KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second Cycle 30.0 credits

Automatic text summarization of French judicial data with pre-trained language models, evaluated by content and factuality metrics

MALO ADLER

Stockholm, Sweden 2024

KTH – Royal Institute of Technology
EECS School

ANFSI, Datalab

Automatic text summarization of French judicial data with pre-trained language models, evaluated by content and factuality metrics

Malo Adler

Supervisor (ANFSI): Jean-Baptiste Delfau
Supervisor (KTH): Mats Nordahl
Examiner: Olov Engwall

February 23, 2024

Abstract

During an investigation carried out by a police officer or a gendarme, audition reports are written, the length of which can be up to several pages. The high-level goal of this thesis is to study various automatic and reliable text summarization methods to help with this time-consuming task. One challenge comes from the specific, French and judicial data that we wish to summarize; and another challenge comes from the need for reliable, factual and invention-free models.

First, this thesis focuses on automatic summarization evaluation, in terms both of content (how well the summary captures essential information of the source text) and factuality (to what extent the summary only includes information from or coherent with the source text). Factuality evaluation, in particular, is of crucial interest when using LLMs for judicial purposes, because of their hallucination risks. Notably, we propose a light variation of SelfCheckGPT, which has a stronger correlation with human judgment (0.743) than the wide-spread BARTScore (0.542), or our study dataset. Other paradigms, such as Question-Answering, are studied in this thesis, which however underperform compared to these.

Then, extractive summarization methods are explored and compared, including one based on graphs via the TextRank algorithm, and one based on greedy optimization. The latter (overlap rate: 0.190, semantic similarity: 0.513) clearly outperforms the base TextRank (overlap rate: 0.172, semantic similarity: 0.506). An improvement of the TextRank with a threshold mechanism is also proposed, leading to a non-negligible improvement (overlap rate: 0.180, semantic similarity: 0.513).

Finally, abstractive summarization, with pre-trained LLMs based on a Transformer architecture, is studied. In particular, several general-purpose and multilingual models (Llama-2, Mistral and Mixtral) were objectively compared on a summarization dataset of judicial procedures from the French police. Results show that the performances of these models are highly related to their size: Llama-2 7B struggles to adapt to uncommon data (overlap rate: 0.083, BARTScore: -3.099), while Llama-2 13B (overlap rate: 0.159, BARTScore: -2.718) and Llama-2 70B (overlap rate: 0.191, BARTScore: -2.479) have proven quite versatile and efficient. To improve the performances of the smallest models, empirical prompt-engineering and parameter-efficient fine-tuning are explored. Notably, our fine-tuned version of Mistral 7B reaches performances comparable to those of much larger models (overlap rate: 0.185, BARTScore: -2.060), without the need for empirical prompt-engineering, and with a linguistic style closer to what is expected.

Keywords. Text summarization · Extractive · Abstractive · Generative AI · Hallucinations · Factuality · Transformer · LLMs · Auto-regressive models · Prompt engineering · Parameter-efficient fine-tuning

Sammanfattning

Under en utredning som görs av en polis eller en gendarm skrivs förhörsprotokoll vars längd kan vara upp till flera sidor. Målet på hög nivå med denna rapport är att studera olika automatiska och tillförlitliga textsammanfattningsmetoder för att hjälpa till med denna tidskrävande uppgift. En utmaning kommer från de specifika franska och rättsliga uppgifter som vi vill sammanfatta; och en annan utmaning kommer från behovet av pålitliga, sakliga och uppfinningsfria modeller.

För det första fokuserar denna rapport på automatisk sammanfattningsutvärdering, både vad gäller innehåll (hur väl sammanfattningen fångar väsentlig information i källtexten) och fakta (i vilken utsträckning sammanfattningen endast innehåller information från eller överensstämmer med källtexten). Faktautvärdering, i synnerhet, är av avgörande intresse när man använder LLM för rättsliga ändamål, på grund av deras hallucinationsrisker. Vi föreslår särskilt en lätt variant av SelfCheckGPT, som har en starkare korrelation med mänskligt omdöme (0,743) än den utbredda BARTScore (0,542), eller vår studiedatauppsättning. Andra paradigm, såsom Question-Answering, studeras i denna rapport, som dock underpresterar jämfört med dessa.

Sedan utforskas och jämförs extraktiva sammanfattningsmetoder, inklusive en baserad på grafer via TextRank-algoritmen och en baserad på girig optimering. Den senare (överlappning: 0,190, semantisk likhet: 0,513) överträffar klart basen TextRank (överlappning: 0,172, semantisk likhet: 0,506). En förbättring av TextRank med en tröskelmekanism föreslås också, vilket leder till en icke försumbar förbättring (överlappning: 0,180, semantisk likhet: 0,513).

Slutligen studeras abstrakt sammanfattning, med förutbildade LLM baserade på en transformatorarkitektur. I synnerhet jämfördes flera allmänna och flerspråkiga modeller (Llama-2, Mistral och Mixtral) objektivt på en sammanfattningsdatauppsättning av rättsliga förfaranden från den franska polisen. Resultaten visar att prestandan för dessa modeller är starkt relaterade till deras storlek: Llama-2 7B kämpar för att anpassa sig till ovanliga data (överlappning: 0,083, BARTScore: -3,099), medan Llama-2 13B (överlappning: 0,159, BARTScore: -2,718) och Llama-2 70B (överlappning: 0,191, BARTScore: -2,479) har visat sig vara ganska mångsidiga och effektiva. För att förbättra prestandan för de minsta modellerna utforskas empirisk prompt-teknik och parametereffektiv finjustering. Noterbart är att vår finjusterade version av Mistral 7B når prestanda som är jämförbara med de för mycket större modeller (överlappning: 0,185, BARTScore: -2,060), utan behov av empirisk prompt-teknik och med en språklig stil som ligger närmare vad som förväntas .

Nyckelord. Sammanfattning · Extraktiv · Abstrakt · Generativ AI · Hallucinationer · Fakta · Transformer · LLMs · Autoregressiva modeller · Snabb konstruktion · Parametereffektiv finjustering

Acknowledgments

I would first like to warmly thank my academic supervisor Mats Nordahl and my examiner Olov Engwall for their guidance throughout this project.

I would also like to express my gratitude to Jean-Baptiste Delfau from the ANFSI Datalab, who supported me and trusted me at every stage of my project. Providing me with advice, feedback and help whenever I needed it, he was, thanks to his expertise and generosity, an essential support in the conduct of this project.

Also, I would like to thank the whole ANFSI Datalab team, who made this project a pleasant time, full of enriching discussions and moments of conviviality. Thanks in particular to Gaël de Léséleuc de Kérouara, head of the Datalab, for his trust and benevolence.

Finally, I would like to sincerely thank my family who have always been there for me, and especially my parents for their daily and unfailing support.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ANFSI	Agence du Numérique des Forces de Sécurité Intérieure
API	Application Programming Interface
BART	Bidirectional and Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from Transformers
BLOOM	BigScience Large Open-science Open-access Multilingual
CPU	Central Processing Unit
FNN	Feed-forward Neural Network
GDPR	General Data Protection Regulation
GHG	Greenhouse Gas
GPT	Generative Pre-trained Transformers
GPU	Graphics Processing Unit
GQA	Grouped-Query Attention
GRU	Gated Recurrent Units
LCS	Longest Common Subsequence
LLM	Large Language Model
LoRA	Low-Rank Adaptation
LSTM	Long Short Term Memory
MoE	Mixture-of-Experts
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
Q&A	Question-Answering
Q&G	Question-Generation
RAG	Retrieval Augmented Generation
RAM	Random Access Memory

RLHF	Reinforcement Learning from Human Feedback
RLAIF	Reinforcement Learning from Artificial Intelligence Feedback
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RNN	Recurrent Neural Network
Seq2Seq	Sequence-to-sequence
T5	Text-To-Text Transfer Transformer
VRAM	Video Random Access Memory
WMD	Word Mover Distance

Chapter 1

Introduction

This Chapter introduces the thesis by briefly presenting the research topic in Section 1.1, and its purpose and motivation in Section 1.2. The research questions are then stated in Section 1.3. A delimitation of the research area is done in Section 1.4. Finally, a few terms used in this thesis are presented in Section 1.5, and the structure of the thesis is outlined in Section 1.6.

1.1 Research topic

Automatic text summarization is the process during which a large text document is condensed into a shorter version, which contains most relevant information of the original document. While it has been a topic of research for decades, automatic text summarization has recently gained a lot of visibility with the rise of new text generation methods, and especially that of Large Language Models (LLMs). Current summarization models have however mainly been designed to handle English data, and to summarize certain types of documents, such as news articles [1] or medical documents [2]. These classical and easily available types of documents do not include sensitive, judicial data such as audition reports from the French police.

Automatic text summarization of French, judicial data is challenging for several reasons. Firstly, due to possible judiciary implications, it is crucial that the proposed summaries highlight the main information of the source text, without inventing or adding any external element (also called hallucination, which LLMs are known to generate [3]). This thesis will therefore first focus on summarization evaluation metrics, both in terms of content and factuality. Secondly, the data is in French language, but most pre-trained language models are usually much more efficient on English language (because trained mainly on English data). Besides, the nature of the data itself is not common: publicly available models have not been trained on judicial data, and therefore do not necessarily know judicial jargon. For this reason, the other main objective of this thesis will be to evaluate how pre-trained language models can perform on this specific data, and how these performances can be improved.

1.2 Purpose and motivation

This thesis is carried out within the ANFSI (French Digital Agency for Internal Security Forces). The operational objective behind this project is to provide agents (police officers and gendarmes) with a reliable and French tool for summarizing audition reports. Each time a person (e.g. a victim, a suspect, or a witness) is interviewed by an agent (when filing a complaint or during an investigation), this agent writes an audition report containing all relevant information. These reports can be quite long (see Section 3.2) and an investigation may include reports of several auditions. The agent then writes a summarized version of the main information contained in this/these report(s), which can then be read by other agents (to save time and avoid them reading several pages), or serve as a basis for the infraction determination. The high-level goal of this project is therefore to help in this very time-consuming synthesis work, by proposing a summary to the agent, who will then have to correct it if necessary, and approve it. Even though this project focuses solely on audition reports, it can also serve as a basis for other similar use cases for the ANFSI (e.g. telephone tappings summarization).

1.3 Research questions

This thesis attempts to answer several distinct yet related research sub-questions, which can be expressed as such:

1. How can the quality of a summarization model be efficiently evaluated, in terms both of content and factuality?

2. What are the performances of open-source, multilingual and general-purpose language models on specific task (text summarization) and data (French, judicial data), according to such evaluation metrics?

3. How can the performances of such models be improved for these specific task and data, with further fine-tuning and prompt engineering techniques?

1.4 Delimitation

Several elements fall outside the research scope of this thesis.

First, the data we seek to summarize are auditions of victims, witnesses or suspects, and are therefore by nature very sensitive. It is not possible, for security reasons, to use proprietary models accessible via APIs, such as ChatGPT, which provide little or no guarantee on the use of user data. This involves considering only open-source models, which can be less efficient.

Likewise, it is impossible for us to host our models on remote servers, or to use GPUs, belonging to Google for example. The development, possible training and inference use of our models have to be done in a controlled and secure manner on our own architectures. The material resources of the Datalab therefore exclude huge open-source LLMs, such as BLOOM [4].

1.5 Terminology

In the remainder of this thesis, we will consider the following terms:

- The **source** (denoted **S**) designates the original text, which we wish to summarize;
- The **prediction** (denoted **P**) designates the summary generated by some model;
- The **reference** (denoted **R**) designates the reference summary, if available.

1.6 Organization of the thesis

Chapter 2 synthesizes the research pre-study of the thesis, including existing methods and their limitations, and related research work. Chapter 3 describes the work done for this thesis, i.e. mainly the followed methods and the conducted experiments. Chapter 4 presents the main results of these experiments, as well as a critical review of these results. Finally, Chapter 5 addresses various topics such as ethics or sustainability of the project and associated technologies.

Chapter 2

Background

This Chapter introduces all relevant background needed to understand or used in this thesis. A brief introduction to text summarization and its different paradigms is presented in Section 2.1. An overview of existing evaluation metrics for summarization is then provided in Section 2.2. The two main paradigms of summarization, i.e. extractive and abstractive, are then discussed in Sections 2.3 and 2.4. Finally, a synthesis of related research work is presented in Section 2.5.

2.1 Introduction to text summarization

Text summarization involves creating a concise version of a text document, while preserving key information and maintaining semantic consistency. Automatic summarization systems therefore aim to condense information while retaining the essential meaning of the source text, the objective being to produce a summary that faithfully reflects the original content while being shorter and easier to understand.

Automatic text summarization can be divided into several categories (see Figure 1), which will be described in the following paragraphs, according to:

- The **nature of the produced summary**: extractive or abstractive.
- The **nature of the source to be summarized**: single-document or multi-document.

2.1.1 Extractive vs abstractive summarization

Extractive summarization

This method consists of selecting sentences from the source, according to their importance, and concatenating them to form a reduced version of the source. In particular, it has the following advantages:

- Few resources (data, GPU, RAM, etc.) are usually required;
- There is no risk of inventions or hallucinations;
- The methods themselves do not impose size limitations.

On the other hand, summaries produced in an extractive manner will certainly lack readability or coherence, and risk forgetting important information. Since producing an abstractive summary was effectively made possible, the extractive approach for final summaries has clearly lost popularity. However, abstractive methods having size constraints (see Section 2.4.4), a hybrid strategy is often considered [5], consisting of first reducing a long text with extractive methods, and then generating an abstractive summary of this reduced version.

Abstractive summarization

This method, for its part, consists of generating a new text, which reformulates the main ideas of the source in new terms. It involves creating new textual data, and thus fits into the very fashionable field of generative AI. It has several significant advantages:

- The produced summaries are closer to a human-written text;
- The generated text is more understandable and easier to read;
- The result can potentially be more exhaustive.

On the other hand, abstractive models are much more expensive and heavier to train and run, and often have a size limit which prevents the processing of texts of several pages. Besides, the main drawback of generative methods, especially damaging for our specific use case, is the risk of hallucination. Due to the nature of the technologies used, there is no guarantee that abstractive models generate true or factual content.

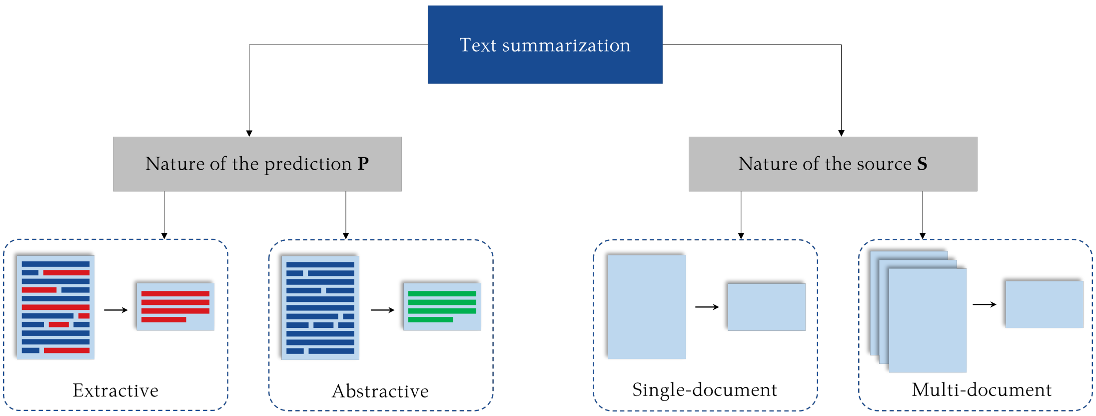


Figure 1: The different summarization paradigms

2.1.2 Single-document vs multi-document summarization

Single-document summarization

In this approach, the goal is to summarize a single textual document: this is the simplest form of summarization. When considering audition reports, it is rare for an investigation to contain only a single audition and therefore a single report, even if this can sometimes happen. However, since multi-document summarization heavily relies on single-document summarization, it is natural to look at it first. For this reason, this project is dedicated to single-document summarization.

Multi-document summarization

For this more general task, the goal is to cross the information present in several textual documents, and to summarize this information in a single text: more challenging than single-document summarization, it also has a broader scope of application.

2.2 Evaluation metrics

When creating automatic summarization models, evaluation is crucial: without clear assessment, there is no way to compare and discriminate between models. Human evaluation is generally considered as the best possible judgment, but it has two major drawbacks: it is very time-consuming and therefore not usable on a large scale, and it is inherently subjective. Creating automatic and objective evaluation metrics is thus decisive, yet very challenging. There are indeed several key elements to take into account when evaluating a prediction (i.e. a summary), as for instance:

- **Content**: How well does the prediction capture essential information of the source?
- **Factuality**: Does the prediction include information that is not present in the source?
- **Language**: How is the prediction written, in grammatical and stylistic terms?
- **Concision**: How condensed is the information in the prediction?

In line with the expectations of the organization this project was carried out in, we focused on content and factuality evaluation, which we considered as more decisive than language and concision.

Evaluation metrics can be categorized according to their evaluation objective (content, factuality, etc.), their evaluation paradigm (overlap, semantic similarity, Question Answering, etc.), but also according to the entities compared. We thus distinguish:

1. **Reference-based metrics**: As in a typical supervised framework, we compare a prediction to a reference (i.e. a label). For text summarization, a reference is a human-written summary. This is very practical and useful for research and development, when you have access to a labeled dataset, but unusable in production.

2. **Source-based metrics**: Conversely, these do not require labels. We directly compare the prediction to the source (i.e. the original text to be summarized).

Figure 2 categorizes the metrics presented below according to their evaluation paradigm, as well as their comparison source. An interesting point to notice is the relationship between the evaluation objective and the comparison source: the metrics focusing on content (overlap of tokens/characters, semantic similarity) are reference-based, while those focusing on factuality (text generation, Question Answering, LLM-based) are source-based. Indeed, a reference tells you which information of a text is central and worth appearing in the summary (content), which the text itself cannot indicate. In contrast, a reference cannot say if a given information was or was not present in the text (factuality), but the text can.

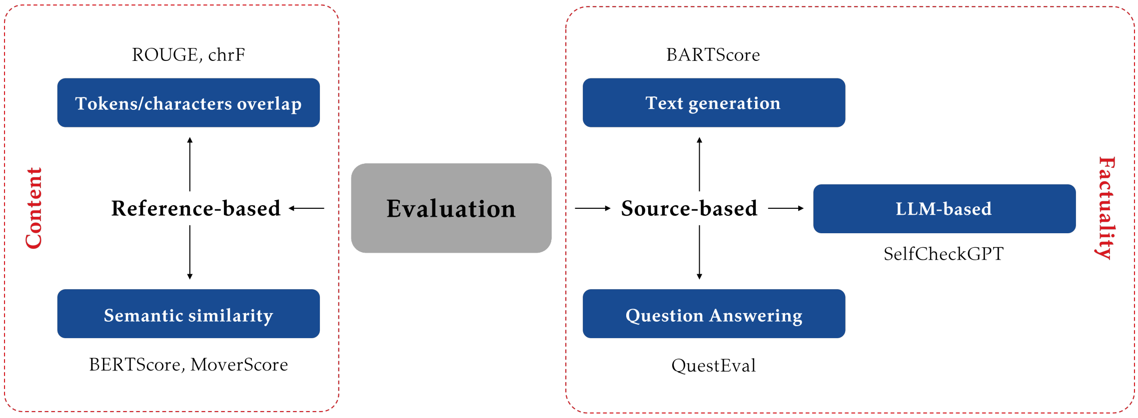


Figure 2: Categorization of the metrics retained

When creating evaluation metrics, the question of the evaluation of these evaluation metrics themselves is a central question: even if, conceptually, a given metric seems interesting and relevant, nothing *a priori* assures us that this metric is actually better than more basic ones. In the literature [6], human judgment is considered as the gold judgment: a metric is considered efficient if it has a high correlation with human judgment. Comparing two evaluation metrics therefore amounts to comparing their correlation with human judgment. Besides, human judgment can itself consist of several components (content, factuality, language, concision), each of which can be compared from a correlation perspective with a given metric.

2.2.1 Content evaluation and reference-based metrics

The oldest and most natural methods to evaluate the content of a prediction (i.e. its capacity to capture the main information of the source) involves looking at the similarity between this prediction and a reference. Of course, this requires a gold, preferably human-written, reference. Labeled datasets are often hard to produce, and hence constitute a major constraint, especially in a production setting. However, for a research and development stage, when a labeled dataset is available, comparing a prediction to a reference makes it possible to determine which relevant elements of the source actually appear in the prediction. Similarity between a prediction and a reference can be expressed in several ways, some of which are presented below.

ROUGE: Overlap of tokens

The historical set of metrics used to evaluate text summarization models is ROUGE [7]. The simplest form of ROUGE is the ROUGE-1 metric, which measures the rate of tokens (also called unigrams) that are present in both the prediction and the reference. A precision and a recall can thus be calculated, depending on whether we want to take the point of view of the prediction or the point of view of the reference. More generally, ROUGE-n measures the number of n-grams (i.e. sequences of n tokens) that are present in both the prediction and the reference. Formally, if $\mathbf{P}^n$ designates the set of n-grams of the prediction, and $\mathbf{R}^n$ designates the set of n-grams of the reference, then the precision and recall of ROUGE-n are calculated according to Equation (1).

$$\mathrm{P}_{\text{ROUGE-}n}(\mathbf{P},\mathbf{R}) = \frac{\sum_{w\in\mathbf{P}^n} 1\left[w \in \mathbf{R}^n\right]}{|\mathbf{P}^n|} \quad \text{and} \quad \mathrm{R}_{\text{ROUGE-}n}(\mathbf{P},\mathbf{R}) = \frac{\sum_{w\in\mathbf{R}^n} 1\left[w \in \mathbf{P}^n\right]}{|\mathbf{R}^n|} \tag{1}$$

where $1[x \in \mathbf{X}]$ is 1 if x is in $\mathbf{X}$, and 0 otherwise. It is then possible to deduce a third measure, the F1-score, which, for any measure of precision and recall, is the geometric mean of the latter, as expressed in Equation (2). Optimizing the F1-score allows one to optimize both precision and recall, while avoiding optimizing one to the detriment of the other, which the arithmetic mean cannot prevent.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

A variation of this token overlap metric takes the order of the tokens into account. Indeed, ROUGE-L (see Equation (3)) measures the Longest Common Subsequence (LCS) between the reference and the prediction: it does not require the tokens of this subsequence to appear in a row, but simply in the correct relative order.

$$\mathrm{P}_{\text{ROUGE-L}}(\mathbf{P},\mathbf{R}) = \frac{\mathrm{LCS}(\mathbf{P},\mathbf{R})}{|\mathbf{P}|} \quad \text{and} \quad \mathrm{R}_{\text{ROUGE-L}}(\mathbf{P},\mathbf{R}) = \frac{\mathrm{LCS}(\mathbf{P},\mathbf{R})}{|\mathbf{R}|} \tag{3}$$

In practice, ROUGE-L (LCS), ROUGE-1 (unigram overlap) and ROUGE-2 (bigram overlap) metrics are the most used, because ROUGE-n metrics with $n > 2$ become too rigid to be relevant.

chrF and chrF++: Overlap of characters

The chrF and chrF++ [8] metrics are variants of the ROUGE metrics, whose principle is quite similar. Instead of looking at token overlaps, the idea here is to consider character overlaps, and specifically overlap of sequences of 6 characters (i.e. character 6-grams). The underlying idea is to have a little more flexibility to identify similar but not identical forms, which the ROUGE metrics are not capable of doing. Formally, the precision and recall of the chrF metric are calculated according to Equation (4), where $\mathbf{P}^{\text{6-chr}}$ represents the set of character 6-grams of the prediction, and $\mathbf{R}^{\text{6-chr}}$ represents the set of character 6-grams of the reference. The chrF++ metric, for its part, also takes the overlap of tokens bigrams into account, in the same way as ROUGE-2. According to its authors [8], the addition of this component increases the correlation with human judgment, making it slightly more relevant.

$$\mathrm{P}_{\text{chrF}}(\mathbf{P},\mathbf{R}) = \frac{\sum_{w \in \mathbf{P}^{\text{6-chr}}} 1\left[w \in \mathbf{R}^{\text{6-chr}}\right]}{\left|\mathbf{P}^{\text{6-chr}}\right|} \quad \text{and} \quad \mathrm{R}_{\text{chrF}}(\mathbf{P},\mathbf{R}) = \frac{\sum_{w \in \mathbf{R}^{\text{6-chr}}} 1\left[w \in \mathbf{P}^{\text{6-chr}}\right]}{\left|\mathbf{R}^{\text{6-chr}}\right|} \tag{4}$$

BERTScore: Semantic similarity

The first and main fault of overlap metrics like ROUGE or chrF is their extreme rigidity: they absolutely do not make it possible to manage cases of synonymy or paraphrase, nor to take the context of words into account. For example, from the point of view of the ROUGE or chrF metrics, the sentence "*The lady had her handbag stolen*" will be closer to "*The lady stole her handbag*" than "*The young woman had her purse robbed*", which is obviously a significant misinterpretation. Semantic similarity metrics, which include the BERTScore and the MoverScore presented below, attempt to solve this inconsistency.

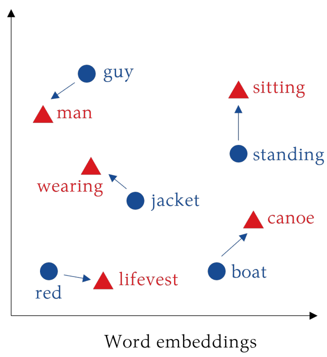
The BERTScore [9] is a metric based on contextual word embeddings provided by the BERT model [10]: it takes the overall context of the sentence into account, as well as the intrinsic meaning of words rather than their surface form. The semantic comparison of two texts is therefore done via the embeddings of the words of each text. The algorithm compares each token of $\mathbf{P}$ (whose embeddings are denoted by $\mathbf{P}_i$) to each token of $\mathbf{R}$ (whose embeddings are denoted by $\mathbf{R}_i$) to compute the precision, and each token of $\mathbf{R}$ to each token of $\mathbf{P}$ to compute the recall. The tokens of one are then greedily associated with the closest token of the other. Precisely, the calculation of the score is done according to Equation (5). An F1-score can then classically be calculated according to Equation (2).

$$\mathrm{P}_{\mathrm{BERTScore}}(\mathbf{P},\mathbf{R}) = \frac{1}{|\mathbf{P}|}\sum_{\mathbf{P}_i\in\mathbf{P}}\max_{\mathbf{R}_j\in\mathbf{R}}\mathbf{P}_i^{\mathrm{T}}\cdot\mathbf{R}_j \quad \text{and} \quad \mathrm{R}_{\mathrm{BERTScore}}(\mathbf{P},\mathbf{R}) = \frac{1}{|\mathbf{R}|}\sum_{\mathbf{R}_i\in\mathbf{R}}\max_{\mathbf{P}_j\in\mathbf{P}}\mathbf{R}_i^{\mathrm{T}}\cdot\mathbf{P}_j \tag{5}$$

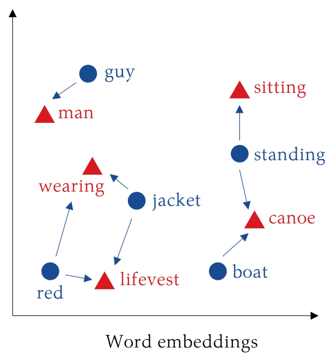
MoverScore: Semantic similarity

The MoverScore [11] is an improvement of the BERTScore, which combines contextual embeddings and the Word Mover Distance (WMD). The authors' goal was double: measure the semantic content shared between the prediction and the reference (which is also the goal of the BERTScore), and measure how the prediction has deviated from the reference. The WMD measures this semantic distance, by associating similar words of the prediction and the reference together (based on their semantic meaning, expressed via their contextual embeddings), and computing the distribution deviation needed to travel from the prediction to the reference.

Concretely, the BERTScore maps one word from the prediction to the semantically closest one in the reference. In contrast, the MoverScore maps one word from the prediction to several semantically related words in the reference (as shown in Figure 3). The general goal of the MoverScore is therefore slightly more ambitious: while the BERTScore measures the content matching between the prediction and the reference, the MoverScore finds the minimum effort to transform the prediction into the reference.



(a) BERTScore



(b) MoverScore

Figure 3: Difference between the BERTScore and the MoverScore – Adapted from [11]. The candidate sentence is here "*A guy with a red jacket is standing on a boat*" (in blue), and the reference sentence is "*A man wearing a lifevest is sitting in a canoe*" (in red).

2.2.2 Factuality evaluation and source-based metrics

While overlap and semantic similarity metrics are quite efficient at capturing the content of a summary, they have two main drawbacks:

1. They are based on a comparison with a gold reference. This is relevant in a research and development context, but not in a production setup, hence the need of source-base metrics, in addition to reference-based metrics.

2. They lack efficiency when it comes to evaluating factuality. Two sentences can have opposite meanings, while using the same words: overlap metrics are not capable of detecting this. Two sentences can differ on a very important point (e.g. the age of a victim for our use-case), and still be extremely close semantically speaking: semantic similarity metrics cannot measure this.

To address both these issues, source-based factuality metrics have been studied and developed, with several different paradigms, some of which are presented below.

BARTScore: Text generation

The BARTScore [12] is a source-based metric, based on the BART model [13], which aims at evaluating text generation tasks within a text generation paradigm, therefore fully using the potential of LLMs for evaluation. The BARTScore is based on the fact that a prediction, possibly generated by an LLM, has a higher content quality and a greater factuality, if the probability of generating this text with BART is high. Hence, it can be seen as a sort of double-check, when the prediction has been generated with another LLM. Computing the BARTScore therefore involves computing the probability (in fact, the log-probability) of generating the prediction with BART. Concretely, if the source is the sequence of tokens $\mathbf{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_n\}$, and if the prediction is the sequence of tokens $\mathbf{P} = \{\mathbf{p}_1, \cdots, \mathbf{p}_m\}$, then the probability of generating the prediction $\mathbf{P}$ given the source $\mathbf{S}$, and with a Seq2Seq language model (such as BART, see Section 2.4.3) parameterized by a set of parameters Θ is given by Equation (6).

$$\mathbb{P}(\mathbf{P}|\mathbf{S},\Theta) = \prod_{t=1}^{m} \mathbb{P}(\mathbf{p}_t|\mathbf{p}_{<t},\mathbf{S},\Theta) \tag{6}$$

The BARTScore is then simply the associated log-probability, with BART as underlying language model.

$$\text{BARTScore}(\mathbf{P},\mathbf{S}) = \log \mathbb{P}(\mathbf{P}|\mathbf{S},\Theta_{\text{BART}}) = \sum_{t=1}^{m} \log \mathbb{P}(\mathbf{p}_t|\mathbf{p}_{<t},\mathbf{S},\Theta_{\text{BART}}) \tag{7}$$

Theoretically, this principle can be adapted to any language model. However, BART is particularly suitable for this task, because it is quite small (and therefore fast) and open-source (and the BARTScore requires having access to the weights of the model to compute the probabilities), and because an implementation of the BARTScore has been released with BART as underlying model.

QuestEval: Question Answering

A new evaluation paradigm [14, 15], based on Question Answering, has emerged recently, based on the following observations:

1. A relevant summary should allow the reader to answer the most important questions that arise from the source;

2. A factual summary should only arise questions that can be answered by the source.

QuestEval [15] is a framework that translates these two elements into a unified metric. It is naturally composed of two blocks, one of which can be considered as a precision measure and the other as a recall measure:

1. **Precision**. Generate, from the prediction, a set of questions. Then generate answers to these questions with the source as context: this provides a precision measure, i.e. a measure of the model's ability to include only elements present in the source. More precisely, the precision is calculated according to Equation (8).

$$\mathrm{P}_{\mathrm{QuestEval}}(\mathbf{P},\mathbf{S}) = \frac{1}{|\mathrm{Q}_{\mathrm{G}}(\mathbf{P})|} \sum_{(q,a)\in \mathrm{Q}_{\mathrm{G}}(\mathbf{P})} \mathrm{F}(\mathrm{Q}_{\mathrm{A}}(\mathbf{S},q),a) \tag{8}$$

   where $\mathrm{Q}_{\mathrm{G}}(\mathbf{P})$ is the set of questions generated from the prediction, $\mathrm{Q}_{\mathrm{A}}(\mathbf{S},q)$ is the answer to the question q generated with the source as context, and $\mathrm{F}(a',a)$ is the F1-score of the similarity between the generated answer a' and the expected answer a (an overlap metric is originally used as similarity metric).

2. **Recall**. Generate, from the source, a set of questions, weighted by relevance. Then generate answers to these questions with the prediction as context: this provides a recall measure, i.e. a measure of the model's ability to restore the main information from the source. The recall is calculated according to Equation (9).

$$\mathrm{R}_{\mathrm{QuestEval}}(\mathbf{P},\mathbf{S}) = \frac{1}{\sum_{(q,a)\in \mathrm{Q}_{\mathrm{G}}(\mathbf{S})} \mathrm{W}(q,\mathbf{S})} \sum_{(q,a)\in \mathrm{Q}_{\mathrm{G}}(\mathbf{S})} \mathrm{W}(q,\mathbf{S})\mathrm{F}(\mathrm{Q}_{\mathrm{A}}(\mathbf{P},q),a) \tag{9}$$

   where $\mathrm{Q}_{\mathrm{G}}(\mathbf{S})$ is the set of questions generated from the source, $\mathrm{Q}_{\mathrm{A}}(\mathbf{P},q)$ is the answer to the question q generated with the prediction as context, $\mathrm{F}(a',a)$ is the F1-score of the similarity between the generated answer a' and the expected answer a, and $\mathrm{W}(q,\mathbf{S})$ is the weight of the question q generated with the source.

The precision and recall measures are then combined as an F1-score to provide the overall score. Figure 4 illustrates how QuestEval works.

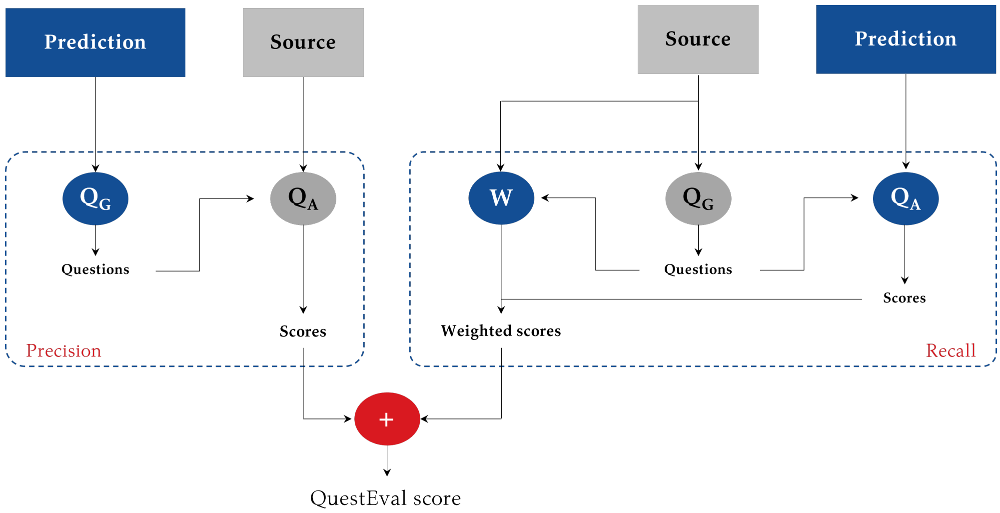


Figure 4: QuestEval metric – Adapted from [15]. $\mathbf{Q_G}$ designates a Q&G model, $\mathbf{Q_A}$ designates a Q&A model, and **W** a weighter.

The implementation proposed by the authors of QuestEval [15] however suffers from several drawbacks for our specific use case, which is why we propose our own implementation of this metric, described in more detail in Section 3.3.

SelfCheckGPT: LLM-based

SelfCheckGPT [16] is a set of metrics that make use of the capacities of LLMs themselves to evaluate their factuality. It is based on several interesting ideas. The first one is that, even though an LLM is capable of producing a factually false output, it is also capable of spotting factually false content when explicitly asked to do so, and is therefore capable of correcting itself. The second main idea is that an LLM is less likely to hallucinate or invent elements several times in a row: generating several outputs from a model allows to double or triple-check information. Besides, one main advantage of these metrics is their use of LLMs as a black box: they do not require having access to the model's architecture as the BARTScore, they only require generating text with it.

SelfCheckGPT-Prompt (see Figure 5) works as follows:

1. Generate several samples (i.e. other summaries) from the source with an LLM;

2. For each sentence of the prediction, explicitly ask the LLM if the sentence is supported by each sample;

3. Average over the samples, and over the sentences of the prediction.

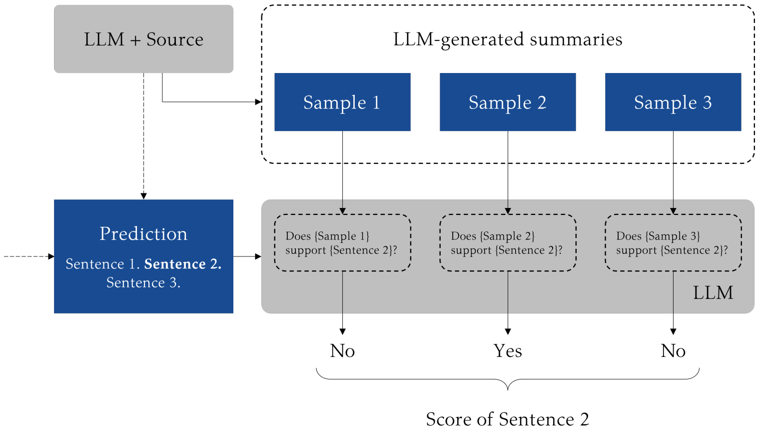


Figure 5: SelfCheckGPT-Prompt metric – Adapted from [16]. The final score of the prediction is given by averaging over the sentences.

More precisely, let us write $\mathrm{N_s}$ the number of samples, $\mathbf{P}_i$ the i-th sentence of the prediction, and $\mathbf{S}^n$ the n-th sample. Let us also consider the function $\mathrm{LLM}(\mathbf{x}, \mathbf{y})$, which is 1 if the LLM answered that $\mathbf{y}$ supported $\mathbf{x}$, and 0 otherwise. Then, the SelfCheckGPT-Prompt Score, at the sentence level, is given by Equation (10).

$$\mathrm{SelfCheckGPT_{Prompt}}(i) = \frac{1}{\mathrm{N_s}} \sum_{n=1}^{\mathrm{N_s}} \mathrm{LLM}(\mathbf{P}_i, \mathbf{S}^n) \tag{10}$$

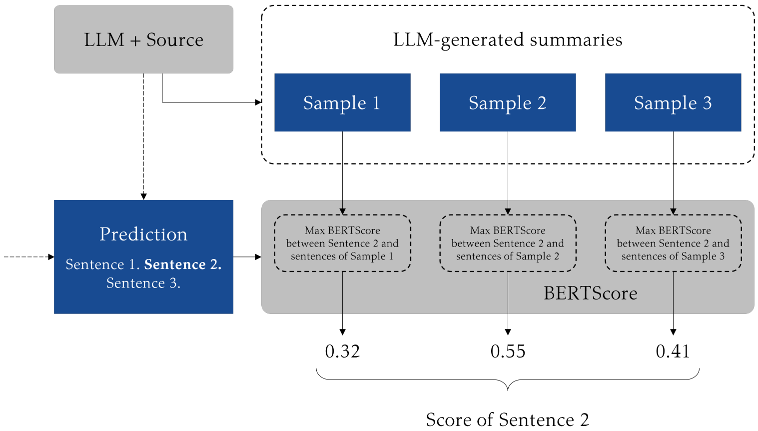


Figure 6: SelfCheckGPT-BERT metric – Adapted from [16]. The final score of the prediction is given by averaging over the sentences.

As for SelfCheckGPT-BERT (see Figure 6), it works as follows:

1. Generate several samples (i.e. other summaries) from the source with an LLM;

2. For each sentence of the prediction, and for each sample, compute the maximum BERTScore over the sample's sentences;

3. Average over the samples, and over the sentences of the prediction.

The computation of the SelfCheckGPT-BERT Score is done, at the sentence level, according to Equation (11), with the same notations as above.

$$\mathrm{SelfCheckGPT}_{\mathrm{BERT}}(i) = \frac{1}{\mathrm{N_s}} \sum_{n=1}^{\mathrm{N_s}} \max_k \left[\mathrm{F_{BERTScore}} \left(\mathbf{P}_i, \mathbf{S}_k^n \right) \right] \tag{11}$$

The score at the sentence level is then averaged over the sentences of the prediction, for both the Prompt and the BERT versions, as shown in Equation (12).

$$\mathrm{SelfCheckGPT}_{\mathrm{Prompt/BERT}}(\mathbf{P}) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathrm{SelfCheckGPT}_{\mathrm{Prompt/BERT}}(i) \tag{12}$$

2.3 Extractive summarization

As explained in Section 2.1.1, extractive summarization has historically been widely used, due to its relative simplicity, and lower technical requirements. The methods presented in the following paragraphs, for instance, date back to the early 2000s.

2.3.1 TextRank

The TextRank [17] algorithm is an adaptation of Google's PageRank [18], which has historically been used to rank web pages. The principle of PageRank is to consider all web pages as a directed graph, whose nodes are the pages, and whose edges are the links between these pages. A web page will therefore have more edges if it is referenced by many other pages. The algorithm then assigns a score to each page, based on its edges and the scores of its neighbors, according to Equation (13), where $\mathbf{u}$ is a web page, $\mathrm{B}_\mathbf{u}$ is the set that contains all pages linking to $\mathbf{u}$, and $\mathrm{L}(\mathbf{v})$ is the number of links from page $\mathbf{v}$.

$$\mathrm{PageRank}(\mathbf{u}) = \sum_{\mathbf{v} \in \mathrm{B}_\mathbf{u}} \frac{\mathrm{PageRank}(\mathbf{v})}{\mathrm{L}(\mathbf{v})} \tag{13}$$

The PageRank therefore favors the central pages, defined by two criteria: they must be referenced by numerous pages, and the pages which reference them must themselves be central.

The idea behind TextRank is to adapt this algorithm to the text summarization framework, with the following steps, also represented on Figure 7:

1. Pre-process the sentences of the source (punctuation and stopwords removal, tokenization, lemmatization, etc.);

2. Represent sentences as vectors: this representation can be first performed at the word level via word embeddings (e.g. Word2Vec [19] or FastText [20]), then at the sentence level (by averaging the embeddings of the words that make up the sentence);

3. Compute a similarity matrix (by calculating the cosine similarity, defined in Equation (14), of each pair of sentences);

$$\mathrm{Sim}_{\mathrm{cosine}}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^{\mathrm{T}} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||} \tag{14}$$

4. Deduce from the similarity matrix a graph representation of the source, the nodes of which represent the sentences of the text, and the edges of which represent the similarity between these sentences;

5. Apply the PageRank algorithm on the resulting graph, which yields scores for each sentence, as described by Equation (13).

6. Extract the k sentences with the highest scores.

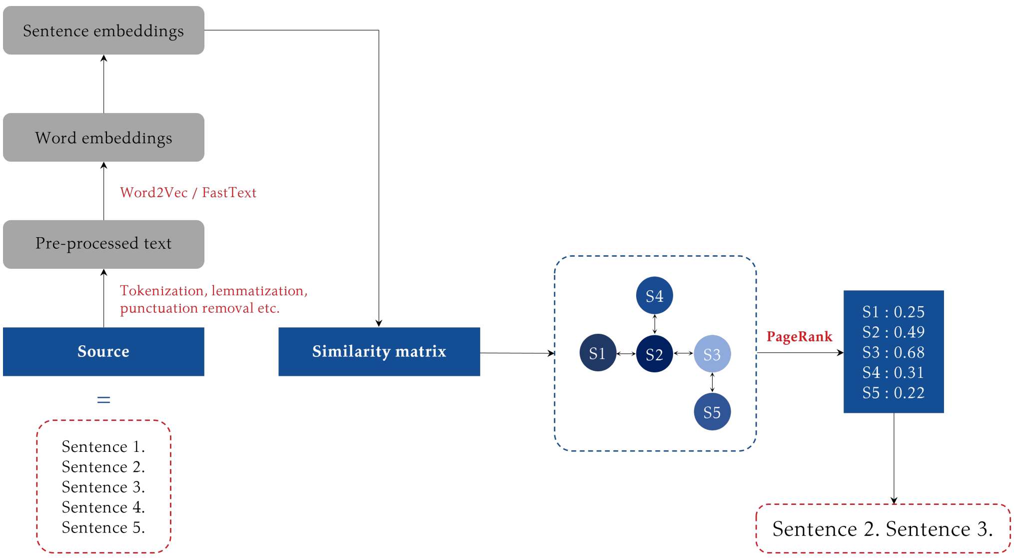


Figure 7: The TextRank algorithm

The TextRank algorithm has many advantages, including its speed of execution, and the fact that it allows you to choose the number of sentences desired *a posteriori* (which can be very useful when used as pre-processing step, see section 3.5). However, it also has the main disadvantage of favoring central but semantically close sentences. Considering an extreme case, this will place two central yet identical sentences very high in the ranking, which is obviously problematic: we do not expect a good summary to be redundant. One way to solve this problem is to use a similarity threshold when extracting sentences (see Section 3.4 for a detailed explanation). Another solution can be the use of a clustering algorithm, as presented in the following paragraph.

2.3.2 K-Means

To preserve the semantic diversity of the source, a classical approach is to use a clustering algorithm to group together sentences with close meanings. This can be done by first pre-processing the source and embedding its sentences (in the same way as for the TextRank), and then applying an algorithm such as the K-Means on these vectorial representations, to create as many clusters as sentences desired in the summary. The underlying idea is that each cluster groups together semantically close sentences, and therefore contains one part of the information from the text. For each cluster, it is then necessary to determine its representative, i.e. the sentence closest to the centroid of the cluster. All that remains is then to sort and concatenate the selected sentences to form the summary. The principle of this approach is schematized on Figure 8.

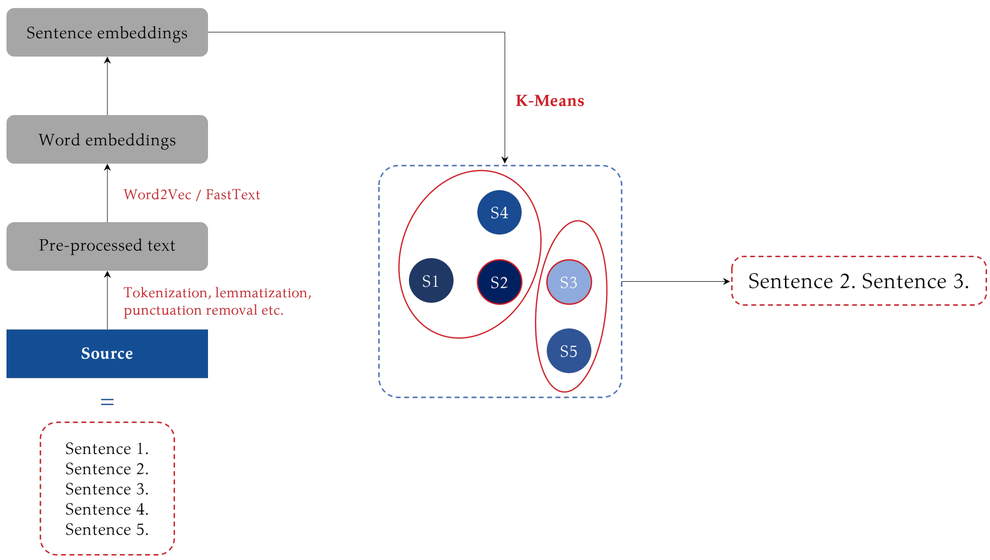


Figure 8: The K-Means algorithm, applied to extractive text summarization

The main disadvantage of this algorithm is that the number of sentences wanted in the final summary must be specified in advance, because the K-Means algorithm requires knowing beforehand the number of clusters to compute. If one wishes to reduce the size of a text up to a certain limit of words or tokens (see Section 3.5), without knowing the number of sentences required for this limit, it is therefore not possible to use it.

2.4 Abstractive summarization

Figure 9 is a simplified timeline of the main improvements in terms of architectures and models, that are or were used in NLP for abstractive text summarization. The most important elements of this timeline will be presented in the following paragraphs.

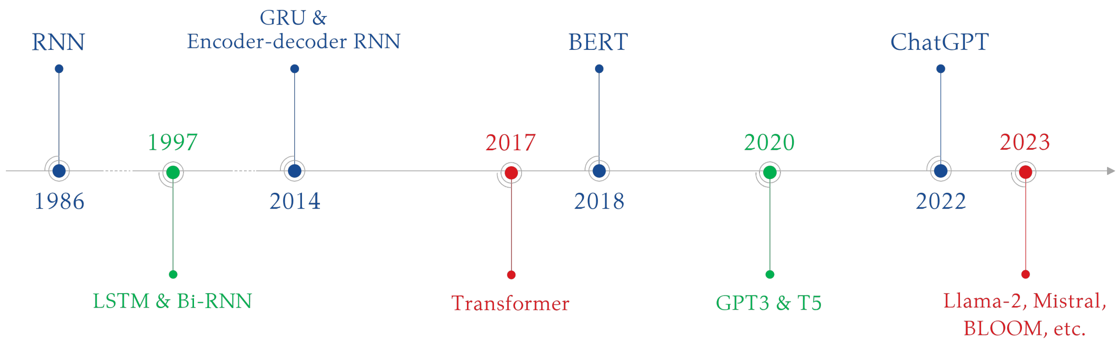
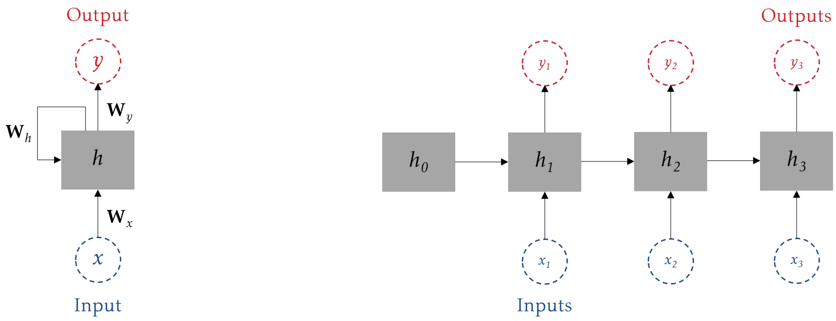


Figure 9: Historical timeline of various abstractive summarization models

2.4.1 Recurrent neural networks and variations

Historically, the first significant advances in Natural Language Processing came from recurrent neural networks (RNN) and their ability to model the temporality of language. Unlike feed-forward neural networks (FNN), RNNs use recurrent connections (see Figure 10), which allows them to keep past information in memory.



(a) Folded representation

(b) Unfolded representation

Figure 10: A Recurrent Neural Network – A part of the hidden layer's input is a recurrent connection: the hidden layer's input is made of the network's current input, as well as the hidden layer's output from the previous time step. On the unfolded representation, each arrow represents a linear transformation, whose weights are shared across different time steps.

Several improvements to this basic architecture have been proposed:

1. **At the level of hidden layer cells**: LSTM [21] and GRU [22] cells;
2. **At the global architecture level**: stacked and bi-directional RNNs, encoders-decoders.

However, all these architectures suffer from the problem of exploding and vanishing gradients, which prevents them from efficiently capturing distant dependencies. In practice, for text generation, such models are not capable of handling contexts of more than a few words or sentences. The Transformer architecture, briefly presented in the following paragraph, offers a revolutionary solution to this problem, based on an attention mechanism.

2.4.2 The Transformer architecture

Introduced in 2017, the Transformer architecture [23] has revolutionized the field of NLP. Compared to RNNs, Transformers allow efficient parallel processing of sequences, leading to a much shorter computation time (both for training and inference), while RNNs, by construction, can only process a sequence sequentially, that is, token by token. Furthermore, the self-attention mechanism, presented below, can effectively capture distant dependencies by mitigating the problem of vanishing and exploding gradients.

Self-attention is the core mechanism of Transformers. It is used to weight, when examining a particular token, the importance, relative to that token, of each other token in the sequence. Concretely, three vectors (which each represent the input sequence in a different role) are deduced from the input sequence $\mathbf{X}$: the queries ($\mathbf{Q}$), the keys ($\mathbf{K}$) and the values ($\mathbf{V}$), by linear transformations as expressed in Equation (15). The weights of the matrices $\mathrm{W}_\mathbf{Q}$, $\mathrm{W}_\mathbf{K}$ and $\mathrm{W}_\mathbf{V}$ are trainable parameters of the model.

$$\mathbf{Q} = \mathbf{X} \cdot \mathrm{W}_\mathbf{Q} \quad , \quad \mathbf{K} = \mathbf{X} \cdot \mathrm{W}_\mathbf{K} \quad \text{and} \quad \mathbf{V} = \mathbf{X} \cdot \mathrm{W}_\mathbf{V} \tag{15}$$

Attention scores are then calculated according to Equation (16): for each input token $\mathbf{X}_i$, the result $\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})_i$ is a combination of all other elements in the sequence, weighted according to their relevance to $\mathbf{X}_i$.

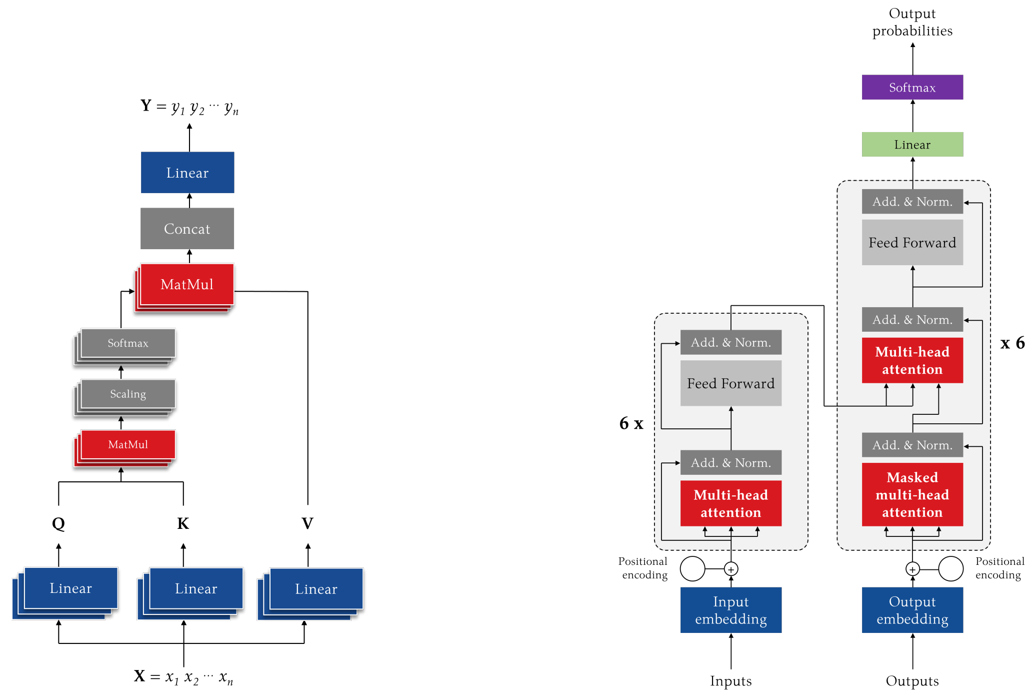
$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathrm{T}}{\sqrt{d}}\right)\mathbf{V} \tag{16}$$

Self-attention as presented above is however not directly used in the Transformer architecture. Instead, an extension, called multi-head attention, allows the model to capture multiple aspects of the relationships and dependencies between elements in the input sequence. This is done by transforming the input sequence into several heads, i.e. into several vectors of queries, keys and values, and by applying a self-attention mechanism on each of these heads. The attention vectors of each head are then concatenated and linearly reduced to the original input size. The calculation of multi-head attention is detailed in Figure 11a, and in Equation (17).

$$\mathrm{MultiHeadAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_h)\mathbf{W}^\mathrm{O} \tag{17}$$

where $\mathrm{head}_i = \mathrm{Attention}(\mathbf{X} \cdot \mathrm{W}^i_\mathbf{Q}, \mathbf{X} \cdot \mathrm{W}^i_\mathbf{K}, \mathbf{X} \cdot \mathrm{W}^i_\mathbf{V})$ for $i = 1, \cdots, h$ with h the number of attention heads. Each attention head can therefore specialize in a specific aspect of the data, and the model can learn to combine these different aspects for a better representation.

The combination of this multi-head attention mechanism with normalization layers and feed-forward layers (FNN) forms a Transformer block. Several blocks (6 in the original implementation) then form the encoder (which has access to the input sequence in its entirety) and the decoder (which has access to the encoded representation of the input sequence, and to the output sequence generated so far). The combination of these two elements make up the original Transformer encoder-decoder, as shown in Figure 11b.



(a) Multi-head attention mechanism

(b) Global architecture

Figure 11: The Transformer architecture – Adapted from [23]

2.4.3 Large Language Models

Large Language Models (LLMs) are deep networks based on a part of (encoder or decoder), or on the whole Transformer architecture. Their specificity is their very high number of parameters (up to several hundred billions), that allow them to capture deep, distant and intrinsic dependencies of textual data. The LLMs developed since 2018 are divided into three categories (see Figure 12):

1. **Decoder-only or auto-regressive models.** These models only use the decoder block from the Transformer architecture. Their pre-training is based on the prediction of the next token: at each step, the model learns to predict a given token, by having access to all the tokens located before the token in question, which is why these models are often called *auto-regressive*. These models are particularly useful and efficient for text generation tasks. Among the best-known decoder-only models, we can cite GPT [24] (on which ChatGPT is based, developed by OpenAI), Llama [25] and Llama-2 [26] (developed by Meta), Mistral [27] and Mixtral [28] (developed by Mistral AI).

2. **Encoder-only or auto-encoders models.** Unlike decoder models, these only use the encoder block of the Transformer architecture. Their pre-training is often based on the reconstruction of sentences: at each step, the model has access to the entire sentence (one sometimes speaks of bidirectional attention), except certain words which have been masked, and learns to find these hidden words. These models are best suited for tasks that require a detailed understanding of an entire text, such as text classification, named entity recognition, or Question Answering. The best-known encoder-only models are BERT [10] (developed by Google) and its derivatives or adaptations (RoBERTa, CamemBERT, etc.).

3. **Encoder-decoder or Seq2Seq models.** These models use the two blocks initially present in the Transformer architecture. The encoder thus has access to the entire input sequence, while the decoder only has access to its hidden representation, and to the output tokens generated so far. These models can be trained in the manner of a decoder model or an encoder model, or in a more complex manner (reconstruction of several successive hidden tokens for example). Encoder-decoder models are effective for tasks that require text generation, given an input text, such as translation or generative question answering. Among the most popular encoder-decoder models are BART [13] (developed by Meta, then called Facebook) and T5 [29] (developed by Google).

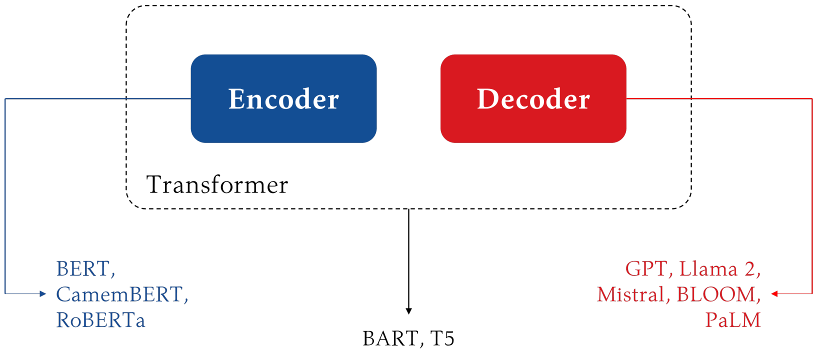


Figure 12: LLMs classification

The development of an LLM involves three main steps, schematized on Figure 13:

1. The **pre-training**. This task depends on the model's architecture and pre-training objectives (see previous paragraph), and is highly dependent on the training dataset and the training duration. The higher the number of parameters, the larger the training dataset and the longer the training should be. To give an order of magnitude, Llama-2 (all versions combined) was trained on 2 trillion tokens for an equivalent of 3.3 million GPU hours [26]. This pre-training task is a form of self-supervised learning: the labels used for training are computed from the data itself (e.g. the next token given a sequence of tokens), instead of being computed by a person.
2. The **instruction-tuning** (optional). During the pre-training step, models such as Llama-2 or GPT learn to predict the next token, based on a sequence of tokens. This, however, is not relevant for various tasks, including text summarization. To be used in a multi-task environment, most open-source and proprietary models are available in an instruction-tuned version (also called *Chat* or *Instruct* version). Such models, after the pre-training step, have further been trained, in a supervised way, to generate an expected output, based on an instruction and a possible input. Common instruction categories include sentiment analysis, code generation, arithmetic reasoning, text summarization, etc. The instruction-tuning step can be improved using Reinforcement Learning from Human Feedback (RLHF) [30]: this consists of updating the weights of the models by receiving feedback from humans. Among other things, this allows the model to align to human preferences regarding helpfulness and safety.
3. The **fine-tuning** (optional). One of the main advantages of LLMs comes from the fact that they can be even more specialized on a specific task and/or data. This optional, supervised step is described in more details in Section 2.4.6.

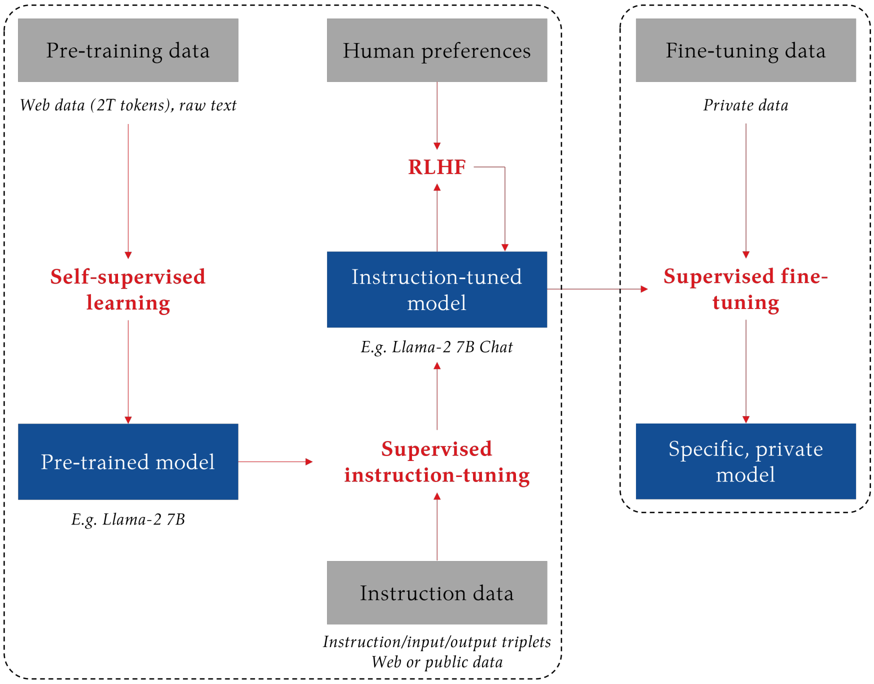


Figure 13: Different steps of the development of an LLM – The first block represents a public LLM such as Llama-2 7B, and the second block represents its specialization (and therefore privatization) on a specific, private data and task (e.g. a summarization model for the ANFSI).

2.4.4 LLM review

As explained in Section 1.4, we have only explored open-source, reasonably sized models, hence excluding proprietary LLMs such as GPT or Bard, and huge open-source models such as BLOOM. Table 1 gives an overview of the size of well-known models. The following paragraphs present Llama-2, Mistral and Mixtral, which are the three models we have used.

Model name	# Parameters
BERT	345M
Llama-2 (small)	7B
Mistral	7B
T5	11B
Mixtral	47B
Llama-2 (big)	70B
GPT-3	175B
BLOOM	176B
PaLM	540B

Table 1: Number of parameters of well-known LLMs

Llama-2 (7B, 13B, 70B)

Llama-2 [26] is an LLM created and distributed in open-source by Meta in July 2023. It is a "small" LLM, in the sense that, depending on the version, it has only 7, 13 or 70 billion parameters (see Table 1). The strength of Llama-2, according to its creators [26], lies in its small size and therefore its lower operating cost, which nevertheless allows it to compete with much larger models like GPT-3 [26]. The authors have also released information concerning the training dataset: notably, the model was overwhelmingly trained on English data (almost 90% of the data) compared to French data (only 0.1% of the data), which may explain its lower performance on French-related tasks. Besides, the training dataset was composed of a massive set of 2 trillion tokens. The raw model was then instruction-tuned in a supervised manner, using a reinforcement learning algorithm based on human feedback (RLHF) as well, on high-quality data, to provide a *chat* version. Finally, the context length of 4096 tokens (around 3000 words, including the input text and the summarization instruction) makes it possible to handle medium sized texts, and the use of Grouped-Query Attention (GQA) provides a good trade-off between performance and computation time.

Mistral (7B)

Mistral [27] is a small LLM created and distributed in open-source by a very recent French start-up, Mistral AI, in September 2023. It has only 7 billion parameters (the equivalent of the smallest Llama-2 model, Llama-2 7B), yet it outperforms the medium sized Llama-2 13B model. The architecture and technical specificities are very similar to those of Llama-2. One main difference probably lies in their training data, about which no information has however been released, but which should contain more multilingual data, and possibly more qualitative data.

Mixtral (8x7B)

In December 2023, Mistral AI has released a more ambitious and innovative model: Mixtral 8x7B [28]. It is based on a Mixture-of-Experts (MoE) architecture. Concretely, each FNN layer in the LLM is replaced by a MoE layer, which consists of the two following elements:

1. A given number of **experts** (8 in the case of Mixtral 8x7B). Experts could be any type of neural network; they are here FNNs.

2. A **router**. It decides which token must be fed to which expert. The output of the model, for each input token, is therefore computed by one expert, using only a part of the model.

MoE architecture allows a very efficient pre-training, and a quick inference. Indeed, only a subset of the parameters of the whole model are used during inference (each token being sent only to one expert). The inference time of Mixtral 8x7B is therefore comparable to that of Llama-2 13B, while being a much bigger and more efficient model. However, the whole model must be loaded in VRAM during inference: the memory requirements are the same than for a dense model with the same number of parameters.

The idea behind MoE, which is quite similar to that of ensemble methods, is that each expert network specializes in a specific task, data or textual representation. After the training of each expert and of the router, the overall performances should be improved, compared to a single, general network. Although this was never confirmed, some rumors claim that GPT-4, on which the latest version of ChatGPT is based, is an MoE. This would explain a hypothetically very high number of parameters with a yet very fast inference and impressive performances.

2.4.5 Inference with pre-trained models

Weight quantization

One of the main problems when using LLMs with a very high number of parameters (from a few billion to a few hundred billion) comes from the memory requirements. Indeed, for quick inference, it is essential to use a GPU to accelerate the calculations, but these calculations require loading the entire model in RAM, or, when using a GPU, in VRAM. As each weight of a model is initially stored as a 16-bit floating point (`float16`), i.e. with a precision of 16 bits (2 bytes), a model with 7 billion parameters (like Llama-2 7B) requires 14 GB of VRAM (to which must be added the memory necessary for the input of the model). Simply using a Llama-2 70B model in inference thus requires 140 GB of VRAM, which is more than the VRAM of any current GPU (this limit can be exceeded by using multiple GPUs with 80 GB of VRAM in parallel).

A solution to reduce the memory requirements is the quantization of a model's weights, using for instance the GPTQ algorithm [31]. Quantization involves converting the 16-bit floating point representation of weights into a lower-precision representation, such as 8-bit integers or 4-bit integers. By using fewer bits to represent each weight, the model's memory requirements can be significantly reduced, leading to more efficient storage and faster computation during both training and inference. However, this gain in memory efficiency comes with a precision reduction, whose impact on the model's performances can be significant.

Prompt engineering

Prompt engineering is a rather empirical technique, specific to LLMs, and especially instruction-tuned LLMs (sometimes called *Chat* or *Instruct* versions of raw LLMs). When interacting with or requesting such models, the user formulates his or her request, in a way that will have an impact on the model's response. For example, asking a model "*Can you please summarize this text?*" or "*Write a condensed version of this text*" will very likely yield different results. The user's request (e.g. a summarization instruction), along with a possible input (e.g. a text to be summarized) is called a *prompt*. Prompt engineering (or prompt design) is the process during which this prompt is created, improved or optimized, so that the model provides the best possible response, given the user's objectives.

Given the very low intrinsic interpretability of LLMs, prompt engineering is mainly empirical. Some common tips when working on a prompt include being very specific on the task and data, add a detailed context and precisely express the expected output. It is however hard to measure the impact of the use of different yet close words in the prompt on the model's outputs, because such choices have very unpredictable and often not generalizable consequences.

A common technique in prompt engineering is few-shot prompting: it consists of giving examples to the model, in the prompt, of close inputs and outputs (be it in terms of task, data, style etc.), so that the model has a clearer view of what is expected. A variation of few-shot prompting using Retrieval Augmented Generation, and applied on our specific summarization task and data, will be presented in detail in Section 4.3.3.

Some techniques have also been developed to automatize and optimize this prompt engineering task, including PromptBreeder [32]. It is based on a standard category of optimization algorithms: genetic algorithms. It consists of creating a generation of individuals (which are, in this case, prompts), that will evolve through mutation and selection steps (based on randomness, and performances on given metrics), to give birth to a new generation of individuals. After a few generations, the population will have evolved to a more advanced and efficient state. The originality of this approach is the mutation of prompts, that is done with the LLM itself (self-referential approach): one specifically asks the model to make a prompt evolve, given some mutation prompt. Such mutation prompts can themselves evolve during training (self-improvement approach), via hyper-mutation mechanisms. PromptBreeder is a promising and precursor work for the very recent domain of prompt engineering or prompt optimization, but has not yet been widely adopted.

2.4.6 Fine-tuning of pre-trained models

Full fine-tuning

Fine-tuning involves training a pre-trained model, in a supervised manner, on a specific task and/or data. Concretely, in our case, it involves repeating the following steps for a large number of examples:

- Generate, with the current model and from the source, a prediction;
- Compare this prediction to the reference, and deduce a loss;
- Backpropagate this loss through the network to update the model weights.

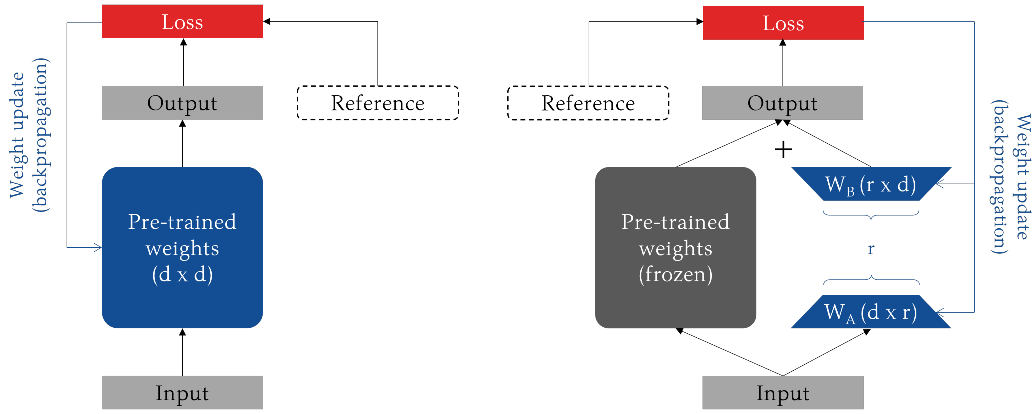
In a basic view of fine-tuning (see Figure 14a), all model weights are updated during this backpropagation step. This is very expensive from several points of view:

1. In **memory**: one must keep in memory all the weights of the model, the vector representations of the current batch, as well as the gradients necessary for backpropagation (i.e. the equivalent of twice the whole model for an optimizer like Adam [33]). For a model with 7 billion parameters (without quantization), this can represent up to 35 GB.

2. In **computational power and time**: with a powerful A100 GPU available in the Datalab, fully fine-tuning a small model on a small dataset (around 1000 samples) and a few epochs would already take several hours.

3. In **data**: each example updates the model's weights with a low learning rate (otherwise the model may overfit the training data), which requires a very large amount of training data to observe a significant improvement.

To overcome these problems and as the development of an LLM (even of moderate size) is currently not within the reach of the vast majority of companies, methods for parameter-efficient fine-tuning have been developed. The following paragraph presents one of the most popular.



(a) Full, basic fine-tuning

(b) LoRA parameter-efficient fine-tuning

Figure 14: Difference between basic and LoRA fine-tuning – Adapted from [34].

Parameter-efficient fine-tuning

A very popular category of parameter-efficient fine-tuning is that of adapters: it consists of adding a set of weights to the model, freezing the weights of the pre-trained model (which will therefore not be updated during the backpropagation step), and only train the additional weights. The selection and determination of these new weights will depend on the chosen method. Adaptation has several significant advantages. Firstly, by training only a small number of parameters, the requirements (computational power and time, memory, data) will be much lower than by training a whole model, without a huge degradation of performance. Secondly, adaptation makes it possible to train several subsets of parameters for specific tasks, and to add them to the pre-trained base model whose parameters have been frozen (and which is therefore the same for all these tasks): several specific models will thus be available, each composed of a large common part, and a small specific part.

The most commonly used adaptation method is called LoRA (Low-Rank Adaptation) [34], the principle of which is schematized in Figure 14b. Instead of considering a square, full-rank weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$, this technique considers a low-rank weight matrix, which can be expressed as the multiplication of two rectangular matrices $\mathbf{W} = \mathbf{W}_A \times \mathbf{W}_B$, with $\mathbf{W}_A \in \mathbb{R}^{d \times r}$, $\mathbf{W}_B \in \mathbb{R}^{r \times d}$ and $r \ll d$. Hence, instead of updating d^2 parameters during backpropagation (and compute the gradients for all these parameters), LoRA fine-tuning updates only $2rd$ parameters. Experiments [34] show that, while impressively reducing the number of parameters to train (1 or 2% of the initial number of parameters for small values of r), this does not come with a significant degradation of performances.

2.5 Related work

2.5.1 Summarization state-of-the-art

Extractive text summarization is a quite old research topic, with still efficient methods such as TextRank [17], a graph-based approach, dating back to the early 2000s. More recently, the development and the increasing popularity of deep neural networks has opened the door to more efficient methods. SummaRuNNer [35] thus uses a recurrent neural network to optimize content, salience and novelty of the extracted summary. Another approach [36] models the extractive summarization task as a reinforcement learning objective, and therefore learns to optimize the ROUGE score of the extracted summary with a reinforcement learning algorithm. These deep models however require an important training to work accurately. To address this issue, training-free methods have also been studied, including the use of genetic algorithms [37].

Since the introduction of the Transformer [23] architecture, abstractive summarization has been made possible and efficient, and has been widely studied. By their architecture, particularly suited for conditional text generation, encoder-decoder models were originally very popular for text summarization. Well-known examples of such efficient models include the denoising encoder-decoder BART [13] and the T5 [29], trained to fill out missing words.

More recently, large decoder-only models, that usually have a much higher number of parameters, have proven more efficient and capable on abstractive text summarization. Among the most popular, we find the unavoidable GPT-3 [24] (175B parameters), and its instruction-tuned version InstructGPT [38]. Even bigger models that perform incredibly well on summarization are BLOOM [4] (176B parameters) and PaLM [39] (560B parameters). On the other side, much smaller and therefore less expensive models have emerged and shown satisfying performances on text summarization. The most famous and popular are Llama-2 [26] (which comes in several sizes: 7B, 13B or 70B parameters), Mistral [27] (7B parameters) and Mixtral [28] (8x7B parameters).

2.5.2 French data summarization

Current state-of-the-art models for text summarization are mainly English or slightly multilingual models. Llama-2, for instance, has been trained on more than 90% English data, and less than 1% French data [26]. Therefore, given the need of language-specific language models and the intrinsic limitations of base multilingual models, various models adapted for French have been proposed. First, French adaptation of the old encoder-only BERT [10] such as CamemBERT [40] and FlauBERT [41] have emerged. By their encoder-only architecture and their small size however, these models are not well suited for text summarization.

BARThez [42] is an encoder-decoder model, similar to the well-known BART, trained specifically on French data. In particular, it is evaluated on a new French summarization dataset, OrangeSum [42], and shows great performances. However, similarly to BART, it has now been outperformed by recent and much larger auto-regressive models. A decoder-only model, of similar size and French-specific, has been proposed recently: CroissantLLM [43] is general-purpose and multi-task, and has been trained on half English and half French data. Its relatively small number of parameters (1.3B) however makes the understanding of complex and long texts difficult, which prevents it from being competitive with English state-of-the-art models.

Since no big enough French model has been proposed, and since multilingual state-of-the-art models are not especially suitable for French language, adaptations of such models have been performed on French data. A current project of the French government is for instance the adaptation of Llama-2 on French language, based on a French parliamentary database, called Albert. Besides, concerning the specific task of summarization, LLaMandement [44] is a fine-tuned version of Llama-2 for the summarization of French legislative proposals.

2.5.3 Non-english data summarization

Bigger and therefore more capable language-specific LLMs have been developed for widely-spoken languages such as Arabic (Jais and Jais-Chat [45]) or Chinese (GLM-130B [46]), making the understanding and summarization of complex texts in these languages quite efficient. Similarly, MediaGPT [47] is a Chinese LLM, trained mostly on media data, and therefore suitable for a certain type of Chinese data. Such work has however, to the best of our knowledge, not been performed on French data.

2.5.4 Judicial data summarization

A lot of research focused on summarization of judicial documents such as court decisions. The language can be more formal than in judicial procedures from the police, but to the best of our knowledge, this particular type of documents has not been studied specifically.

Authors of [48] propose a comparison of extractive models, in Italian, for the summarization of judicial decisions from the Constitutional Court and the Supreme Court in Italy. Similarly, [49] focuses on extractive summarization of US court opinions. Specifically, it introduces a massive dataset of 430K court opinions with annotations for key passages, and releases open-source models. A deep reinforcement learning approach, based on Proximal Policy Optimization algorithm, is presented in [50] for legal extractive summarization.

Taking into account both extractive and abstractive models, [51] compares several different models on English datasets (from the UK and from India) from the Supreme Court, with a focus on long documents. For long legal judgments summarization specifically, [52] proposes a two-step process: a set of sentences and a set of keywords are first selected in an extractive manner, before being given to an abstractive model that can thus focus on essential points.

Concerning abstractive summarization, [53] studies the fine-tuning of a pre-trained Longformer encoder-decoder, on a dataset of the Supreme Court of Pakistan. Finally, LegalSumm [54] is a method to abstractively summarize long legal documents, focusing on hallucination detection. This detection is based on the generation of several independent summaries, and the evaluation of the textual entailment between these.

2.5.5 Summarization evaluation

Concerning factuality metrics and hallucination detection, some methods have been developed and implemented in a English-specific way, such as QuestEval [15] or QAEval [14]. Other metrics are not specific to text summarization and apply to more general text generation tasks, such as SelfCheckGPT [16].

Chapter 3

Methods and experiments

This Chapter presents the main work carried out for this thesis. After briefly stating the resources and data we had access to in Sections 3.1 and 3.2, it presents the methods and experiments conducted concerning evaluation metrics in Section 3.3, extractive summarization in Section 3.4, and abstractive summarization in Section 3.5.

3.1 Resources

The use of certain automatic summarization methods, and in particular that of Large Language Models (LLMs), is very costly in terms of computation power and time, RAM, VRAM, and data (the latter concerning some methods only, see Section 2.4.6). The resources available for this thesis mainly consisted of:

- A database of audition reports, accompanied by their summary (see Section 3.2);
- An Nvidia A100 GPU with 80 GB of VRAM.

3.2 Available data

To carry out this thesis, we have access to audition reports, originally stored as structured PDF files. The structure of these files is represented on Figure 15, and is always the same for all reports, which makes it quite easy to extract the body of the audition, i.e. the only part which we are interested in.

The sensitive and personal nature of the data raises the question of GDPR and the protection of personal data. For this reason, the data we have access to has been pseudonymized, meaning that all personal information has been replaced by fake, non-personal information, using a pseudonymization tool developed within the ANFSI. This prevents the persons concerned from being recognized on the basis of this personal data, and it prevents models trained on this data from leaking sensitive personal data.

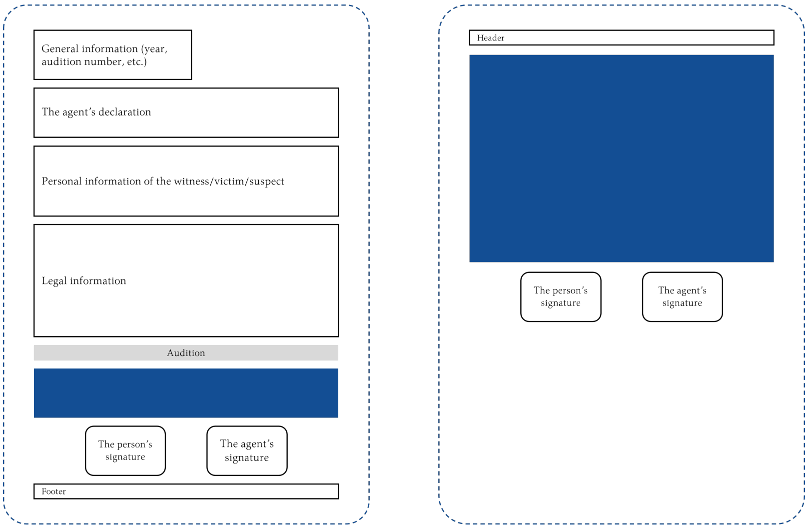
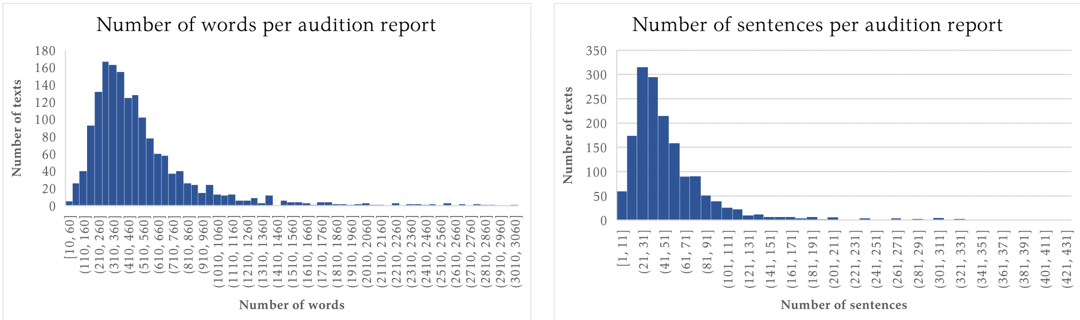


Figure 15: Template of an audition report – The body of the audition, which we are interested in, is the blue part.

In addition to these reports, we have in our possession summaries, written by field agents. This data can be either single-document (i.e. an audition report, and its associated summary), or multi-documents (i.e. a set of audition reports, that refer to the same investigation, and the summary associated with this investigation). For reasons explained in Section 2.1.2, we focused primarily on single-document data, some statistics of which are shown in Table 2 and in Figure 16b. Although there is a variability in the data in terms of length, it must be noted that the source texts are quite long texts, composed of several dozen sentences. The abstractiveness of a summary, explained in more details in Section 3.3.6, measures the extent to which the summary makes use of new words or expressions.

	Train set	**Test set**	**Valid. set**
Text length (words)	530.7	567.93	524.86
Text length (tokens)	971.78	1047.91	979.24
Text length (sentences)	54.03	57.06	53.09
Summary length (words)	64.14	65.5	63.46
Summary length (tokens)	119.77	122.92	116.64
Summary length (sentences)	5.65	6.09	5.41
Summary abstractiveness	0.627	0.618	0.616
Number of report/summary pairs	1632	181	215

Table 2: Data statistics – The abstractiveness has here been defined as the geometric mean of the percentages of novel unigrams, bigrams, trigrams and fourgrams of the summary compared to the text (see Section 3.3.6). A Llama-2 tokenizer was used to compute the number of tokens.



(a) Number of words per audition report

(b) Number of sentences per audition report

Figure 16: Histograms of the length of training source texts

3.3 Evaluation metrics

3.3.1 Global framework

For clarity and synthesis reasons, we choose to create one global metric per evaluation paradigm. When several metrics belong to the same paradigm, the global metric is the geometric mean of these metrics, providing a more accurate vision than the arithmetic mean. Besides, when a precision and a recall are available, the F1-score is systematically used. The retained metrics are therefore the following:

1. **Reference-based metrics**:
   - Overlap (ROUGE-1, ROUGE-2, ROUGE-L, chrF, chrF++);
   - Semantic similarity (BERTScore, MoverScore).
2. **Source-based metrics**:
   - Text generation (BARTScore);
   - Question Answering (custom QuestEval, see Section 3.3.4);
   - Self check (simplified SelfCheckGPT-Prompt, see Section 3.3.5);
   - Multi check (SelfCheckGPT-BERTScore);
   - Abstractiveness (see Section 3.3.6).
   - Unique words rate (to evaluate the diversity within the summary).

3.3.2 Existing implementations

Concerning the individual implementation of each metric, we use, when available, the official implementation to be consistent with the literature. For ROUGE, chrF and BERTScore, we therefore work with the implementation of the `torchmetrics` Python package [55]. For the MoverScore [11] and the BARTScore [12], we use the official implementations from their respective papers.

3.3.3 Personal implementations

In addition to the existing implementations stated above, we propose several personal implementations, for the following metrics:

- **Question Answering**: a variation of QuestEval (see Section 3.3.4);
- **Self check**: an adaptation of SelfCheckGPT-Prompt, more suited for summarization (see Section 3.3.5);
- **Multi check**: an implementation of SelfCheckGPT-BERT;
- **Abstractiveness**: a measure of the novelty rate of the summary (see Section 3.3.6).

3.3.4 Q&A metric implementation

The original implementation of QuestEval [15] has several disadvantages: it is based on exclusively English Q&G and Q&A models; it has a custom, deep, weighter model; and it evaluates the generated answers using overlap metrics. Because of these, we propose a personal implementation, that makes it more relevant for our data. It is based on the following French models:

1. For **Question-Answering**: The model `etalab-ia/camembert-base-squadFR-fquad-piaf` developped by Etalab, which uses a CamemBERT model [40], fine-tuned on context-question-answer triplets from three French Q&A datasets [56, 57, 58].

2. For **Question-Generation**: The model `lincoln/barthez-squadFR-fquad-piaf-question-generation`, which uses a Barthez model [42], fine-tuned on context-question pairs from the same three datasets [56, 57, 58].

While based on the original QuestEval architecture, this custom implementation differs from it in several ways. It works as follows:

1. **Precision**:
   - Generate one question per sentence of **P**.
   - Answer each of these questions with **S** as context (predictions).
   - Answer each of these questions with **P** as context (references).
   - Compute the average BERTScore between the predictions and the references.

2. **Recall**:
   - Generate one question per sentence of **S**.
   - Weight these questions based on the TextRank scores of the sentences of **S**.
   - Answer each of these questions with **P** as context (predictions).
   - Answer each of these questions with **S** as context (references).
   - Compute the weighted average BERTScore between the predictions and the references.

The main differences with the original QuestEval architecture therefore lie in the models of Q&G and Q&A themselves, in the precision and recall scores computation (BERTScore instead of tokens overlap), and in the recall weighter (TextRank weights instead of a custom neural model).

3.3.5 Self check metric implementation

We propose a simplified LLM-based metric, inspired by SelfCheckGPT-Prompt. One problem of the SelfCheckGPT metrics is their computation time: they involve generating several samples, and then perform a lot of computations at the sentence level, for each sample. Besides, these metrics are suited for any text generation task, but can be adapted for text summarization, which is a source-based task.

In our custom metric, instead of asking the LLM if a given sentence is supported by each sample, we directly ask the model if the sentence is supported by the source. This provides us with a fast, efficient, source-based factuality metric. More precisely, for each sentence of the prediction to be evaluated, we furnish the following prompt to the LLM:

> *CONTEXTE : {source} ##### PHRASE : {phrase} ##### La phrase est-elle vraie, selon le contexte ci-dessus ? Réponds uniquement par Oui ou Non, sans te justifier.*

This can be translated by:

> *CONTEXT: {source} ##### SENTENCE: {sentence} ##### Is the sentence true, according to the context provided above? Answer only by Yes or No, without justification.*

For each sentence, the score is 1 if the model answers Yes, 0 if the model answers No, and 0.5 if the answer is undetermined (i.e. does not contain Yes nor No). Results presented in Section 4.1.1 show that this metric, although simplified compared to original SelfCheckGPT metrics, is more efficient than SelfCheckGPT-BERT. We tried using different LLMs as factuality checkers, and experiments show that using Mistral 7B was almost as efficient as using Llama-2 70B, while much quicker and lighter.

3.3.6 Abstractiveness metric implementation

We propose a simple metric to quantify the level of abstraction, or abstractiveness, of a prediction, with respect to the source. Abstractiveness, i.e. the amount of novelty of a prediction, can be seen as a comparison between the surface form of words. It is therefore natural to measure the rate of n-grams novelty, for different values of n, between the prediction and the source. More precisely, we decide to take the unigrams, bigrams, trigrams and fourgrams novelty rates into account, and to combine these rates into one single metric, by considering their geometric mean. Precisely, the metric can be computed with Equation (18).

$$\text{Abstractiveness}\,(\mathbf{P},\mathbf{S}) = \exp\left[\frac{1}{4}\sum_{n=1}^{4}\ln\left(\frac{\sum_{w\in\mathbf{P}^n} 1\,[w \notin \mathbf{S}^n]}{|\mathbf{P}^n|}\right)\right] \tag{18}$$

where $\mathbf{P}^n$ is the set of n-grams of the prediction, and $\mathbf{S}^n$ is the set of n-grams of the source. The main goal of this metric is to determine whether or not the level of abstraction is somehow related to the quality of a summary, be it in terms of content or factuality. It is also useful to spot models that have a tendency to extract sentences or expressions from the source to create the prediction.

3.3.7 Evaluating evaluation metrics

To compare evaluation metrics, and especially factuality metrics, the standard framework is to compute correlation with human judgment, which is considered as the gold judgment. Of course, this requires having access to a human-annotated, French dataset. To the best of our knowledge, no dataset with such requirements is publicly available. However, authors of [16] released an English, human-annotated factuality dataset, composed of Wikipedia articles and GPT3-generated summaries (where hallucinations have artificially been added). Each summary has then been evaluated, at a sentence level, by 3 workers: each of them classified each sentence as ‘Accurate’, ‘Minor Inaccurate’ or ‘Major Inaccurate’, given the article as context.

We use this dataset as basis for our evaluation. As a first step, we translate the articles and their candidate summaries using the high-quality translator Google Translate (for this public, non-sensitive data, using such tools is not problematic in terms of data governance). We then compute a human score, based on the annotations presented above: the score of each summary is the average of the sentence scores, that are equal to 1 if the annotation is ‘Accurate’, 0.5 if the annotation is ‘Minor Inaccurate’, and 0 if it is ‘Major Inaccurate’.

Modifying this dataset for our specific need therefore allows us to compute correlation between human factuality judgment, and each source-based factuality metric. These results are presented in Section 4.1.1.

3.3.8 Work done and experiments conducted

Here is a summary of the work done concerning evaluation metrics, and of the experiments conducted, the results of which will be presented in Section 4.1:

1. **Implementation of several existing content and factuality metrics.**
2. **Creation of new factuality metrics.** See Sections 3.3.4, 3.3.5 and 3.3.6.
3. **Evaluation of factuality metrics.** We compute the correlation of each metric with human judgment on a human-labeled dataset, as explained in Section 3.3.7.
4. **Comparison of evaluation metrics.** We generate a set of summaries (produced by several different language models, for different source texts), and compute the correlation between each metric.

3.4 Extractive summarization

3.4.1 TextRank

For this approach, we rely on the PageRank implementation of the `networkx` [59] Python package. We then use this implementation to apply the PageRank on a textual document, building our own implementation of the TextRank algorithm. This involves classically pre-processing the text (tokenization, lowercase conversion, lemmatization, stopwords and punctuation removal), before vectorially representing each word and then each sentence (by averaging the representation of the words that make up the sentence). Once this is done, we compute cosine similarities between all pairs of sentences, and deduce a similarity matrix and a graph representation of the text. We can then simply apply the PageRank algorithm of the `networkx` package to order the sentences according to their relevance and importance within the source text.

Compared to the original paper [17], we besides propose two improvements.

Instead of TF-IDF weights as in the original paper, we use word embeddings to vectorially represent words: it was not the case when the TextRank paper was released back in 2004, but using word embeddings is now a usual and efficient approach to represent words as vectors. We tried different embeddings, namely Word2Vec [19] and FastText [20], without noticing a significant difference between both. For the results presented in Section 4.2.2, we therefore chose to use French 200-dimensional Word2Vec embeddings, since they are lighter and faster for inference.

One main drawback of the TextRank algorithm, as explained in Section 2.3.1, is its incapacity to avoid redundancy between extracted sentences. To overcome this issue, we propose a threshold-based selection. When adding a new sentence to the current summary, we compute the cosine similarity between the embedding of this sentence and that of every other sentence already in the summary. If one of these similarities is greater than a given threshold (i.e. this candidate sentence is too close semantically from another sentence of the summary), then this sentence is rejected. This approach enhances the diversity among the sentences of the extracted summary, therefore increasing the information contained in the same number of sentences. Experiments are carried out to evaluate the impact of the threshold on the quality of the summary: results are presented in Section 4.2.1.

3.4.2 K-Means

For this algorithm, we base our work on the K-Means implementation of the `sklearn` [60] Python package. To apply it, we go through the same first steps than for the TextRank implementation (pre-processing, word and sentence embeddings, similarity matrix, graph representation).

3.4.3 Greedy algorithm

We propose a novel approach for extractive text summarization, inspired by [61]. This article aims at proposing different methods to find an upper-bound ROUGE score for extractive summarization methods. Indeed, extractive methods are almost systematically evaluated using reference-based metrics (such as ROUGE or BERTScore), with respect to abstractive, human-written references. It is therefore interesting to determine the gold *extractive* summary, which maximizes these metrics, simply to have an idea of what performance is achievable by extractive methods. One way to achieve this, described in this article, is to greedily add sentences from the source to the prediction, while this increases the global ROUGE score of the summary thus formed.

This technique is purely indicative and usable only in a research and development context, since it requires having access to gold abstractive references. However, the principle of optimizing a reference-based metric with a greedy algorithm can be adapted, to optimize a given *source-based* metric, provided it is relevant enough. One such metric is the BARTScore, presented in Section 2.2.2. The proposed approach is therefore a simple greedy algorithm, that works as follows. At each step of the algorithm, we add to the current summary the sentence that increases the BARTScore the most, if this sentence exists. If no sentence increases the score, or if the maximum number of sentences has been reached, then the algorithm ends. The optimization carried out by this algorithm is however not exact, but exhaustively exploring all possibilities is computationally infeasible, and the greedy algorithm makes it possible to efficiently approach the upper-bound of this score.

Since the correlation of the BARTScore is not high with all other evaluation metrics (see Section 4.1.2), it is *a priori* not sure that this algorithm will yield very good results in terms of other metrics. However, results (see Section 4.2.2) show a significant improvement over more classical methods such as TextRank and K-Means.

3.4.4 Work done and experiments conducted

Here is a brief synthesis of the work done concerning extractive summarization, and of the experiments conducted, the results of which will be presented in Section 4.2:

1. **Implementation of the TextRank algorithm**. We convert a textual document to a suitable graph representation, to apply an existing implementation of the PageRank. We also add a threshold mechanism, as explained in Section 3.4.1, to increase semantic diversity.
2. **Analysis of the impact of the threshold mechanism on the TextRank results.** We vary the threshold value (from 0.5, i.e. a very selective threshold, to 1, i.e. no threshold at all), and compare the average performances of the TextRank algorithm for these different values.
3. **Adaptation of the K-Means algorithm for a summarization task.**
4. **Creation and implementation of a greedy algorithm, based on BARTScore optimization.** See Section 3.4.3.
5. **Objective comparison of the three implemented methods.** We choose to create summaries containing at most 6 sentences (the average length of the summaries in our dataset), and we evaluate each algorithm, from a content perspective, with reference-based metrics.

3.5 Abstractive summarization

3.5.1 General methods

One central point when using abstractive methods is the management of long documents. In the case of audition reports, it is quite common to encounter documents of several pages, which exceed the context length of models such as Llama-2 or Mistral (4096 tokens, see Section 2.4.4). In this case, we choose a hybrid approach, i.e. we first use an extractive method, as a sort of pre-processing step, to reduce the length of the document while preserving its main information, before passing it to an abstractive model. We decide to use the TextRank algorithm for this extractive step, for two main reasons:

1. It is very fast: while a greedy algorithm as presented in Section 3.4.3 is more efficient than the TextRank (see Section 4.2.2), it is also way longer (several seconds for the TextRank, against several tens or even hundred seconds for the greedy algorithm). Since the inference time of the LLM later used must be added to this pre-processing time, it is more relevant to consider the fastest choice, even if it comes with a slight drop in performances.
2. The number of sentences extracted can be chosen post-computation: the number of sentences is here not fixed, since we want to add as many sentences as possible, while remaining in the token limit of the model. As each sentence of each text has a different number of tokens, it is essential to run the algorithm once and for all, and incrementally increase the summary length. This can not be done with the K-Means approach (see Section 2.3.2).

3.5.2 Inference use and fine-tuning

For the inference of LLMs, we rely on the Huggingface implementation, based on the `transformers` [62] package. All experiments are carried out with contexts of 4096 tokens, and with a generation limit of 512 tokens. The prompts given to the models are formatted, according to Llama-2 and Mistral formats, as follows:

<s>[INST] {instruction} ##### {text} [/INST]

For inference use, seeing very little difference between quantized and non-quantized small models (Llama-2 7B and Mistral 7B), we choose to use the built-in 8-bit quantization of the `transformers` package for all models.

Concerning models fine-tuning, we exclusively consider LoRA fine-tuning, for reasons explained in Section 2.4.6. We use the implementation of `peft` [63] and `trl` [64] packages from Huggingface. An important question when fine-tuning models is that of hyperparameters. After a lot of empirical tests, here are the main hyperparameters we use:

- Learning rate: 5×10^{-5};
- Target modules: all linear modules;
- LoRA rank: 64;
- LoRA alpha: 128;
- Optimizer: `paged_adamw_32bit`;
- Effective batch size: 8.

In particular, the number of trainable parameters were between 2% and 3% of the original number of parameters of the model.

3.5.3 Prompt engineering

As explained in Section 2.4.5, prompt engineering is mainly empirical. To prevent subjectivity and too much specificity in the prompt creation process, and to increase generalization capacities of the created prompts, we use the following methodology:

1. Select empirically promising prompts, based on subjective performances on a few examples;
2. Objectively compare these prompts on a validation set, using previously defined metrics;
3. Keep the few prompts that yield the best results.

Examples of prompts thus created and selected are presented in Section 4.3.2, as well as the comparison between basic and prompt-engineered prompts.

3.5.4 Retrieval augmented prompting

We propose a way to automatize and adapt the few-shot prompting technique, based on Retrieval Augmented Generation (RAG) [65]. Usually, RAG is used to extract a relevant piece of text, from a corpus of texts, to help the model answer a specific question more accurately, or to provide it with external knowledge, in the context of a chatbot for instance. We adapt this retrieval step to select a given number of texts, from a corpus of text/summary pairs, that are semantically close to the text to summarize (for this semantic search, we embed the source text, as well as the corpus texts, with the `sentence-transformers` package [66], and use cosine similarity to compute the closest texts in the corpus). We then provide the model with the summaries associated to these selected texts, as examples, to help it capture the writing style and/or the kind of important elements to include.

3.5.5 Work done and experiments conducted

Here is a short recap of the work done concerning abstractive summarization, and of the experiments conducted, the results of which will be presented in Section 4.3:

1. **Inference use and comparison of different models.** We try the different versions of Llama-2 (7B, 13B, 70B), Mistal 7B and Mixtral 8x7B, and perform an objective comparison of these models, on a validation set, using previously defined metrics. We uniformly sample the prompts from a set of basic summarization prompts, as explained in Section 4.3.1.

2. **Prompt-engineering and prompts comparison.** We create more evolved prompts empirically, as explained in Section 3.5.3. We then perform an objective evaluation of these prompts, by evaluating different models with these prompts (instead of basic prompts), using previously defined metrics.

3. **Implementation of a retrieval-augmented prompting framework.**

4. **Analysis of the impact of retrieval-augmented prompting.** We vary the number of example summaries added in the prompt (from 0 to 4), and evaluate Llama-2 70B for these different cases.

5. **Hyperparameters tuning for LoRA fine-tuning.** Because seriously evaluating a model is quite costly, especially in terms of time, this step was done empirically.

6. **LoRA fine-tuning of Mistral 7B.**

Chapter 4

Results

This Chapter presents the main results of the experiments described in Chapter 3, along with interpretations and analyses. The results concerning evaluation metrics are presented in Section 4.1. Then, the results for extractive summarization and abstractive summarization are provided in Sections 4.2 and 4.3.

4.1 Evaluation metrics

4.1.1 Correlation with human judgment

Figure 17 presents the correlation of factuality metrics, between themselves and with human judgment, as explained in Section 3.3.7. These metrics are ranked in Table 3, according to their correlation with human judgment. The metric 'Self check', which corresponds to our personal, simplified implementation of SelfCheckGTP-Prompt (see Section 3.3.5), outperforms all other metrics from a large margin. In particular, it seems that generating several samples and comparing the BERTScore between these samples and the prediction (done in the 'Multi check' metric, which is our implementation of SelfCheckGPT-BERT) is clearly less efficient than simply using the LLM itself as double-checker.

Notably, our personal implementation of a Question Answering metric underperforms compared to other factuality metrics, which can be explained by the quality of the Q&G and Q&A models. Even though they are state-of-the-art models on French data, they lack flexibility, and can therefore sometimes focus on non-central elements.

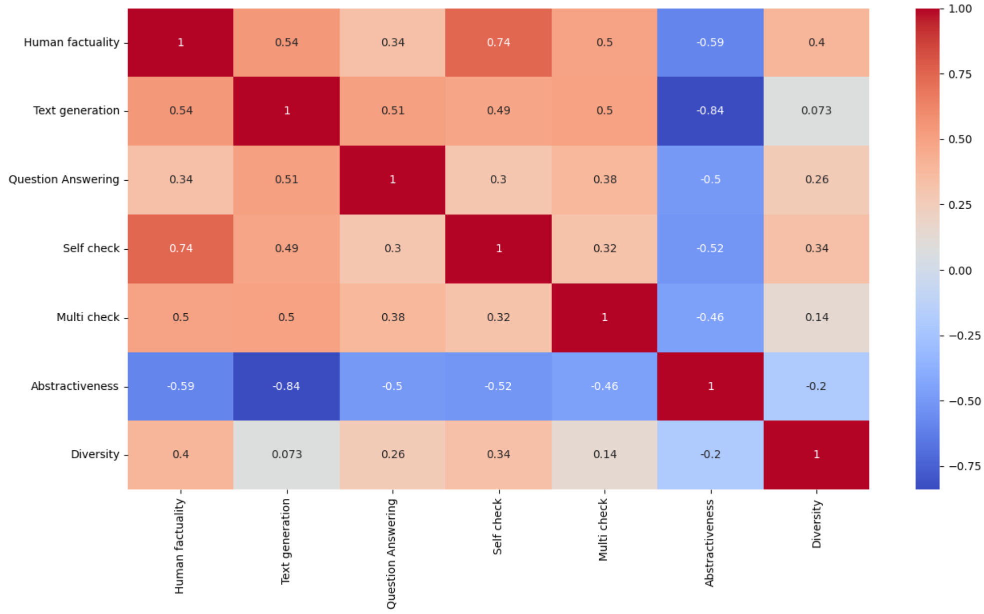


Figure 17: Correlation heatmap of factuality metrics – Results on [16] dataset (238 samples, translated in French with Google Translate). Pearson correlation coefficient was used to compute correlation. The field 'Human judgment' is considered as the gold reference.

Ranking	**Evaluation paradigm**	**Correlation**
1	Self check	0.743
2	Text generation	0.542
3	Multi check	0.496
4	Question Answering	0.335

Table 3: Correlation of several factuality metrics with human gold judgment – Results on [16] dataset (238 samples, translated in French with Google Translate). Pearson correlation coefficient was used to compute correlation. Llama-2 70B was used as sampler for 'Multi check' and as judge for 'Self check'. Higher is better.

Because of the poorer results of our Question Answering metric and of our implementation of SelfCheckGPT-BERT, the models and methods will be compared, in the following paragraphs, using only 'Text generation' and 'Self check' paradigms (for factuality evaluation).

4.1.2 Correlation between metrics

Apart from the correlation with a gold, human judgment, it is also interesting to notice how the different metrics correlate with each other. Figure 18 show the correlation heatmap of the different metrics presented or developed earlier.

The first thing to notice is the very high correlation between overlap and semantic similarity metrics (i.e. the two metrics focusing on content evaluation). In particular, it highlights the fact that old metrics such as ROUGE (tokens overlap) or chrF (characters overlap) remain relevant, even with more advanced metrics such as BERTScore being developed, and it explains why the ROUGE score remains a must for summarization evaluation.

Concerning factuality metrics, it is striking that the 'Self check' metric, which correlates with human judgment the most, does not, on our data, correlate with other factuality metrics. This could be due to the fact that using an LLM to check information has more flexibility than our Question-Answering metric for instance, and therefore does not focus on the same elements. Thus, these metrics could possibly be complementary for factuality evaluation.

Besides, it seems that the abstractiveness of a summary does not have a specific impact on its factuality, but rather a global impact on its quality (in terms of both content and factuality).

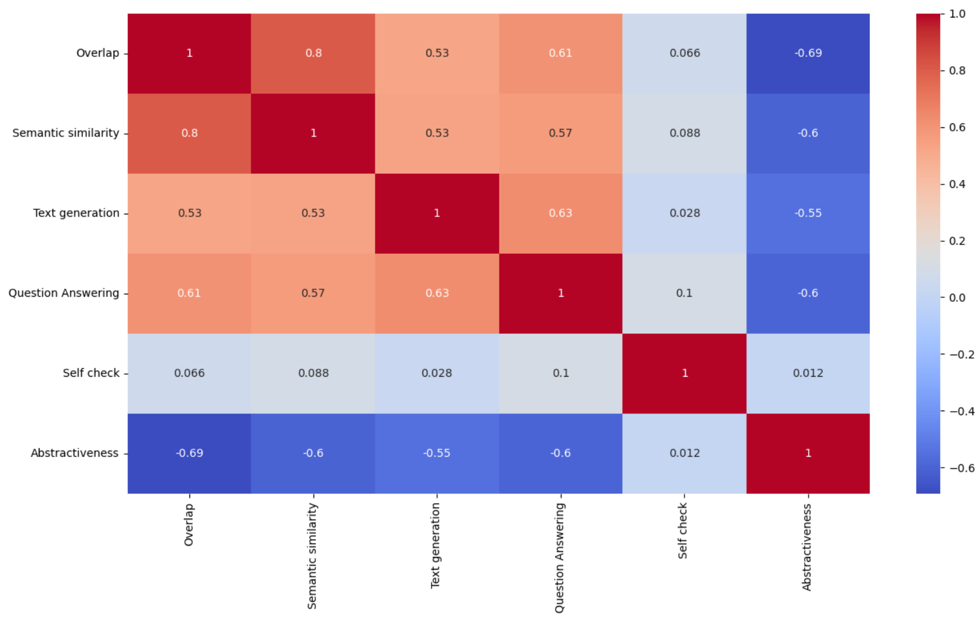


Figure 18: Correlation heatmap of evaluation metrics – Based on the evaluation of 1075 summaries (generated by various LLMs based on the evaluation dataset). Pearson correlation coefficient was used to compute correlation. Higher is better.

4.2 Extractive summarization

Since extractive summaries only include sentences from the source, there is no interest in evaluating their factuality: they are by nature factual. The hallucination risk and factuality evaluation therefore only concern the abstractive methods. The following paragraphs present the main experiments conducted for extractive methods and their results.

4.2.1 Threshold impact for the TextRank algorithm

Table 4 shows the impact of the use of a threshold, as explained in Section 3.4, in the TextRank algorithm. The lower the threshold, the more demanding the algorithm will be about novelty, when adding a new sentence to the summary. This naturally results in a unique words rate (i.e. a diversity among the sentences of the summary) that is higher for a lower threshold, and that decreases as the threshold increases. It is also worth noticing that a basic TextRank without this threshold mechanism (i.e. with a threshold of 1, in the last column) is clearly less efficient than a moderate threshold (0.8 or 0.9): redundancy is quite negative, and can be easily reduced without losing in relevance.

	Threshold	**0.5**	**0.6**	**0.7**
Cont.	Overlap	0.163 ∓ 0.006	0.168 ∓ 0.007	0.179 ∓ 0.005
Cont.	Semantic similarity	0.497 ∓ 0.008	0.502 ∓ 0.006	0.510 ∓ 0.009
	Unique words rate	**0.852 ∓ 0.012**	0.819 ∓ 0.011	0.780 ∓ 0.013
	Threshold	**0.8**	**0.9**	**1**
Cont.	Overlap	**0.180 ∓ 0.007**	0.176 ∓ 0.007	0.172 ∓ 0.006
Cont.	Semantic similarity	**0.513 ∓ 0.008**	0.509 ∓ 0.007	0.506 ∓ 0.006
	Unique words rate	0.754 ∓ 0.012	0.735 ∓ 0.010	0.719 ∓ 0.012

Table 4: Threshold impact for the TextRank algorithm – Results on the evaluation dataset (215 samples), for summaries of 6 sentences. The best result for each metric appears in **bold**. Higher is better. The confidence intervals are computed at the 95% level, i.e. as $\mu \mp 1.96\sigma/\sqrt{N}$ where $N = 215$ is the number of samples, μ is the empirical mean, and σ the standard deviation.

4.2.2 Methods comparison

		TextRank	**K-Means**	**Greedy**
Cont.	Overlap	0.172 ∓ 0.010	0.164 ∓ 0.007	**0.190 ∓ 0.011**
Cont.	Semantic similarity	0.506 ∓ 0.012	0.499 ∓ 0.009	**0.513 ∓ 0.010**
	Text generation	-0.766 ∓ 0.075	-1.028 ∓ 0.068	**-0.285 ∓ 0.053**
	Unique words rate	0.719 ∓ 0.016	**0.823 ∓ 0.017**	0.791 ∓ 0.014
	Execution time (s)	2.3	**1.1**	62.2

Table 5: Extractive methods comparison – Results on the evaluation dataset (215 samples), for summaries of 6 sentences. The best result for each metric appears in **bold**. Higher is better. The confidence intervals are computed at the 95% level, i.e. as $\mu \mp 1.96\sigma/\sqrt{N}$ where $N = 215$ is the number of samples, μ is the empirical mean, and σ the standard deviation.

Table 5 presents a comparison of the different extractive methods considered. The results are quite clear: our greedy algorithm presented in Section 3.4.3 clearly outperforms the TextRank and

K-Means algorithms, both in terms of overlap and semantic similarity. However, it is also much longer, because of the high number of calls to the BARTScore function, and its time complexity grows exponentially with the text length, which is not the case for the TextRank or the K-Means. These observations make the greedy algorithm a more relevant choice for short texts, or if the computation duration is not a problem. In our specific use case, time is important, and source texts can be very long: this makes the TextRank a more suitable choice.

4.3 Abstractive summarization

4.3.1 Pre-trained LLMs

	Model name	Llama-2		
	# Parameters	**7B**	**13B**	**70B**
Cont.	Overlap	0.083 ∓ 0.005	0.159 ∓ 0.011	**0.191 ∓ 0.009**
Cont.	Semantic similarity	0.453 ∓ 0.005	0.501 ∓ 0.008	**0.529 ∓ 0.007**
Fact.	Text generation	-3.099 ∓ 0.069	-2.718 ∓ 0.086	**-2.479 ∓ 0.063**
Fact.	Self check	0.616 ∓ 0.036	0.634 ∓ 0.036	0.696 ∓ 0.037
	Abstractiveness	0.924 ∓ 0.018	0.704 ∓ 0.029	0.571 ∓ 0.017
	Unique words rate	0.629 ∓ 0.012	0.696 ∓ 0.012	0.668 ∓ 0.013

	Model name	Mistral	Mixtral
	# Parameters	**7B**	**8x7B**
Cont.	Overlap	0.113 ∓ 0.009	0.185 ∓ 0.010
Cont.	Semantic similarity	0.471 ∓ 0.011	0.527 ∓ 0.007
Fact.	Text generation	-2.600 ∓ 0.103	-2.575 ∓ 0.086
Fact.	Self check	0.609 ∓ 0.034	**0.699 ∓ 0.038**
	Abstractiveness	0.732 ∓ 0.039	0.577 ∓ 0.020
	Unique words rate	0.592 ∓ 0.012	**0.725 ∓ 0.015**

Table 6: Pre-trained LLMs comparison (basic prompts) – Results on the evaluation dataset (215 samples), using basic summarization prompts (see Table 7). The confidence intervals are computed at the 95% level, i.e. as $\mu \mp 1.96\sigma/\sqrt{N}$ where N = 215 is the number of samples, μ is the empirical mean, and σ the standard deviation. Higher is better.

Table 6 shows a comparison of different models presented in Section 2.4.4, using basic summarization prompts, examples of which are given in Table 7. From this comparison, it is clear that increasing the size of the models yields incomparably better performances. It is also interesting to notice the performance of Mistral 7B, which distinctly outperforms Llama-2 7B, but is not yet comparable to larger models, as such.

The underperformance of Llama-2 7B can besides be explained by its difficulties to generate French text. Because it was trained on a majority of English text, it has a tendency to answer in English (which can be noticed thanks to its extremely high abstractiveness), even when explicitly asked to answer in French. Apart from prompt engineering and fine-tuning, one way to address this issue would be to detect English answers, and translate them in French with an efficient translation model, as discussed in Section 5.3.

4.3.2 Prompt engineering

As explained earlier, prompt engineering has mainly been empirical. Table 7 presents a few examples of basic, generic summarization prompts, while Table 8 shows several examples of more detailed, specific prompts, established after empirical experiments. To objectively study the impact of the use of such prompts, a systematic comparison has been carried out, the results of which are presented in Table 9.

What are the main points of the following text?
Write a short summary of the following text.
How would you reformulate the following text in a few words?

Table 7: Examples of basic summarization prompts – Translated from French

The following text is an audition report. You should write a short version, which summarizes the main factual information in 2-3 sentences. You must speak in French, in the 3rd person, in a simple and easily understandable style, and without including external elements.
Can you summarize the following text in French? This is a report of a hearing, during which a witness, a victim or a suspect was heard. This text was written by a gendarme or a police officer. You must write the summary in French, in a few concise sentences, from an external, neutral and factual point of view.
You must produce a short summary of the following report, like a police officer, in 3-4 sentences. You absolutely must speak in French, in the 3rd person, and use a simple and easily understandable style. You must include all factual information and important details, but only those present in the text.

Table 8: Examples of prompt-engineered prompts – Translated from French

	Model name	Llama-2		
	# Parameters	**7B**	**13B**	**70B**
Cont.	Overlap	0.124 ∓ 0.010	0.194 ∓ 0.009	**0.206 ∓ 0.008**
	Semantic similarity	0.483 ∓ 0.006	0.526 ∓ 0.005	**0.538 ∓ 0.004**
Fact.	Text generation	-2.882 ∓ 0.084	-2.691 ∓ 0.059	-2.409 ∓ 0.058
	Self check	0.701 ∓ 0.038	**0.712 ∓ 0.036**	0.707 ∓ 0.037
	Abstractiveness	0.805 ∓ 0.027	0.632 ∓ 0.018	0.574 ∓ 0.014
	Unique words rate	0.635 ∓ 0.013	0.729 ∓ 0.009	0.678 ∓ 0.011

	Model name	Mistral	Mixtral
	# Parameters	**7B**	**8x7B**
Cont.	Overlap	0.191 ∓ 0.010	0.192 ∓ 0.008
	Semantic similarity	0.535 ∓ 0.007	0.528 ∓ 0.006
Fact.	Text generation	**-1.824 ∓ 0.074**	-2.704 ∓ 0.061
	Self check	0.639 ∓ 0.032	0.670 ∓ 0.030
	Abstractiveness	0.440 ∓ 0.018	0.562 ∓ 0.021
	Unique words rate	0.645 ∓ 0.010	**0.746 ∓ 0.011**

Table 9: Pre-trained LLMs comparison (prompt-engineered prompts) – Results on the evaluation dataset (215 samples), using prompt-engineered summarization prompts (see Table 8). The confidence intervals are computed at the 95% level, i.e. as $\mu \mp 1.96\sigma/\sqrt{N}$ where N = 215 is the number of samples, μ is the empirical mean, and σ the standard deviation. Higher is better.

Several observations are to be made based on these results. First, the usefulness of prompt engineering is very clear, in terms both of content and factuality, and especially for smaller models. The results for Mistral 7B are particularly striking: a relatively small model, if prompted accurately, can see its performance significantly improved.

However, we only notice a slight increase in performance due to prompt engineering on Llama-2 70B and Mixtral 8x7B. Possibly, the results with basic prompts are already pretty good, and the models face an intrinsic limit, due to the (judicial) nature of the data: this limit could be approached, but not exceeded, with such empirical prompt engineering.

4.3.3 Retrieval augmented prompting

	Number of examples	**0**	**1**	**2**	**3**	**4**
Cont.	Overlap	0.201 ∓ 0.011	0.205 ∓ 0.010	**0.209 ∓ 0.009**	0.207 ∓ 0.009	**0.209 ∓ 0.010**
	Semantic similarity	0.533 ∓ 0.007	0.538 ∓ 0.005	0.539 ∓ 0.005	**0.540 ∓ 0.007**	0.539 ∓ 0.005
Fact.	Text generation	**-2.470 ∓ 0.054**	-2.572 ∓ 0.051	-2.506 ∓ 0.062	-2.498 ∓ 0.061	-2.490 ∓ 0.065
	Self check	**0.723 ∓ 0.035**	0.673 ∓ 0.032	0.662 ∓ 0.038	0.663∓ 0.032	0.673 ∓ 0.030
	Abstractiveness	0.594 ∓ 0.018	0.611 ∓ 0.015	0.606 ∓ 0.016	0.599 ∓ 0.017	0.603 ∓ 0.016
	Unique words rate	**0.700 ∓ 0.009**	0.686 ∓ 0.012	0.679 ∓ 0.010	0.686 ∓ 0.010	0.675 ∓ 0.009

Table 10: Retrieval augmented prompting results – Results on the evaluation dataset (215 samples). The model used here is Llama-2 70B. No example means no retrieval augmented prompting. The confidence intervals are computed at the 95% level, i.e. as $\mu \mp 1.96\sigma/\sqrt{N}$ where N = 215 is the number of samples, μ is the empirical mean, and σ the standard deviation. Higher is better.

Table 10 presents the result of our retrieval-augmented prompting strategy, explained in Section 3.5.4. Adding a few close summaries as examples to the prompt yields slightly better performances in terms of content. However, results also tend to be less factual: the model may be disturbed by the additional information contained in the examples. Overall, this slight content improvement is not important enough to increase the hallucination risk: this few-shot prompting technique based on RAG, although it could be useful and efficient for other use cases, is not relevant for our objective.

4.3.4 LoRA fine-tuning

	Number of epochs	**0**	2	4
Cont.	Overlap	0.113 ∓ 0.009	0.132 ∓ 0.011	0.151 ∓ 0.012
	Semantic similarity	0.468 ∓ 0.007	0.490 ∓ 0.005	0.499 ∓ 0.005
Fact.	Text generation	-2.557 ∓ 0.074	-2.725 ∓ 0.070	-2.423 ∓ 0.076
	Self check	0.604 ∓ 0.032	0.629 ∓ 0.038	0.629 ∓ 0.035
	Abstractiveness	0.729 ∓ 0.019	0.776 ∓ 0.016	0.676 ∓ 0.021
	Unique words rate	0.586 ∓ 0.009	0.708 ∓ 0.010	0.729 ∓ 0.012

	Number of epochs	**6**	**8**	**10**
Cont.	Overlap	0.179 ∓ 0.013	0.167 ∓ 0.010	0.174 ∓ 0.009
	Semantic similarity	0.522 ∓ 0.007	0.514 ∓ 0.006	0.519 ∓ 0.007
Fact.	Text generation	-2.104 ∓ 0.068	-2.185 ∓ 0.071	-2.099 ∓ 0.073
	Self check	0.650 ∓ 0.033	0.650 ∓ 0.037	**0.668 ∓ 0.035**
	Abstractiveness	0.555 ∓ 0.018	0.592 ∓ 0.020	0.559 ∓ 0.018
	Unique words rate	0.717 ∓ 0.010	**0.731 ∓ 0.009**	0.725 ∓ 0.011

	Number of epochs	**12**	**14**	**16**
Cont.	Overlap	**0.185 ∓ 0.010**	0.180 ∓ 0.012	0.178 ∓ 0.009
	Semantic similarity	**0.525 ∓ 0.006**	0.523 ∓ 0.008	0.522 ∓ 0.007
Fact.	Text generation	-2.060 ∓ 0.069	-2.066 ∓ 0.067	**-2.034 ∓ 0.071**
	Self check	0.667 ∓ 0.035	0.650 ∓ 0.032	0.651 ∓ 0.038
	Abstractiveness	0.539 ∓ 0.015	0.542 ∓ 0.014	0.533 ∓ 0.017
	Unique words rate	0.729 ∓ 0.010	0.716 ∓ 0.013	0.713 ∓ 0.012

Table 11: LoRA fine-tuning results on Mistral 7B – Results on the evaluation dataset (215 samples). The model was trained on the training dataset (1632 samples), with the following hyperparameters: learning rate = 5×10^{-5}, all linear modules targeted, LoRA rank = 64, LoRA alpha = 128, optimizer = `paged_adamw_32bit`, effective batch size = 8. The first column (0 epoch) represents the base model. The model was trained and evaluated using always the same basic summarization prompt. The confidence intervals are computed at the 95% level, i.e. as $\mu \mp 1.96\sigma/\sqrt{N}$ where N = 215 is the number of samples, μ is the empirical mean, and σ the standard deviation. Higher is better.

Concerning parameter-efficient fine-tuning with LoRA, one important question was that of the prompts to use for fine-tuning. Indeed, fine-tuning a model on a set of basic prompts would give no insurance that this fine-tuned model would yield improve performances with optimized prompts. On the other side, fine-tuning the model on a set of optimized prompts would likely be too rigid, and the fine-tuned model would probably behave strangely on different prompts. After having tried these different approaches, we have chosen to always use the same basic prompt, "*Summarize the following text*". The idea here would be to propose an improved model, independently of the user, by removing the empirical aspect of prompt engineering.

Table 11 presents the fine-tuning results for the model Mistral 7B. These results are quite classical: first, the model impressively gets better; then, it reaches a performance peak (around 12-14 epochs); and finally, its performance slightly decreases, probably because it starts overfitting the training data.

Overall, these results are really impressive: the best Mistral 7B fine-tuned model (at epoch 12) matches or outperforms Mixtral 8x7B and Llama-2 70B on several evaluation metrics (both base model and with prompt engineering, see Tables 6 and 9). These results are particularly satisfying, because this makes it possible to use a very light model that yields performances comparable to those of much bigger models (10 times its size for Llama-2 70B), which is game-changing for inference use. Besides, making the summarization prompt constant reduces the probability of having a worse result due to a bad user prompt.

4.4 Examples of generated summaries

Tables 12 and 13 show two examples of extractive summaries, generated by the TextRank and the greedy algorithm. Since they are extractive, as expected, they are hard to read and really lack coherence. Some sentences are not related to the previous ones, or are somehow unfinished. This simply emphasizes that extractive methods are not suitable as a final summary.

En vous attendant, j'ai fait le tour de la maison avec ma compagne pour voir ce qui avait été fouillé et ce qui avait pu être dérobé. Avant les faits ou le jour des faits, avez-vous remarqué un individu faire du repérage près de votre habitation ? Avant les faits ou le jour des faits, est ce que vous avez été démarché par quelqu'un ? Suite à la constatation du cambriolage, avez-vous remarqué que des objets dans votre domicile auraient été déplacés ? Car une pièce en or, j'ai regardé sur Internet, ça vaut environ 300€. Et pour la gourmette, mes parents m'ont dit que ce n'était pas de l'or pur et que ça valait moins de 100€. J'ai une facture pour 7 pièces en or que mes parents avaient achetées dans les années 80. Pouvez-vous nous donner la disposition de votre maison ?

Table 12: Example of extractive summary #1 – Generated with the TextRank algorithm

Entendons la personne dénommée ci-dessus qui nous déclare : Je me présente ce jour à votre unité afin de déposer plainte pour le cambriolage survenu à mon domicile entre le 21/10/2022 à 15h00 et le 22/10/2022 à 17h30, alors que j'étais absent. En partant, nous avons laissé les volets des fenêtres ouverts, car nous comptions repasser dans l'après-midi. Finalement, nous n'avons pas eu le temps. Nous avons passé la soirée chez ma compagne à son domicile à Chateauroux. Le lendemain, le 21/10/2022, nous sommes revenus à mon domicile vers 17h30.

Table 13: Example of extractive summary #2 – Generated with the greedy algorithm

Table 14 shows, for the same audition source text, a summary generated by Mistral 7B with a basic summarization prompt. The first thing to notice is that the model fully answers in English. Besides, it is not very concise, focuses on details that are not central.

The person is reporting a burglary that occurred at their home between October 21st, 2022 at 3:00 PM and October 22nd, 2022 at 5:30 PM, while they were absent. They explain that they and their companion had left the house around 4:00 PM, and their son had left around 3:00 PM. They had left the windows open, as they planned to return later in the afternoon. However, they did not have time to return, as they spent the night at their companion's home in Chateauroux. The next day, they returned to their home at 5:30 PM and discovered that the front door had been left open and the house had been ransacked. They immediately called the authorities and conducted a search of the house to determine what had been stolen. They found that an envelope containing 300 euros in cash that their parents had given them for their birthday had been taken from their bedroom, and a box containing three pieces of gold and a silver gourmette, which had their birthdate and astrological sign engraved on it, had been taken from their son's bedroom. The box had been found open on the bed, but the items were not there. The person is the owner of the house and states that it was not equipped with an alarm or video surveillance system. They did not notice any suspicious individuals or vehicles before or after the burglary, and they did not see anyone making any reconnaissance near their home. They did not notice any signs or traces that could have been left by the burglars. This is the first time they have been burgled, and they are insured through Generali. They state that they did not move any objects in the house, except for the box containing the gold and silver gourmette and the envelope with the cash. The total estimated damage is around 1,200-1,300 euros. The person does not have any photos of the stolen items and does not have any other information to add. They are filing a complaint.

Table 14: Example of abstractive summary #1 – Generated with the base Mistral 7B model, and the prompt "Résume le texte suivant."

Tables 15 and 16 show much better quality summaries, generated respectively by the base Llama-13B model with a more complex prompt, and with Llama-2 70B. The important elements are mentioned, and the text is easy to read and well-structured.

Le texte est un procès-verbal d'audition. Il s'agit d'une plainte pour un cambriolage qui s'est produit à la maison d'un individu entre le 21/10/2022 à 15h00 et le 22/10/2022 à 17h30, alors que le propriétaire était absent. Le propriétaire a découvert le cambriolage en ouvrant sa porte d'entrée et a constaté que les meubles de l'entrée étaient tous ouverts et avaient été fouillés. Il a découvert que l'enveloppe qui contenait les 300€ en liquide que ses parents lui avaient offerts pour son anniversaire avait disparu, et que la fenêtre de la chambre de Charles était ouverte. Le propriétaire a appelé les services directement et a fait le tour de la maison avec sa compagne pour voir ce qui avait été fouillé et ce qui avait pu être dérobé. Il a constaté que la gourmette en or et une boite en métal contenant trois pièces en or et une gourmette en or avaient disparu. Le propriétaire a déposé plainte pour les faits.

Table 15: Example of abstractive summary #2 – Generated with the base Llama-2 13B model, and the prompt "Le texte suivant est un procès-verbal d'audition. Tu dois en écrire une version synthétique, qui résume les informations factuelles principales en 2-3 phrases. Tu dois parler en français, à la 3ème personne, dans un style simple et facilement compréhensible, et sans inclure d'éléments extérieurs."

Le propriétaire d'une maison à Chateauroux a déposé plainte pour cambriolage après avoir découvert que son domicile avait été fouillé et que plusieurs objets avaient été volés, notamment 3 pièces en or, une gourmette en or et 300€ en liquide. Le propriétaire et sa compagne étaient absents lors des faits, ayant laissé la maison pour une soirée chez la compagne. Le cambriolage a été commis entre 15h00 et 17h30. Le propriétaire a rapporté que la maison était munie d'une armoire à coffre qui avait été fouillée, ainsi qu'une boîte en métal contenant les pièces en or et la gourmette, qui avait été ouverte et vide. Il a également signalé que les meubles de l'entrée avaient été fouillés et que les lumières étaient éteintes dans les chambres. Le propriétaire est assuré chez Générali, mais il n'a pas le numéro de contrat. Il a estimé le montant du préjudice à environ 1200/1300€. La police a été saisie pour enquêter sur les faits.

Table 16: Example of abstractive summary #3 – Generated with the base Llama-2 70B model, and the prompt "Résume le texte suivant."

Finally, Table 17 shows a summary generated by our fine-tuned Mistral 7B model. With a quite small model, the summary is concise, contains the important information, and captures the style often used in such audition reports.

Le ou les auteurs ont cambriolé la maison de la victime entre le 21/10/2022 à 15h00 et le 22/10/2022 à 17h30. Ils ont fouillé les meubles de l'entrée et les chambres à coucher, ouvert la fenêtre de la chambre de Charles et ont dérobé 3 pièces en or (valeur de 300€), une gourmette en or (valeur de 100€), 300€ en liquide, pour une valeur totale de 1200/1300€. La victime est assurée chez Générali mais ne dispose pas du numéro de contrat. La maison est de plain pied, fait environ 150 mètres carré et est composée de cuisine ouverte avec salon/salle à manger, deux buanderies/cellier, trois chambres dont une en mezzanine avec une salle de jeu, salle de bain avec WC, autre WC séparé, garage et sas/veranda entre la cuisine et l'une des buanderies pour accéder à la terrasse.

Table 17: Example of abstractive summary #4 – Generated with the fine-tuned Mistral 7B model.

Chapter 5

Discussions

This Chapter addresses several various points concerning this thesis. First, Section 5.1 discusses several ethics-related topics, such as the management of bias and toxicity, or the question of hallucinations for LLMs. A sustainability study is presented in Section 5.2, and some future avenues of work are finally mentioned in Section 5.3.

5.1 Ethics

5.1.1 Bias

Working with LLMs, as with a lot of AI models, raises strong ethical issues. One of them is the presence of bias: as such models were trained on real-world data, they have naturally learned in some way the biases that existed in the training data. Such biases can be related to ethnicity, genre, sexual orientation etc. Most open-source and proprietary models include bias evaluation, but such risks still remain and must be considered. Some models, such as Llama-2 [26], after running such bias tests, have put emphasis during their instruction-tuning on bias reduction, making their instruction-tuned model a quality choice. However, bias cannot fully be removed from these models, and their use must come with an important sensitization on the presence of bias. Only a critical use of such tools will benefit from them: when making these tools available to agents on the field, it is essential to present their results as proposals, that absolutely need a human review, to remove the possibly remaining bias.

5.1.2 Toxicity

Toxicity, when dealing with LLMs and generative AI models, is often addressed simultaneously with bias. These models have been fine-tuned, to avoid generating harmful content, and general instructions are almost always given to proprietary chatbot such as ChatGPT (in a *system prompt*), to prevent them from answering the user about possibly harmful topics (weapons, violence, aggressions etc.). However, in our case, toxicity is clearly separated from bias, since the data we study is by nature toxic. It includes all kinds of complaints, testimonials or confessions, and especially toxic ones such as descriptions of sexual aggression, of physical or moral violence, of discrimination etc. And this information is precisely often the most important one, that we aim at detecting and summarizing: it is therefore impossible to use a system prompt that prevents the model from generating such content. However, it still is important that it does not generate additional toxicity, but this, as unwanted bias, must be controlled by the user themselves, because there is, to this date, no guarantee when using generative language models.

5.1.3 Hallucinations

While a large part of this project has been about detecting hallucinations and evaluating factuality of generated summaries, hallucinations remain an intrinsic limit of LLMs. To this day, no automatic solution has been developed to detect these with a satisfying precision. The only way to mitigate their negative possible consequences is to detect them manually, with a rigorous and critical review of the generated text. The use of LLMs, even more so in a judicial context, must be reserved for agents who have been sensitized to hallucinations risks. To give an example of hallucination that is subtle and difficult to detect, Llama-2 70B once deduced the age of a victim based on the context (age of friends, grade, etc.) when the original report never mentioned their age.

5.1.4 Data privacy and GDPR

As explained in Section 3.2, the data we worked on in this thesis is sensitive and personal data. Pseudonymization of source texts has been performed on raw audition reports, meaning that all personal information has been replaced by fake ones. However, a full anonymization is not possible: it is always possible, if one has access to a pseudonymized audition report, to cross information, with news articles for instance, to deduce the persons involved. For this reason, models trained on such data, as there is always a risk of training data leakage, must be reserved to a controlled, secure and internal use.

5.2 Sustainability

The main sustainability issue for this project comes from the training and the inference use of LLMs. First, such models were intensively pre-trained using energy-consuming GPUs. To give an order of magnitude, Llama-2 70B model's pre-training and instruction-tuning emitted 291.42 tCO_2eq [26]. Besides, inference use of such models, along with possible fine-tuning, requires again a lot of energy-consuming computations. Of course, GPUs use electricity to work, which is why their greenhouse gas (GHG) emissions hugely depend on the energy mix of the country in which they operate. In France, the electricity generation has low GHG emissions, in contrast with the United States for instance, as shown on Table 18.

		France	United States	gCO_2eq/kWh
Low GHG emissions	Wind	8.19%	10.13%	11
	Solar	4.26%	4.78%	44.5
	Bioenergy	2.13%	1.21%	230
	Hydro	9.83%	5.80%	24
	Nuclear	63.29%	18.00%	12
	Other renewable	0.12%	0.42%	/
	Total	**87.82%**	**40.34%**	/
High GHG	Gas	9.18%	39.35%	490
	Coal	0.94%	19.40%	820
	Other fossil	2.06%	0.90%	/
	Total	**12.18%**	**59.65%**	/

Table 18: Electricity generation sources in 2022 – Statistics by country from [67], emissions by sources from [68].

In France, in 2022, considering the weighted average of the electricity emissions presented in Table 18, electricity emits 71.9 gCO_2eq/kWh. Besides, according to the constructor Nvidia [69], an A100 GPU consumes at most 400W. Combining these two elements, one finds that using an A100 GPU emits 28.78 gCO_2eq per hour of use. In order to check order of magnitudes, we used the statistics given by [26]: the pre-training of Llama-2 70B used 3.3 million hours of A100 GPU, for a global emission of 291.42 tCO_2eq, which corresponds to 0.09 gCO_2eq per hour of use. Our estimation is clearly higher than this one, but their estimation does not come with an explanation. Although this is still an approximation, we can consider that our estimation is an upper bound of GPU-related greenhouse gas emissions.

Parameter-efficient fine-tuning is computationally quite efficient: each epoch (on around 1000 samples) requires only around 20 minutes of training. Therefore, this whole project (including fine-tuning, models and methods evaluation, testing, etc.) required between a few dozen and a few hundred hours of GPU usage, leading to a global carbon footprint of at most 4 or 5 $kgCO_2eq$.

5.3 Future work

Several points remain to investigate, among which:

1. Explore other fine-tuning techniques, namely Reinforcement Learning from Human Feedback (RLHF) [30], or from Artificial Intelligence Feedback (RLAIF) [70];
2. Use the Question-Generation and Question-Answering capabilities of LLMs to develop an LLM-QA-Metric;
3. In the logical continuation of the factuality evaluation, develop methods to concretely increase factuality and reduce hallucinations, with for instance techniques such as, or inspired

by, Chain-of-Verification [71];

4. Take advantage of the efficiency of LLMs on English language, by translating the source texts in English, generating an English text, and translating back in French;

5. Extrapolate the techniques developed and tested in this project to multi-document summarization, which is of great interest in the case of audition reports.

Chapter 6

Conclusion

To begin with, this thesis focused on summarization evaluation, and in particular, on hallucination detection, which is a central issue when using generative models for judicial purposes. We proposed a light variation of SelfCheckGPT, which has a strong correlation with human judgment (0.743), compared to the wide-spread BARTScore (0.542). Our adaptation of a Question-Answering-based metric, using French models, surprisingly underperforms compared to these (0.335).

In a second time, since most pre-trained language models have restrictive size constraints, we adopted a hybrid approach for automatic summarization of French judicial data, with a first extractive pass. In this context, a greedy approach to optimize the BARTScore of the extracted summary was proposed. Experiments have shown that this approach (overlap rate: 0.190, semantic similarity: 0.513) clearly outperforms classic methods such as the base TextRank algorithm (overlap rate: 0.172, semantic similarity: 0.506). We also proposed an improvement of the TextRank algorithm with a threshold mechanism, leading to a non-negligible improvement (overlap rate: 0.180, semantic similarity: 0.513).

The second and abstractive step of this two-step summarization process was then studied. Several pre-trained, general-purpose and multilingual models (Llama-2, Mistral and Mixtral) were objectively compared on a summarization dataset of judicial procedures from the French police. When prompted quite basically, the performances of these models are directly related to their size: Llama-2 7B struggles to adapt to uncommon data (overlap rate: 0.083, BARTScore: -3.099), while Llama-2 13B (overlap rate: 0.159, BARTScore: -2.718) and Llama-2 70B (overlap rate: 0.191, BARTScore: -2.479) have proven quite versatile and efficient. However, when prompted with more details and context, small models such as Mistral 7B (overlap rate: 0.191, BARTScore: -1.824) and Llama-2 13B (overlap rate: 0.194, BARTScore: -2.691) reach high performances, becoming better or similar to Llama-2 70B or Mixtral 8x7B, on all evaluation metrics, while being much lighter. An adaptation of few-shot prompting with retrieval-augmented generation was proposed, but offers very little improvement in terms of content, while slightly increasing the hallucination risk.

Finally, because running huge models is computationally costly, and to reduce the need to empirically adapt prompts, parameter-efficient fine-tuning of Mistral 7B, with LoRA technique, was studied. After about ten epochs, with a small dataset of around 1500 samples, the fine-tuned model reaches the performances of much larger models (overlap rate: 0.185, BARTScore: -2.060). Qualitatively, this fine-tuned model generates summaries with a style and conciseness more faithful to what is expected.

Bibliography

[1] Tanya Goyal, Junyi Jessy Li, and Greg Durrett. **News Summarization and Evaluation in the Era of GPT-3**. *arXiv preprint arXiv:2209.12356*, 2022.

[2] Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, Eduardo Pontes Reis, Anna Seehofnerova, et al. **Clinical Text Summarization: Adapting Large Language Models can Outperform Human Experts**. *Research Square*, 2023.

[3] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. **A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions**. *arXiv preprint arXiv:2311.05232*, 2023.

[4] BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. **Bloom: A 176b-parameter Open-access Multilingual Language Model**. *arXiv preprint arXiv:2211.05100*, 2022.

[5] Fabian Retkowski. **The Current State of Summarization**. *arXiv preprint arXiv:2305.04853*, 2023.

[6] Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. **SummEval: Re-evaluating Summarization Evaluation**. *Transactions of the Association for Computational Linguistics*, 9:391–409, 2021.

[7] Chin-Yew Lin. **ROUGE: A Package for Automatic Evaluation of Summaries**. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics, 2004.

[8] Maja Popović. **chrF: Character n-gram F-Score for Automatic MT Evaluation**. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395. Association for Computational Linguistics, 2015.

[9] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. **BERTScore: Evaluating Text Generation with BERT**. *arXiv preprint arXiv:1904.09675*, 2019.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[11] Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. **MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance**. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China, November 2019. Association for Computational Linguistics.

[12] Weizhe Yuan, Graham Neubig, and Pengfei Liu. **BARTScore: Evaluating Generated Text as Text Generation**. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc., 2021.

[13] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. **BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension**. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[14] Daniel Deutsch, Tania Bedrax-Weiss, and Dan Roth. **Towards Question-Answering as an Automatic Metric for Evaluating the Content Quality of a Summary**. *Transactions of the Association for Computational Linguistics*, 9:774–789, 2021.

[15] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, Alex Wang, and Patrick Gallinari. **QuestEval: Summarization Asks for Fact-based Evaluation**. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6594–6604, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[16] Potsawee Manakul, Adian Liusie, and Mark Gales. **SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models**. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore, December 2023. Association for Computational Linguistics.

[17] Rada Mihalcea and Paul Tarau. **TextRank: Bringing Order into Text**. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411. Association for Computational Linguistics, 2004.

[18] Wikipedia. **PageRank**, 2023. [Online, Accessed 29 December 2023].

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. **Efficient Estimation of Word Representations in Vector Space**. *arXiv preprint arXiv:1301.3781*, 2013.

[20] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. **Enriching Word Vectors with Subword Information**. *Transactions of the Association for Computational Linguistics*, pages 135–146, 2017.

[21] Sepp Hochreiter and Jürgen Schmidhuber. **Long Short-Term Memory**. *Neural Computation*, 9(8):1735–1780, 11 1997.

[22] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. **Learning Phrase Representations using RNN Encoder-decoder for Statistical Machine Translation**. *arXiv preprint arXiv:1406.1078*, 2014.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. **Attention is All you Need**. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[24] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. **Language Models are Few-shot Learners**. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[25] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. **Llama: Open and Efficient Foundation Language Models**. *arXiv preprint arXiv:2302.13971*, 2023.

[26] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. **Llama 2: Open foundation and Fine-tuned Chat Models**. *arXiv preprint arXiv:2307.09288*, 2023.

[27] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. **Mistral 7B**. *arXiv preprint arXiv:2310.06825*, 2023.

[28] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. **Mixtral of Experts**. *arXiv preprint arXiv:2401.04088*, 2024.

[29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. **Exploring the Limits of Transfer Learning with a Unified Text-to-text Transformer**. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[30] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. **Fine-tuning Language Models from Human Preferences**. *arXiv preprint arXiv:1909.08593*, 2019.

[31] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. **GPTQ: Accurate Post-training Quantization for Generative Pre-trained Transformers**. *arXiv preprint arXiv:2210.17323*, 2022.

[32] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. **Promptbreeder: Self-referential Self-improvement via Prompt Evolution**. *arXiv preprint arXiv:2309.16797*, 2023.

[33] Diederik P Kingma and Jimmy Ba. **Adam: A Method for Stochastic Optimization**. *arXiv preprint arXiv:1412.6980*, 2014.

[34] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. **LoRA: Low-Rank Adaptation of Large Language Models**. *arXiv preprint arXiv:2106.09685*, 2021.

[35] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. **SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents**. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[36] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. **Ranking Sentences for Extractive Summarization with Reinforcement Learning**. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[37] Bui Thi Mai Anh, Nguyen Tra My, and Nguyen Thi Thu Trang. **Enhanced Genetic Algorithm for Single Document Extractive Summarization**. In *Proceedings of the 10th International Symposium on Information and Communication Technology*, page 370–376. Association for Computing Machinery, 2019.

[38] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. **Training Language Models to Follow Instructions with Human Feedback**. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[39] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. **PaLM: Scaling Language Modeling with Pathways**. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[40] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. **CamemBERT: a Tasty French Language Model**. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online, July 2020. Association for Computational Linguistics.

[41] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. **Flaubert: Unsupervised Language Model Pre-training for French**. *arXiv preprint arXiv:1912.05372*, 2019.

[42] Moussa Kamal Eddine, Antoine Tixier, and Michalis Vazirgiannis. **BARThez: a Skilled Pretrained French Sequence-to-Sequence Model**. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9369–9390, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[43] Manuel Faysse, Patrick Fernandes, Nuno Guerreiro, António Loison, Duarte Alves, Caio Corro, Nicolas Boizard, João Alves, Ricardo Rei, Pedro Martins, et al. **CroissantLLM: A Truly Bilingual French-English Language Model**. *arXiv preprint arXiv:2402.00786*, 2024.

[44] Joseph Gesnouin, Yannis Tannier, Christophe Gomes Da Silva, Hatim Tapory, Camille Brier, Hugo Simon, Raphael Rozenberg, Hermann Woehrel, Mehdi El Yakaabi, Thomas Binder, et al. **LLaMandement: Large Language Models for Summarization of French Legislative Proposals**. *arXiv preprint arXiv:2401.16182*, 2024.

[45] Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, Osama Mohammed Afzal, Samta Kamboj, Onkar Pandit, Rahul Pal, et al. **Jais and Jais-chat: Arabic-centric Foundation and Instruction-tuned Open Generative Large Language Models**. *arXiv preprint arXiv:2308.16149*, 2023.

[46] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. **GLM-130B: An Open Bilingual Pre-trained Model**. *arXiv preprint arXiv:2210.02414*, 2022.

[47] Zhonghao Wang. Mediagpt: A large language model target chinese media. *arXiv preprint arXiv:2307.10930*, 2023.

[48] Flavia Achena, David Preti, Davide Venditti, Leonardo Ranaldi, Cristina Giannone, Fabio Massimo Zanzotto, Andrea Favalli, and Raniero Romagnoli. **Legal Summarization: to Each Court its Own Model**. In Federico Boschetti, Gianluca E. Lebani, Bernardo Magnini, and Nicole Novielli, editors, *Proceedings of the 9th Italian Conference on Computational Linguistics, Venice, Italy, November 30 - December 2, 2023*, volume 3596 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[49] Emmanuel Bauer, Dominik Stammbach, Nianlong Gu, and Elliott Ash. **Legal Extractive Summarization of US Court Opinions**. *arXiv preprint arXiv:2305.08428*, 2023.

[50] Duy-Hung Nguyen, Bao-Sinh Nguyen, Nguyen Viet Dung Nghiem, Dung Tien Le, Mim Amina Khatun, Minh-Tien Nguyen, and Hung Le. **Robust Deep Reinforcement Learning for Extractive Legal Summarization**. In *Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part VI 28*, pages 597–604. Springer, 2021.

[51] Abhay Shukla, Paheli Bhattacharya, Soham Poddar, Rajdeep Mukherjee, Kripabandhu Ghosh, Pawan Goyal, and Saptarshi Ghosh. **Legal Case Document Summarization: Extractive and Abstractive Methods and their Evaluation**. *arXiv preprint arXiv:2210.07544*, 2022.

[52] Yue Huang, Lijuan Sun, Chong Han, and Jian Guo. **A High-Precision Two-Stage Legal Judgment Summarization**. *Mathematics*, 11(6), 2023.

[53] Ayesha Sarwar, Seemab Latif, Rabia Irfan, Adnan Ul-Hasan, and Faisal Shafait. **Text Summarization from Judicial Records using Deep Neural Machines**. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–6, 2022.

[54] Diego de Vargas Feijo and Viviane P. Moreira. **Improving Abstractive Summarization of Legal Rulings Through Textual Entailment**. *Artificial Intelligence and Law*, 31(1):91–113, 2021.

[55] Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. **TorchMetrics - Measuring Reproducibility in PyTorch**, 2022.

[56] Martin d'Hoffschmidt, Wacim Belblidia, Quentin Heinrich, Tom Brendlé, and Maxime Vidal. **FQuAD: French Question Answering Dataset**. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1193–1208, Online, November 2020. Association for Computational Linguistics.

[57] Rachel Keraron, Guillaume Lancrenon, Mathilde Bras, Frédéric Allary, Gilles Moyse, Thomas Scialom, Edmundo-Pavel Soriano-Morales, and Jacopo Staiano. **Project PIAF: Building a Native French Question-Answering Dataset**. In *Language Resources and Evaluation Conference*, pages 5481–5490, 2020.

[58] Ali Kabbadj. **Something new in French Text Mining and Information Extraction (Universal Chatbot): Largest Q&A French training dataset**, 2018. [Online, Posted 11 November 2018].

[59] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. **Exploring Network Structure, Dynamics, and Function using NetworkX**. In *Proceedings of the 7th Python in Science Conference*, pages 11–15, 2008.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. **Scikit-learn: Machine Learning in Python**. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[61] Iskander Akhmetov, Rustam Mussabayev, and Alexander Gelbukh. **Reaching for Upper Bound ROUGE Score of Extractive Summarization Methods**. *PeerJ Computer Science*, 8, 2022. Publisher Copyright: © Copyright 2022 Akhmetov et al.

[62] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. **Transformers: State-of-the-Art Natural Language Processing**. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[63] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. **PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods**. `https://github.com/huggingface/peft`, 2022.

[64] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. **TRL: Transformer Reinforcement Learning**. `https://github.com/huggingface/trl`, 2020.

[65] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. **Retrieval-augmented Generation for Knowledge-intensive NLP Tasks**. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[66] Nils Reimers and Iryna Gurevych. **Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks**. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, November 2019.

[67] Ember. **Countries and Regions Electricity Generation**. `https://ember-climate.org/countries-and-regions/`. [Online, Accessed 30 December 2023].

[68] IPCC. **Table A.III.2 | Emissions of selected electricity supply technologies**. *Climate Change 2014: Mitigation of Climate Change: Working Group III Contribution to the IPCC Fifth Assessment Report*, page 1335, 2014.

[69] Nvidia. **Nvidia A100, Tensor Core GPU**. `https://www.nvidia.com/en-us/data-center/a100/`. [Online, Accessed 30 December 2023].

[70] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. **RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback**. *arXiv preprint arXiv:2309.00267*, 2023.

[71] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. **Chain-of-Verification Reduces Hallucination in Large Language Models**. *arXiv preprint arXiv:2309.11495*, 2023.

TRITA –