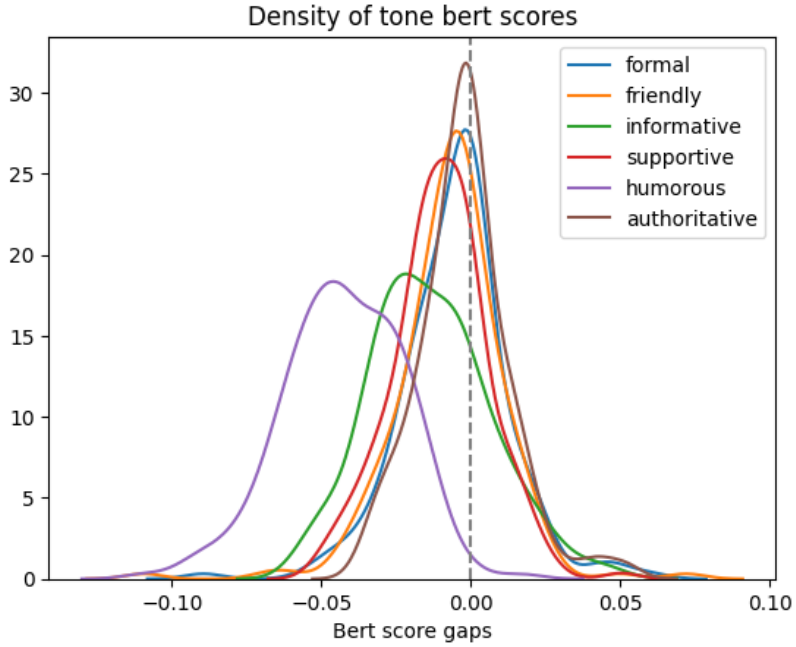
```
for i in range(1,len(gap_rouge_tone)):
  # seaborn histogram
  sns.kdeplot(gap_rouge_tone[i],label=tones[i])

  # Add labels
plt.title('Density of tone rouge scores')
plt.xlabel('Rouge score gaps')
plt.ylabel('')
plt.axvline(x = 0, color = 'grey', linestyle='dashed')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7aebbd281ab0>
```




```
for i in range(1,len(gap_bert_tone)):
  # seaborn histogram
  sns.kdeplot(gap_bert_tone[i],label=tones[i])

  # Add labels
plt.title('Density of tone bert scores')
plt.xlabel('Bert score gaps')
plt.ylabel('')
plt.axvline(x = 0, color = 'grey', linestyle='dashed')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7aebbd7ae740>
```



✓ 1s completed at 09:06