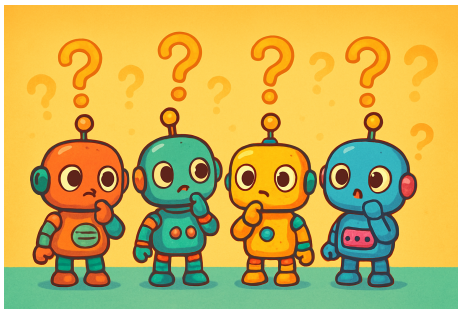
TÉCNICO LISBOA



TokenCP – Non-Exchangeable Conformal Prediction for Uncertainty Quantification

Rodolfo Jorge Antunes Amorim

Thesis to obtain the Master of Science Degree in

Eletrical and Computer Engineering

Supervisor(s): Doctor Chrysoula Zerva
Master Margarida Gerald Oliveira Moreira de Campos

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Dedicated to my grandparents.

Acknowledgments

Firstly, I want to express my gratitude towards Professor Chrysoula Zerva and citizen of the world Margarida Campos, for their patience and indispensable guidance throughout this thesis and valuable research insights, which I will definitely use in my future career.

Secondly, I would like to thank all of my friends that I made throughout these years, from my high school to both IST and TU Graz: Afonso, Anna, Alexandre Coelho, Alexandre Ferreira, Alexandre Baptista, Dinis, Diogo, Duarte, Gergő, Gonçalo, Elena, Joaquim, José, Lucas, Luka, Martim, Pedro Cordeiro, Pedro Gonçalves and Saúl. Without you I would not have had such good moments and insightful conversations, which have impacted the outcome of this thesis. Additionally, I want to thank my family, especially my younger brother Ricardo (who cannot read in english), for all of his sports betting advice that I did not use and my twin brother Rodrigo, which helped me immensely during these 5 years. Last but not least, I want to thank my girlfriend Inês, for her unwavering support and positive words which were cornerstone to this thesis. Without her this work would not be as good.

Finally, a heartfelt thank you to *Instituto de Telecomunicações* for allowing the use of one of its servers (thank you to Professor Marcos Treviso and Professor Chryssa for helping set up everything). This work was supported by the 'OptiGov' project, with ref. n. 2024.07385.IACDC (DOI: 10.54499/2024.07385.IACDC), fully funded by the 'Plano de Recuperação e Resiliência' (PRR) under the investment 'RE-C05-i08 - Ciência Mais Digital' (measure 'RE-C05-i08.m04'), framed within the financing agreement signed between the 'Estrutura de Missão Recuperar Portugal' (EMRP) and Fundação para a Ciência e a Tecnologia, I.P. (FCT) as an intermediary beneficiary.

Abstract

Large Language Models (LLMs) have seen widespread adoption due to strong performance across various language tasks, even though their errors can lead to serious consequences in high-risk scenarios. To rely on these models, it is essential to have a sound way to assess the model's confidence in each output, thus leading to the emergence of uncertainty quantification (UQ) methods. However, there is no established method for estimating confidence in LLMs, partly due to the lack of a universal evaluation procedure and the inherent complexity of UQ in text generation – poorly calibrated models and the complex nature of the task.

Conformal prediction (CP) is a popular UQ framework that offers statistical guarantees but remains underexplored in language modeling. We introduce a novel white-box UQ method for LLMs' text generation based on CP – TokenCP, which is model-agnostic and easy to implement. TokenCP creates a prediction set of tokens for each decoding stage at generation time, and their sizes are used as a proxy of the model's confidence. Furthermore, TokenCP can be combined with other state-of-the-art approaches to make sentence-level uncertainty measurements.

In the closed-book open-generation question-answering (QA) task, although TokenCP does not outperform other state-of-the-art solutions, it yields competitive performance across QA datasets and LLMs, exhibiting similar robustness with regard to model and dataset changes. Moreover, we show the potential for improving TokenCP, namely through the selection of relevant tokens in an answer.

Keywords: Uncertainty Quantification, Large Language Models, Conformal Prediction, Natural Language Processing

Resumo

Os Grandes Modelos de Linguagem (GMLs) têm sido amplamente adoptados devido ao seu bom desempenho em tarefas de linguagem natural, mesmo que possam errar em cenários de alto risco. Para se confiar nestes modelos, é essencial um método que avalie a sua confiança nos seus outputs – o que levou ao aparecimento de métodos de quantificação da incerteza (QI). Ainda não existe nenhum QI consagrado, devido à ausência de um procedimento de avaliação universal e à complexidade inerente da QI na geração de texto: modelos mal calibrados e a natureza complexa da tarefa.

Conformal Prediction (CP) é um procedimento de QI que oferece garantias estatísticas, mas permanece pouco explorado no contexto de geração de texto. Por conseguinte, apresentamos um novo método *white-box* de QI para a geração de texto baseado em CP — TokenCP, que é independente do modelo utilizado e fácil de implementar. O TokenCP cria um conjunto de *tokens* elegíveis para cada etapa de geração do GML. Os tamanhos destes conjuntos são usados como um *proxy* da confiança do modelo. Além disso, o TokenCP pode ser combinado com outras abordagens para dar medições de incerteza considerando todos os conjuntos.

Na tarefa *closed-book open-generation question-answering* (QA), embora o TokenCP não supere outras soluções existentes, este apresenta um desempenho competitivo em vários conjuntos de dados de QA e GLMs, exibindo robustez semelhante relativamente a alterações de modelo e de conjunto de dados. Adicionalmente, mostramos o possível melhoramento do TokenCP, através da seleção de *tokens* relevantes de uma resposta.

Palavras-chave: Quantificação de Incerteza, Grandes Modelos de Linguagem, *Conformal Prediction*, Processamento de Linguagem Natural

Contents

Acknowledgments vii
Abstract ix
Resumo xi
List of Tables xvii
List of Figures xix
Nomenclature xxi
Glossary xxiii

1 Introduction **1**
1.1 Motivation 1
1.2 Objectives and Deliverables 2
1.3 Challenges 3
1.3.1 Navigating Uncharted Territory 3
1.3.2 Hardware Constraints 3
1.4 Thesis Outline 3

2 Background **5**
2.1 Large Language Models 5
2.1.1 Overview 5
2.1.2 How does an LLM generate a response 5
2.1.3 Transformer Architecture 6
2.1.4 Decoding Stage 7
2.2 Uncertainty 7
2.2.1 What is Uncertainty? 8
2.2.2 How to measure uncertainty? 9
2.3 Conformal Prediction 10
2.3.1 Procedure 10
2.3.2 Non-conformity scoring functions 11
2.3.3 Going beyond exchangeable data 11
2.3.4 Evaluating Conformal Prediction 12

3 Related Work **13**
3.1 Evaluation of uncertainty quantification for LLM generations . . . 13
3.1.1 What are the underlying objectives? . . . 13
3.1.2 How to evaluate an uncertainty quantification method? . . . 14
3.2 Uncertainty Quantification methods . . . 16
3.2.1 Common baselines . . . 16
3.2.2 White box uncertainty quantification methods for LLMs . . . 17
3.3 Applying conformal prediction to LLM generations . . . 19
3.3.1 Non-Exchangeable Conformal Prediction . . . 19

4 Methodology **21**
4.1 TokenCP . . . 21
4.1.1 Motivation . . . 21
4.1.2 Algorithm . . . 21
4.1.3 Conformal prediction component . . . 22
4.1.4 Prediction set size analysis component . . . 24
4.1.5 Differences between the CP procedure in TokenCP and the standard approach . . . 26
4.2 Evaluating our approach . . . 27
4.2.1 Similarities with common evaluation pipeline . . . 27
4.2.2 Changes to the typical evaluation pipeline . . . 28
4.2.3 Model and dataset diversity . . . 28

5 Implementation **31**
5.1 Infrastructure . . . 31
5.1.1 Models . . . 31
5.1.2 Datasets . . . 32
5.1.3 Software . . . 33
5.1.4 Server . . . 33
5.2 Code Workflow . . . 33
5.2.1 Step 1: Data preparation . . . 34
5.2.2 Step 2: Calibration dataset creation . . . 34
5.2.3 Step 3: TokenCP version . . . 34

6 Results and discussion **37**
6.1 General setting . . . 37
6.2 Calibration dataset sizes . . . 37
6.3 *Top-k* values . . . 38
6.4 Subword approaches . . . 39
6.5 Non-conformity functions . . . 41
6.6 Final results . . . 43

6.7 Token Selection . . . 44
6.7.1 Impact of token selection on TokenCP . . . 46
6.8 UQ methods sensitivity to model or dataset changes . . . 46
6.9 Final comments . . . 47

7 The exploration behind TokenCP **49**
7.1 Calibration datasets . . . 49
7.2 How to use prediction sets for UQ . . . 50

8 Conclusions **53**
8.1 Contributions . . . 53
8.1.1 Insights drawn for UQ methods . . . 53
8.1.2 TokenCP . . . 54
8.2 Future Work . . . 54
8.2.1 Evaluating the CP procedure of TokenCP . . . 54
8.2.2 Token Selection . . . 55
8.2.3 Calibration datasets . . . 55
8.2.4 Further evaluation of TokenCP . . . 55
8.3 Final remarks . . . 56

Bibliography **57**

A Prompts **65**
A.1 Answer generation prompt . . . 65
A.2 Token selection prompt . . . 65

List of Tables

5.1 LLMs description. DPO stands for Direct Preference Optimization (Rafailov et al. [61]) and RLHF stands for Reinforcement Learning From Human Feedback (Ouyang et al. [62]). 32
5.2 Examples of datasets' datapoints 32
5.3 Percentage of correct answers by each model for a batch of 200 questions (test set). . . . 32
5.4 Dataset statistics (test set) 33

6.1 Type of tokenizer used in each model and the token vocabulary size. 40
6.2 Percentage of merges in the calibration dataset, test set and for the correct and wrong answers in the test set. 40
6.3 AUROC scores comparison between UQ methods. Scores in bold represent the highest score for a certain model and dataset. Percentages in red or green indicate the relative difference between a score and the score in bold. 44
6.4 AUROC comparisons between UQ methods considering the use of token selection. UQ methods with **+ Selection** indicate the use of token selection. The percentages show the relative difference between the AUROC score of a UQ method and the same UQ method **+ Selection**. 45
6.5 Comparison of AUROC scores for TokenCP in regard to the use of token selection. The percentages show the relative difference to the corresponding *TokenCP version* with no selection. 46
6.6 AUROC variability per UQ method measured using Coefficient of Variation (CV). Lower values indicate more consistent performance across models. Percentages in red or green show the relative difference between the CV of a UQ method using selection relative to its counterpart not using. Percentages in green indicate that the UQ method became more stable in comparison to its counterpart. 47

List of Figures

2.1 Transformer architecture 6
2.2 Decoding phase 7
2.3 Suggested Taxonomy 8

4.1 Suggested UQ method 22
4.2 Detailed conformal prediction procedure 23
4.3 Calibration dataset creation process 24
4.4 Combination of TokenCP with another UQ token-level method 26
4.5 Evaluation pipeline 27

5.1 Step 1 workflow 34
5.2 Step 1-2 workflow 35
5.3 Step 1-2-3 workflow 35

6.1 AUROC scores for different calibration dataset size 38
6.2 AUROC scores for different *top-k* values 39
6.3 AUROC scores for different token types 41
6.4 Step 1-2 workflow 42
6.5 AUROC scores for different non-conformity functions. 43

7.1 Histograms for the summary statistics, containing two classes: Correct answers and Wrong answers. The results plotted are for the Llama-3.1-8B model and the dataset TriviaQA. 50

Nomenclature

Roman symbols

$\hat{q}$	Quantile of the non-conformity scores in the calibration dataset
k	Sequence of calibration samples followed by a test sample
n	Number of calibration data points
$s(x, y)$	Non-conformity function
w	Weights
y	Predicted label
c	Prediction set size
g	Score from TokenCP
N	Total number of tokens in a response
r	Generated response
T	Set of all tokens from the token vocabulary
x	Input prompt
z	Token

Glossary

APS	Adaptive Prediction Sets
AUROC	Area Under the Receiver Operator Curve
BERT	Bidirectional Encoder Representations from Transformers
CP	Conformal Prediction
DPO	Direct Preference Optimization
GPU	Graphics Processing Unit
LAC	Least Ambiguous set-valued Classifiers
LLM	Large Language Model
LN-PE	Length Normalized Predictive Entropy
MARS	Meaning-Aware Response Scoring
MPS	Maximum Sequence Probability
NLG	Natural Language Generation
NLP	Natural Language Processing
PRR	Prediction Rejection Ratio
QA	Closed-book open-generation question-answering task
RHLF	Reinforcement Learning from Human Feedback
UQ	Uncertainty Quantification

Chapter 1

Introduction

This thesis presents the study of applying conformal prediction (CP) for uncertainty quantification (UQ) in natural language generation (NLG). It introduces TokenCP, a novel UQ method based on CP. It first introduces the theoretical foundations, reviews related work, describes the proposed methodology, and concludes with an evaluation of TokenCP.

In this chapter, we address the motivation behind this work, the objectives of what we propose to create, the organization of this work, and the challenges we faced.

1.1 Motivation

Large Language Models (LLMs) have become ubiquitous nowadays due to their considerable performance over multiple natural language processing (NLP) tasks, such as text summarization and machine translation, Qin et al. [1]. Additionally, these models are being used in high-risk scenarios, such as fraud/scam detection (Huang and Wang [2]) and medical diagnosis (Panagoulias et al. [3]), in which mistakes can lead to terrible consequences. While LLMs have improved on different NLP tasks in recent years, they are still prone to making mistakes. This makes a notion of the model's uncertainty about its response, crucial to avoid wrong decisions.

Obtaining an uncertainty measure from an LLM is a challenging task. These models are often poorly calibrated, meaning that the probabilities they assign to outputs do not reliably reflect the true likelihood of correctness, as mentioned by Desai and Durrett [4], Shen et al. [5] and Zhu et al. [6]. In other words, an LLM may express high confidence even when its output is incorrect. This task is further complicated by some models having undisclosed internal parameters and NLG tasks having specific and complex structures.

As a result, UQ methods are heuristic-based methods that can either be white-box methods, requiring the internal parameters of an LLM, or black-box methods, not needing any parameters from an LLM, as pointed out by Krishnan et al. [7]. Currently, there is no standard approach to evaluate UQ methods due to varying definitions of uncertainty across authors, different structures in text generation tasks and overlapping evaluation metrics. Accordingly, there is no established solution, since it is not only

hard to compare such methods but also that none has significantly outperformed other options (for any considered evaluation process).

Lately, conformal prediction procedures – based on the conformal prediction framework introduced by Vovk et al. [8], have been gaining attention in the context of natural language processing (NLP) (Campos et al. [9]). Briefly, a conformal prediction procedure enables the creation of prediction sets which contain plausible outputs (with theoretical guarantees on containing the correct output) for each input-to-output mapping (with the possibility of being applied to machine learning models). It offers multiple advantages, including ease of implementation and high efficiency, all the while being distribution-free and model-agnostic, as mentioned by Angelopoulos and Bates [10]. More importantly, it can be used to provide uncertainty measurements based on the outputs of machine learning models.

Considering all that was said before, we propose a white-box UQ method called **TokenCP** that can be applied to the response generation phase of LLMs. This conformal prediction based process produces proxies for uncertainty measurements for each token generated by an LLM. As a result, when TokenCP is combined with other state-of-the-art UQ methods, we can transform these token uncertainty proxy measurements into a single uncertainty measurement for the whole generated sentence. This overall measure is more informative than unitary measurements, since it better captures the relation between tokens in an answer.

Lastly, the code for this work is open-source and available for public use via an open-access repository [1].

1.2 Objectives and Deliverables

The main objective of this thesis was to use a conformal prediction procedure for uncertainty quantification in the context of NLG. After some exploration and experimenting, we outlined TokenCP (discussed in Chapter 4), a UQ method based on the split conformal prediction procedure.

For the implementation of TokenCP (specifically the combination of TokenCP with other UQ methods), we defined a set of desiderata for the implementation, based on its applicability in real world scenarios:

- **Be easily integrated into a wide range of LLMs.** Due to the large spectrum of LLMs that are possible to use, it is important to make sure that our proposed UQ method can be integrated regardless of differences among LLMs.
- **Have competitive performance with current state-of-the-art approaches.** The main driving point for integration of an UQ method is its performance. As such it is important to keep our proposed method on par with current state-of-the-art approaches.
- **Maintain consistent performance across varying datasets and models.** Both models and datasets can differ vastly from one another. As such it is important to create a robust UQ method that can maintain its performance even when the model or dataset being used changes.

[1] https://github.com/Dolfas/TokenCP

1.3 Challenges

In this section, we introduce the challenges faced during this research work, which had an impact on both the overall development and evaluation of TokenCP.

1.3.1 Navigating Uncharted Territory

The creation of TokenCP constituted novel and exploratory work, which made its development roadmap unclear from the beginning. The broad and exploratory roadmap led to the testing of many ideas, some of which did not ultimately contribute to the thesis. While this process did not bring direct results, it provided some insights that shaped both the overall implementation of TokenCP and its possible future work, as discussed in Chapter 7. Furthermore, the implementation of conformal prediction for natural language generation presented significant challenges due to the limited availability of prior research and reference code. This thesis addresses this gap by exploring the use of CP for uncertainty quantification – in the context of NLG, and by open-sourcing the used code.

1.3.2 Hardware Constraints

In this research work, we used three *RTX 2080 Ti - 12GB* GPUs which unfortunately could not support LLMs with more than 10B parameters (due to VRAM shortage). As a result, we were limited to using smaller LLMs, excluding the larger and more commercially prevalent models from the evaluation of TokenCP in Chapter 6. This creates a gap in TokenCP evaluation, as larger models behave differently from smaller models and constitute a more representative benchmark for real-world applications. Additionally, the limited hardware constrained both the experimentation and debugging time, as the used models took longer to generate answers.

1.4 Thesis Outline

In this section, we present the structure of the thesis and the topics discussed in each of the 8 chapters (including this one):

- **Background** (Chapter 2) - Core concepts of LLMs, uncertainty quantification and conformal prediction are introduced with the goal of providing readers with essential information for understanding the presented work.
- **Related Work** (Chapter 3) - Following the previous introduction, the relevant work to the thesis is discussed. It begins with the study of UQ methods, going from their evaluation pipeline to different proposed solutions, and it ends with a use case of conformal prediction for language generation.
- **Methodology** (Chapter 4) - Considering the previous description of the UQ research landscape, TokenCP and its evaluation pipeline are presented.

- **Implementation** (Chapter 5) - Having defined TokenCP, its implementation along with the hardware, software, and data used are described.
- **Results** (Chapter 6) - Throughout this chapter the performance of TokenCP is examined. Building from experiments designed to select the optimal parameters for this process, to comparing the use of TokenCP with other UQ methods. In the end, a possible addition to both TokenCP and other UQ methods is discussed.
- **The exploration behind TokenCP** (Chapter 7) - The experiments conducted on the components of TokenCP are presented, along with the insights gained from this exploration.
- **Conclusions** (Chapter 8) - In this final chapter the main conclusions of this thesis are given alongside the possible future work.

Chapter 2

Background

In this chapter we provide the theoretical background of the core concepts necessary to understand the presented work. We provide an introduction to: large language models, uncertainty quantification in the context of NLG, and the framework of conformal prediction.

2.1 Large Language Models

2.1.1 Overview

LLMs are systems that contain sophisticated neural networks that leverage deep learning techniques, particularly transformer architectures introduced by Vaswani et al. [11], to learn and understand the complex patterns and structures present in textual data, as defined by Hadi et al. [12]. These models can be used for a wide range of NLP tasks, such as: summarization, translation, and text generation.

When it comes to their training phase, large amounts of text data is needed as mentioned by Brown et al. [13]. Additionally powerful hardware, like GPUs, are required to make the training of such models feasible. These models demonstrate a state-of-the-art performance in a variety of NLP tasks, but they have a large number of trainable parameters, which constitutes a challenge for training and finetuning. Examples of commonly known LLMs are: Gemini [14], Llama [15] and the GPT4 model that powers ChatGPT [16].

2.1.2 How does an LLM generate a response

Given an input prompt, in other words the instructions given to an LLM, transformed into tokens denoted as $x = (x_1, x_2, ..., x_n)$, the LLM generates its response, $y = (y_1, y_2, ..., y_n)$, in an autoregressive manner. This means that the next token, in a sentence generated by an LLM, is always sampled from a distribution that is dependent on the previous tokens and prompt. Consequently, the next token is chosen by sampling from the following distribution:

$$y_j \sim p_\theta(\cdot|x, y_1, ..., y_{j-1}). \tag{2.1}$$

Where θ denotes the LLM parameters. There are many different sampling strategies to choose the next token. For example, one could use the *top-k* sampling method (Fan et al. [17]), where out of the token vocabulary only *k* tokens are selected to take a random sample from (in other words choosing the next token only out of these *k* tokens).

This process occurs until we have reached the *end-of-sequence* token, hence ending the response generation. Another possible way to understand this mechanism is that at each token sampling step, the logits are computed. The logits are values that the LLM assigns to each token in its vocabulary as a way to reflect their likelihood of being the next token. Then, usually, the softmax function is used to put these values between 0 and 1, so that we have a ”probability” of each token in the LLM vocabulary being sampled as the next token. Having the softmax of the logits, we obtain a distribution like the one in equation 2.1 from which we can sample the next token.

2.1.3 Transformer Architecture

Most LLMs use transformer architectures, introduced by Vaswani et al. [11], as their building block. This architecture can be observed in Figure 2.1 (adapted from Vaswani et al. [11]).

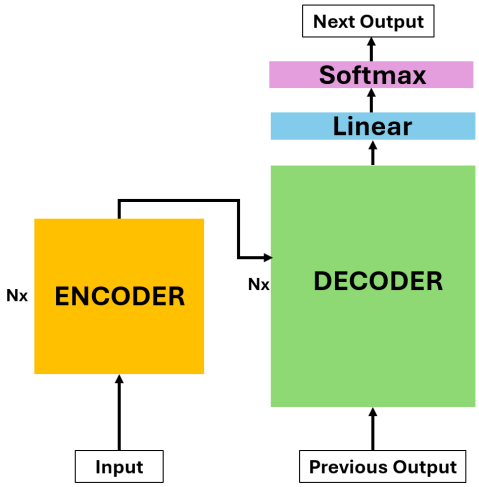


Figure 2.1: Transformer architecture

This architecture has an encoding stage and a decoding stage. In the encoding stage, represented by the left part of Figure 2.1, an encoder is used to transform the input into a meaningful representation, in the case of NLP it could be transforming words into tokens. In the decoding stage, represent by the rightmost part of Figure 2.1, explained in detail in the next section, the output is generated in an autoregressive manner, since the previous output is used to produce the next output (in this case an output could be a word in a sentence). The decoder integrates the encoded representation with the representation of past outputs to predict the subsequent token (e.g., a word in a sentence). At each stage, stacks of encoders/decoders are used, in the original paper a total of $N = 6$ for each.

The are two key innovations with this architecture. The first one is the *self-attention* mechanism, which allows the architecture to capture the relationship between different elements of a sequence in an efficient manner. In the case of a sentence, this *self-attention* mechanism helps capture the importance

of each word in relation to the other words present in the sentence. The second one is the *multi-head attention* layer, which allows to use multiple attention layers for the same representation in parallel, combining it all in the end. With each layer capturing diverse patterns and dependencies within the representation.

Besides what was said previously, these architectures allow for the parallelization of many of its computations and to process multiple input sequences at the same time, therefore improving the training and inference speed. Finally, a LLM has a stack of transformers, meaning that the next transformer receives the output from the previous transformer as input. This allows an LLM to extract sophisticated patterns in the data continuously.

2.1.4 Decoding Stage

As mentioned earlier, the transformer architecture has an encoding and a decoding stage. The decoding stage, can be observed in the following Figure:

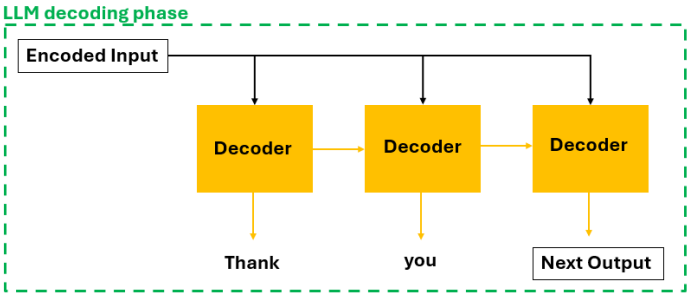


Figure 2.2: Decoding phase

As explained before, the output is generated auto regressively. The whole process begins by transforming the previous output, otherwise the start of sequence token, into a different representation, for example transforming words into tokens. Then this new representation is passed into the decoders, which use the attention mechanisms explained previously and the information from the encoding stage, to refine the new representation. Since a stack of decoders is used, the output of the previous decoder is the input of the following one. Once the new representation has passed through all of the decoders it is passed into a linear layer, that transforms this new representation into a vector of size of the vocabulary. Afterwards, the softmax layer converts the previous vector into a probability distribution over the vocabulary. Given this probability distribution, one can choose the next output.

2.2 Uncertainty

As mentioned before, in safety critical scenarios it is important to have an indication on the trustworthiness and reliability of LLM's answers. As such, measuring the uncertainty of these models outputs in these scenarios is crucial.

2.2.1 What is Uncertainty?

Uncertainty has become a more frequent topic of deep learning research over the years, yet there is still no clear consensus on what defines it well. One possible definition, is that uncertainty is a measure of how well the LLM knows what it knows and what does not know.

Commonly, uncertainty is divided into two types, with the first type being aleatoric uncertainty and the second type being epistemic uncertainty. On one hand, as per Hu et al. [18], the first type refers to the uncertainty inherent in data due to its randomness or noise and is considered to be irreducible, meaning that it cannot be eliminated even through model improvements or tuning. On the other hand, the second type refers to model uncertainty, that is caused by a lack of understanding about the model itself, (i.e. lack of understanding regarding its architecture and parameters). In contrast to the first type, this type is considered reducible. Besides this, it may not be explicit which type of uncertainty is dominant in a given NLP system, with these two types of uncertainty being able to interact in complex ways (hence making their combination hard to decouple).

While this dichotomy for uncertainty is quite popular in the deep learning community, Baan et al. [19] state that this dichotomy does in fact wrongly combine two aspects that are generally unrelated: the source of uncertainty and its reducibility. For this reason they propose a taxonomy, present in Figure 2.3, that still encapsulates the notion of uncertainty caused from data and model but that does not have a clear cut boundary between these two. Finally, this taxonomy uses a vertical axis which captures reducibility, and gives suggestions on changes the agent can make in order to move a source along the reducibility dimension.

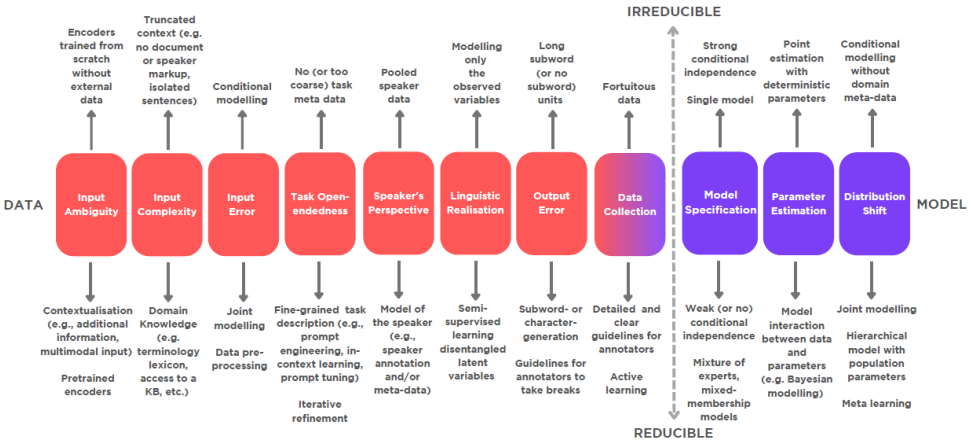


Figure 2.3: Suggested Taxonomy

When it comes to LLM uncertainty, there also two additional types. Token level uncertainty and sentence level uncertainty, described in Duan et al. [20]. Token level uncertainty is concerned with the uncertainty of the model at the token generation phase, which means that we can obtain this value as the LLM generates a response. Sentence level uncertainty is the uncertainty of the model when it comes to the whole response generated, accordingly we can only obtain this measure once the LLM has finished generating its response. Comparing these two, sentence level uncertainty tends to be more complete in comparison to token level uncertainty, as it better captures the complex relations between

tokens in an answer. However, token level uncertainty comes with the benefit of being obtained as the answer generates and not at the end, which can enable, for example, early stopping of the generated answer.

LLM uncertainty can originate from many sources, basing ouservels on Figure 2.3 we can point out to, for example, both input (prompt) ambiguity and complexity but also to model specification choices.

2.2.2 How to measure uncertainty?

When it comes to measuring uncertainty, we can look into the confidence of the model in its output or on the likelihood that the output is correct, with these two being separate tasks.

On the one hand, according to Liu et al. [21], **uncertainty estimation** for LLMs can be broadly defined as the task of predicting the quality of the generated response based on the input, with quality being: confidence, truthfulness or uncertainty. Consider an LLM and an input prompt $x = (x_1, ..., x_k)$ which leads to the following generated output $y = (y_1, ..., y_m)$. With this in mind we can define the task of uncertainty estimation mathematically, which has the goal of learning the function *g* which is used to predict the following score:

$$g(x, y) \approx \mathbb{E}[r(y, y_{true})|x, y]. \tag{2.2}$$

With r being a scoring function, $r(\cdot, \cdot) : Y \times Y \rightarrow [0, 1]$, that evaluates the quality of the response y given a true response y_{true}. The true response is usually chosen by factual truth, humans or domain experts and is assumed to follow a distribution conditioned on prompt x. An example of a scoring function r is the ROUGE-L metric (Lin [22]), which gives a score based on the overlap between an answer and a reference answer (thus giving a score that reflects truthfulness). The use of ROUGE-L can be found in the works: Duan et al. [20] and Vazhentsev et al. [23]. On the other hand, **uncertainty calibration**, is the task concerned with learning a function $g(x, y)$ that correctly conveys the probability of y, (the output of the LLM), being true.

When it comes to measuring LLM uncertainty, the methods proposed can be generally divided into two groups: white-box methods or black-box methods. White-box methods require the knowledge of the LLM internal parameters in order to measure the uncertainty of an LLM, whilst black-box methods do not require such information, depending only on the input prompt and the response generated by the LLM. Furthermore, black-box methods may require multiple runs – in order to obtain multiple answers for the same prompt or an external model. In contrast, white-box methods can be used on a single answer generation and without additional models. An example of a white-box method is analyzing token entropy, introduced by Malinin and Gales [24] – since it requires the softmax scores of the whole token vocabulary, at each token decoding step. Whereas, two examples of black-box methods include: using conformal prediction on a sentence level, as suggested by Ye et al. [25], or computing the semantic entropy of multiple answers, introduced by Kuhn et al. [26] – both approaches only need multiple answers to the same prompt for giving an uncertainty score.

2.3 Conformal Prediction

2.3.1 Procedure

Conformal prediction is a methodology used to create prediction sets/intervals for machine learning models, originally created by Vovk et al.[8], that are guaranteed to have the true label with a probability over the randomness of the data, defined by the user.

So as to obtain these prediction sets, the following procedure must be followed:

1. Identify a heuristic notion of uncertainty using a model (this can be for example *softmax* scores).
2. Obtain a set of *n* data points never seen during training. Accordingly, we can define a *calibration dataset*: $D_{cal} = \{(x_1, y_1), ..., (x_n, y_n)\}$ to be used. Each calibration data point consists of an input and a corresponding true label.
3. Define a non-conformity score function $s(x, y) \in \mathbb{R}$, that outputs scores that reflect the agreement between x and y, with larger scores showing less agreement between x and y. We then compute the non-conformity score for each calibration data point in D_{cal}.
4. Compute $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the non-conformity scores in the calibration dataset.
5. Use $\hat{q}$ to form prediction sets for test data points: $C(x_{test}) = \{y : s(x_{test}, y) \leq \hat{q}\}$

As mentioned before describing this procedure, we are guaranteed to have the true label with a probability defined by the user within each prediction set. This is also known as coverage guarantee, which is valid without distribution and model assumptions, given that all data points used as calibration data and test data are exchangeable. For this reason, the probability of having the true label inside a prediction set is expressed in the following inequality:

$$1 - \alpha \leq \mathbb{P}[y_{test} \in C_\alpha(x_{test})] \leq 1 - \alpha + \frac{1}{n+1}. \tag{2.3}$$

Where α is a value defined by the user and *n* the number of data points in the calibration set.

The coverage guarantee, states that the coverage will be of at least $1 - \alpha$ on average over the randomness in the data. Subsequently, there can be cases of prediction sets that do not have the true label with a probability of at least $1 - \alpha$, as long as on average we see this coverage for the prediction sets. The distribution of coverage has an analytical form, firstly introduced by Vovk [27], which can be seen in the following equation:

$$P(y_{test} \in C(x_{test}) | (x_i, y_i)_{i=1}^n) \sim \text{Beta}((n+1)(1-\alpha), (n+1)\alpha). \tag{2.4}$$

Using this distribution Angelopoulos and Bates [10] computed the number of calibration points needed to achieve a coverage of $(1 - \alpha \pm \epsilon)$, with a probability of $1 - \delta$, with δ a user specified value and ϵ a small number. Again, the average coverage is always at least $1 - \alpha$ and the parameter δ controls the tail probabilities of the coverage conditionally on the calibration data. Accordingly, the authors suggest

a calibration set of size: $n = 1000$ for most scenarios, as it substantially reduces coverage fluctuations, with larger calibration datasets offering little additional benefit.

2.3.2 Non-conformity scoring functions

As mentioned in Section 2.3.1, a CP procedure requires the use of a non-conformity scoring function. There are many non-conformity functions one could use in conformal prediction, in this work we focus on two popular and simple non-conformity scoring functions:

- **Least Ambiguous set-valued Classifiers (LAC)**:

$$s(X, Y) = 1 - f(X)_Y \tag{2.5}$$

  This non-conformity scoring function is represented in equation 4.1, where $f(X)_Y$ represents the softmax score of the predicted label *Y* for input *X*. It has been shown that this score leads to prediction set sizes with the smallest average, as per Angelopoulos and Bates [10]. Nonetheless, it may also lead to the creation of small prediction sets for hard instances and large ones for easy instances (which must be the opposite as harder choices should lead to larger prediction sets and vice-versa), according to Ye et al. [25].

- **Adaptive Prediction Sets (APS)**:

$$s(X, Y) = \sum_{Y' \in \mathcal{Y}: f(X)_{Y'} \geq f(X)_Y} f(X)_{Y'}. \tag{2.6}$$

  APS, (shown in equation 4.2), is defined as the cumulative probability over classes (after sorting descendingly) necessary to reach the correct class. It addresses the limitation of LAC but suffers from, on average, larger prediction sets, as also mentioned by Ye et al. [25].

2.3.3 Going beyond exchangeable data

Data exchangeability can be an unrealistic assumption for many NLP applications, especially for language generation methods, as a response is generated in an autoregressive manner. Accordingly, extensions have been proposed for handling non-exchangeable data. One such extension was proposed by Barber et al. [28], which provides prediction guarantees without the exchangeability assumption, where:

$$P[y_{test} \in C_\alpha(x_{test})] \geq 1 - \alpha - \sum_{i=1}^{n} \tilde{w}_i d_{TV}(k, k^i). \tag{2.7}$$

Where $\tilde{w_i} := \frac{w_i}{1+\sum_{i=1}^{n} w_i}$ are weights (with $w_i \in [0, 1]$), and $d_{TV}(k, k^i)$ is the total-variation distance between the distributions of k and k^i. Where $k_i = (x_i, y_i)$ and k is a sequence of n calibration samples followed by a test sample, $k = (k_1, ..., k_n, k_{n+1})$. Finally k^i denotes k after swapping k_i with k_{n+1}. A high

weight w_i should be assigned to a data point k_i that is "trusted" more, in other words, that is believed to come from (nearly) the same distribution as the test point k_{n+1}.

When applying, for instance, conformal prediction to the token generation phase of LLMs (by creating a prediction set of possible next tokens), the procedure can run into data non-exchangeability of the calibration dataset – compromising its theoretical guarantees. Hence, making unclear the predicted coverage of the prediction sets. With, for example, the method proposed by Barber et al. [28], we could compute the predicted coverage of the prediction sets for the next possible tokens, by following the equation 2.7, something that seemed impossible at face value.

2.3.4 Evaluating Conformal Prediction

So as to evaluate a conformal prediction procedure, one must take into account its adaptivity and correctness, as proposed by Angelopoulos and Bates [10]. Adaptivity refers to the ability of the CP procedure to make prediction sets that adapt to the difficulty of the input, and correctness reflects the possibility of the CP procedure to produce prediction sets which can empirically approximate the expected theoretical coverage. Before describing the evaluation process we must introduce the terms: marginal and conditional coverage. Marginal coverage, can be defined in equation 2.3, where the probability is marginal (averaged) over the randomness in the calibration and test points. Alternatively, conditional coverage can be expressed in equation 2.8, where for every input in x_{test} we seek to return prediction sets with $1 - \alpha$ coverage.

$$P[y_{test} \in C(x_{test})|x_{test}] = 1 - \alpha. \tag{2.8}$$

Both terms are central for the evaluation procedures explained next:

- **Adaptivity:** On one hand, an adaptive conformal prediction procedure returns larger prediction sets for harder inputs and smaller prediction sets for easier inputs. There are many ways to assess the adaptiveness of a conformal prediction procedure (Angelopoulos and Bates [10] and Campos et al. [9]). Next we give two formal metrics that are based on the conditional coverage of a CP procedure: feature-stratified coverage metric (Angelopoulos and Bates [10]) and size-stratified coverage metric (Angelopoulos et al. [29]).

- **Correctness:** On the other hand, correctness tells if conformal prediction has been correctly implemented, by measuring its marginal coverage. This can be done by running the procedure over R trials with new calibration and validation sets (by randomly splitting the data), and then calculating the empirical coverage for each. Subsequently, the empirical coverage is averaged over the number of trials R, with values close to 1-α indicating a correct implementation of conformal prediction. Additionally, a correct implementation of conformal prediction also reflects the validity of the data exchangeability assumption, with bigger deviations between the average empirical coverage and $1 - \alpha$ showing misalignment of distributions in the data.

Chapter 3

Related Work

In this chapter, we present a review of the current literature on uncertainty quantification (UQ) methods for natural language generation (NLG). We begin by presenting the way to evaluate such methods, followed by an overview of commonly used baselines and state-of-the-art UQ methods, and ending with the review of an application of CP in NLG.

3.1 Evaluation of uncertainty quantification for LLM generations

One of the challenges of defining uncertainty methods for LLM generations is their evaluation. A typical approach is to focus on tasks with a target reference so that the output of the LLM can be classified as correct or wrong. Subsequently, one or more metrics are used to reflect the relation between uncertainty measurements and output correctness. In the following subsections the common evaluation pipeline is described in-depth, based on: Kuhn et al. [26], Aichberger et al. [30], Bakman et al. [31] and Duan et al. [20].

3.1.1 What are the underlying objectives?

Uncertainty quantification methods for LLMs have the purpose of reflecting a model's confidence in the generated outputs, as mentioned by Fadeeva et al. [32]. However, creating such methods and evaluating them is hard to do in practice, due to:

- LLMs being different in many aspects, including architecture and training.
- No clear definition of what uncertainty is, leading many works to interpret uncertainty in alternative ways.
- Diversity among NLP tasks, consequently making the creation of a general UQ evaluation procedure challenging – since, for example, not all tasks have a reference/correct generation.

These challenges have made the current research landscape fragmented, with no standard processes for both scenarios (as mentioned by Shorinwa et al. [33]). As a result, next we present the

typical approaches in contemporary literature.

3.1.2 How to evaluate an uncertainty quantification method?

Even though there may not be an established evaluation procedure, we identified some common approaches that appear in the existing literature.

What task to choose?

The first step in building an UQ evaluation pipeline is to clearly define a task for the LLM. It is the task that provides the basis for the LLM to generate outputs, from which we can obtain model confidence scores – using UQ methods.

A common choice is the closed-book open-generation question and answering (QA) task, as exemplified by Duan et al. [20], Aichberger et al. [34] and Yaldiz et al. [35]. In this task, an LLM is given a question for which it generates an answer, without any additional information besides the question itself. For instance, for the question: “What is the capital of Kazakhstan?”, the model outputs an answer for it: “The capital of Kazakhstan is Astana.”. Each question has a designated golden/reference answer – correct response to the question, which in this case could be: ”Astana.”. The domains of the questions can differ from dataset to dataset, with some focusing, for example, on the medical domain and others on the sports domain. Furthermore, the length of the reference answers may vary (reflecting different complexity levels from the datasets), and the origin of the questions can also differ (ranging from random user queries to tailored questions made by experts). Correspondingly, commonly used datasets for this task include: TriviaQA [36], NaturalQA [37], CoQA [38] and MedQA [39], as used in the works from: Duan et al. [20], Yaldiz et al. [35] and Bakman et al. [31].

However, it is possible to use other tasks such as: machine translation (Fadeeva et al. [32]), summarization (Vazhentsev et al. [23] and Manakul et al. [40]) or hallucination detection (Fadeeva et al. [41] and Yadkori et al. [42]).

How to define the correctness of an answer?

Considering the optimal behavior of an UQ method, it is necessary to define the criteria for the classification of LLM responses as either correct or wrong. The criteria for such classification varies between different approaches. Some of the common metrics used for QA classification are:

1. **Similarity with reference answer:** In this scenario, we can use two metrics: ROUGE-L (Lin [22]) and BERTScore (Zhang et al. [43]), each receiving a generated answer and a reference answer. Subsequently, each computes a score between 0 and 1, that reflects how similar the two sentences are. Usually, scores greater than or equal to 0.5 are considered to reflect relevant similarity. Therefore, for scores greater than or equal to 0.5 generated answers are classified as correct (and incorrect otherwise). This criteria is present in the following research work: Duan et al. [20], Vazhentsev et al. [23], Fadeeva et al. [32] and Aichberger et al. [30]. Even though these

metrics are simple to use and implement, they are not very precise as it is possible to obtain high similarity scores (≥ 0.5) for answers that contradict each other.

2. **LLM as a judge:** In the research work from: Bakman et al. [31], Yaldiz et al. [35] and Vashurin et al. [44], an LLM is employed to work as an evaluator. This method provides the LLM with: the generated answer, the context/question and a reference answer. Then the LLM classifies the generated answer as either correct or wrong. Compared to the previous approach, using an LLM can lead to more precise classifications, however it comes with implementation issues – such as prompt design, and an increase in classification time.

How to assess the performance of an UQ method?

In this scenario, the generated answers can be classified as either correct or incorrect. Accordingly, the performance of an UQ method can be assessed by examining how its scores relate with the correctness of the LLM's answers. Ideally, a well performing UQ method would assign high uncertainty scores to incorrect answers and low uncertainty scores for correct answers. This line of thought is present in: Duan et al. [20], Bakman et al. [31] and Yaldiz et al. [35]. Next, we focus on two possible ways to study the relation between uncertainty measurements and response correctness. Other alternatives include: Prediction rejection ratio (Malinin et al. [45]) – suited for model selective prediction/abstention settings, and Precision Recall Area Under the Curve (Davis and Goadrich [46]) – which reflects the ability of correctly identifying the positive class.

One way to study the correlation between uncertainty measurements and response correctness, would be to compute the **Pearson Correlation**, as used in the work of: Xiao and Wang [47] and Zhang et al. [48]. Since the correctness of an answer must be given as a continuous value, one could compute the similarity of an answer with the golden answer – when the answer correctness is given as a binary value. A strong negative correlation would indicate that higher uncertainty scores are consistently associated with incorrect answers (i.e., answers that differ from the reference). This would demonstrate a good performance of the used UQ method. Even though it is easily interpretable and ready to implement, it comes with a few downsides. It requires all variables to be continuous, it is sensitive to outliers, and can give misleading correlations.

Another possibility would be to use the **Area Under the Receiver Operator Curve (AUROC)** metric (Hanley et al. [49]), as done for example by: Duan et al. [20], Bakman et al. [31] and Aichberger et al. [34]. This metrics outputs range from 0 to 1, with values closer to 1 indicating better performance. In this case, a high AUROC score would mean that correct responses were consistently assigned lower uncertainty measurements than incorrect ones. Additionally, a score of 0.5 would suggest that the uncertainty measurements are no better than random guessing when it comes to classifying the correctness of responses. In comparison to the previous metric, this one does not require all variables to be continuous, however a low positive class count can have an impact on the score given by it.

In summary, UQ methods can be evaluated by assessing how their uncertainty measurements relate to answer correctness, with two common approaches being the use of **Pearson correlation** and

AUROC. Each apporach has its strengths and limitations: **Pearson correlation** is easily interpretable, while **AUROC** offers an easier implementation.

3.2 Uncertainty Quantification methods

In this section, we present some common baselines and relevant state-of-the-art UQ methods for natural language generation.

3.2.1 Common baselines

Here we introduce some commonly used uncertainty quantification baselines, used for example by: Yaldiz et al. [35], Duan et al. [20] and Vashurin et al. [50]. The baselines are the following:

- **Maximum sequence probability (MSP)**, (Fadeeva et al. [32]), defined as:

$$\text{MSP}(r, x) = 1 - \prod_{i=1}^{N} p(z_i | r_{<i}, x), \tag{3.1}$$

  with *r* being the response generated, x the prompt, z_i the *i*-th token, $r_{<i}$ the response up to the i-th token and *N* the total number of tokens in response *r*. The equation 3.1 represents: 1 minus the probability of generating the response r given the prompt x, $p(r|x)$. One of the key issues of this uncertainty quantification baseline is that the value of $p(r|x)$ tends to decrease as the length of the answer increases.

- **Maximum token entropy (MTE)**, (Fomicheva et al. [51]), being:

$$H_i = -\sum_{z \in T} p(z | r_{<i}, x) \log(p(z | r_{<i}, x)), \tag{3.2}$$

$$\text{MTE}(r, x) = \max_i H_i, \tag{3.3}$$

  where H_i is the token entropy at the i-th token of response r. The entropy is computed over all the tokens z belonging to the token vocabulary T. This baseline returns the maximum of the entropy computed for all the tokens in the answer, thus outputting the score related to the moment were the model was the most uncertain about choosing a token – since this moment is the one where the token probabilities in the token vocabulary are the most uniform. A problem with this baseline is that it depends on the token vocabulary size (making it less robust across different models).

- **Length normalized predictive entropy (LN-PE)**, (Malinin and Gales [24]), which can be defined as:

$$\text{LN-PE}(r, x) = -\frac{1}{N} \sum_{i}^{N} \log(p(z_i | r_{<i}, x)). \tag{3.4}$$

  This uncertainty baseline corresponds to the negative log of length-normalized scoring (proposed

by Malinin and Gales [52]):

$$\prod_{i=1}^{N} p(z_i|r_{<i}, x)^{\frac{1}{N}}, \tag{3.5}$$

which assigns equal weights to each token in the generation where these weights are inversely proportional to the sequence length N. Even though it is not length dependent like the MSP baseline, it does not correspond to the negative log of $p(r|x)$.

3.2.2 White box uncertainty quantification methods for LLMs

We now present two state-of-the-art methods that are relevant for the presented work, namely because thay are also white-box and token-level approaches.

Shifting Attention to Relevance: Towards the Predictive Uncertainty Quantification of Free-Form Large Language Models

In the paper by Duan et al. [20], the authors introduced three new uncertainty quantification methods, of which we will focus on the approach that samples once: **TokenSAR**. This method is based on the idea that tokens are not all equally relevant when conveying the semantic meaning of an answer, and as such, not all tokens are equally relevant, then they should not be treated as equal in uncertainty measurements.

Suitably, the authors devised a way to assign different importance weights to tokens that reflect their semantic relevance in the answer. The authors propose the following procedure:

1. Define a function $g(\cdot, \cdot)$ that computes the semantic similarity between two sentences, and returns a value between 0 and 1. Values closer to 1 indicate a greater semantic similarity between the two sentences.

2. For a certain token z_i belonging to the answer string, compute the semantic similarity between the whole answer and the answer minus this token, so as to obtain the following score:

$$R_T(z_i, r, x) = 1 - |g(x \cup r, x \cup r \setminus \{z_i\}|, \tag{3.6}$$

where x is a prompt and r is the answer. Larger values of R_T indicate that removing a certain token leads to a bigger change in the semantic meaning of the answer, hence also indicating that the token at hand is semantically relevant for the answer.

3. After repeating the previous step for all the tokens, the weights assigned to each token are computed by normalizing each R_T score:

$$\tilde{R}_T(z_i, r, x) = \frac{R_T(z_i, r, x)}{\sum_i^N R_T(z_i, r, x)}, \tag{3.7}$$

with N being the total number of tokens in r.

Considering the varying token weights, an uncertainty measurement, for a given response r and prompt x, can be obtained using the **TokenSAR** method as follows:

$$\text{TokenSAR}(r, x) = \sum_{i}^{N} -\log P(z_i|r_{<i}, x)\tilde{R}_T(z_i, r, x), \tag{3.8}$$

Where $r_{<i}$ is the generated response up until the *i*-th token. **TokenSAR** corresponds to the length normalized predictive entropy (LN-PE) UQ baseline (refer to Section 3.2.1), where the weights are the normalized semantic importance weights $\tilde{R}_T$.

In conclusion, TokenSAR is an uncertainty quantification method that accounts for the varying semantic importance of tokens in an answer, providing more informative and meaningful uncertainty measurements.

MARS: Meaning-Aware Response Scoring for Uncertainty Estimation in Generative LLMs

The authors of the **MARS** paper, Bakman et al. [31], introduce a way to give tokens in an answer semantic weights that reflect their semantic importance given the answer and the context at hand (question). Even though this idea is analogous to what was proposed in the paper by Duan et al. [20], Bakman et al. [31] fix the main limitation of **TokenSAR** – for longer answers the importance weights become approximately equal for all tokens, and obtains these semantic weights in a different way.

The semantic weights given by the **MARS** procedure are the result of a convex combination of two parts: one related to the sequence length ($\frac{1}{N}$) and the other related to measuring the semantic importance of tokens (u):

$$w(r, x, N, i) = \frac{1}{2N} + \frac{u(r, x, i)}{2}, \tag{3.9}$$

where *r* is the answer, *x* the question, *N* the total sequence length, and i the position of a token within an answer. The importance function $u(r, x, i)$ is a BERT like model, Devlin et al. [53], that the authors trained, which has two functions:

1. The first one is to divide an answer into phrases instead of tokens. The authors defend this procedure by stating that tokens are not totally independent of each other, e.g. the tokens: ”William” and ”Shakespeare” are not independent of each other as they relate to the same entity. As such the authors’ model combines these two tokens into the phrase: ”William Shakespeare”.

2. The second function is to make the sematic comparison, between the original answer and the answer minus a certain phrase, given the context at hand. These comparison result in scores that are between 0 and 1, with significant semantic changes receiving scores closer to 0. Accordingly the authors store the values: 1-score.

To obtain the final weights, softmax is applied to the set of importance values obtained for all phrases. These final weights are then equally distributed between the tokens of each phrase.

Given the final weights from **MARS**, it is possible to combine them with UQ methods and obtain

uncertainty measurements, e.g using length-normalized scoring (Malinin and Gales [52]):

$$\text{MARS} = \prod_{i=1}^{N} P(z_i|r_{<i}, x)^{w(r,x,N,i)}, \tag{3.10}$$

where z_i is the *i*-th token and $r_{<i}$ the response r until the *i*-th token.

In summary, MARS is an uncertainty quantification method that assigns semantic weights to phrases and tokens based on their contextual importance, all the while addressing limitations of TokenSAR for longer answers.

3.3 Applying conformal prediction to LLM generations

Next we present a case of applying a non-exchangeable conformal prediction procedure to the generation of responses by an LLM.

3.3.1 Non-Exchangeable Conformal Prediction

Ulmer et al. [54] proposed a method called non-exchangeable conformal nucleus sampling that is based on conformal prediction (where the calibration and test data are non-exchangeable). When an LLM generates a response, this method creates a prediction set of possible tokens to be chosen (for each token decoding step). In order to do so, this method works as follows:

1. An LLM begins generating a response for a prompt.
2. At a token decoding step, a latent representation from the decoder is obtained.
3. This latent representation is used to query a datastore for similar previously stored representations using a k-nearest neighbors (kNN) algorithm.
4. Gather the non-conformity scores from the similar latent representations – which were obtained using the adaptive prediction sets non-conformity function, explained in Section 2.3.2.
5. Using the theory for non-exchangeable conformal prediction presented by Barber et al. [28], a $\hat{q}$ is computed so that the coverage seen in the prediction sets is closed to what is expected (using equation 2.7).

The authors later show in their experiments that the loss of coverage (relative to what is expected) is limited and that their method performed similarly or better than other sampling methods in terms of generation quality (in the considered tasks). This showcases a successful use of conformal prediction in the case of natural language generation. Unfortunately, we cannot directly use this work, as this method faces the limitation of being computationally inefficient during generation time – which goes against our goal of developing an UQ method that can be efficiently integrated into LLMs.

Chapter 4

Methodology

In this chapter we introduce our proposed white-box UQ method for language generation called **TokenCP** and provide a detailed explanation of its theoretical foundations and inner-workings. Additionally, we also go through the evaluation pipeline for our method, which is aligned with currently practiced methodologies.

4.1 TokenCP

Having in mind the objectives mentioned in Section 1.2, we now provide a detailed description of how our proposed white-box UQ method works and the hypothesis that ground it.

4.1.1 Motivation

For a given input, a standard CP procedure creates a prediction set, with its size ideally reflecting the model's uncertainty about the produced output. Larger prediction set sizes express higher model uncertainty (since there are more possible candidates to choose) whereas smaller prediction set sizes convey smaller model uncertainty, as mentioned by Angelopoulos and Bates [10].

When an LLM generates an answer, there are multiple token decoding stages. If we apply a CP procedure for each stage we obtain multiple token prediction sets in the end, with each size representing a proxy token-level uncertainty estimate. It is possible to combine these sizes to obtain an uncertainty estimate of the whole generation, which is more informative than the independent unitary measurements – as we are able to take into account the token's semantic relationships and context relevance. In Section 4.1.4, we provide a detailed explanation of how we compute the sentence level uncertainty estimate.

4.1.2 Algorithm

To meet the presented desiderata, the algorithm of our proposed UQ method (presented in Figure 4.1), has two main components: the **conformal prediction procedure** (explained in Section 4.1.3) which produces a prediction set for each token decoding stage and the **prediction set size analysis**

(explained in Section 4.1.4) which returns proxy token-level uncertainty measurements (for each token decoding step).

These two components must work in unison, as the **CP procedure** component directly creates the basis for the proxy token-level uncertainty scores given by the **prediction set size analysis** component. Ideally, the **CP procedure** should be adaptive (see Section 2.3.4), while the **prediction set size analysis** should return informative proxy scores (that when combined can correlate with the correctness of the generated answer). Next we detail the inner workings of these two crucial components.

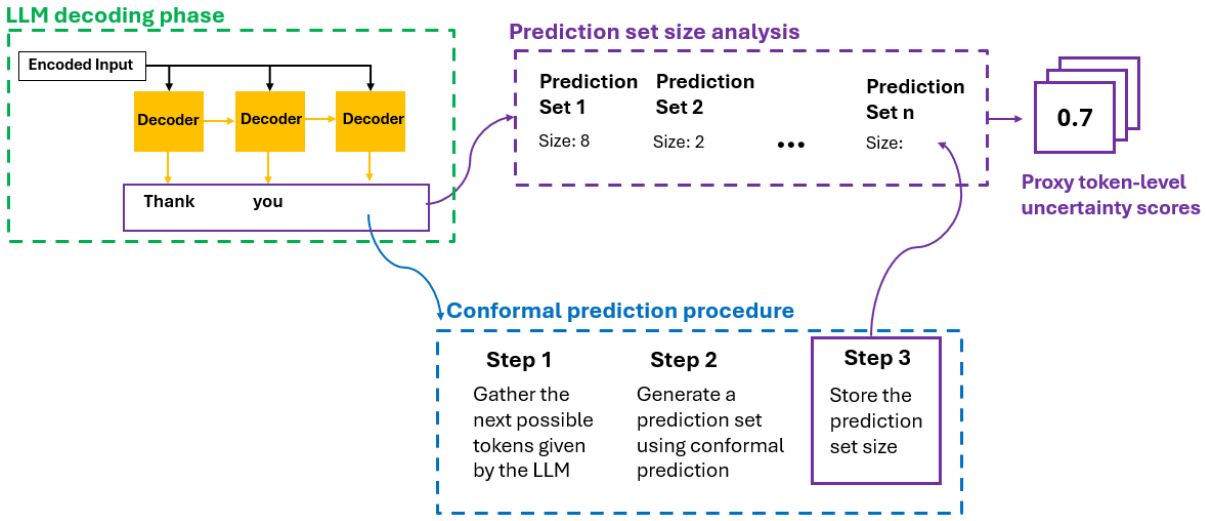


Figure 4.1: Suggested UQ method

4.1.3 Conformal prediction component

As mentioned before, our algorithm incorporates a CP procedure (with its theoretical foundations explained in Section 2.3). The CP procedure used in TokenCP (shown in Figure 4.2), has two steps:

1. **Calibration step**. This step is done before the generation time begins – due to the calibration dataset we have chosen to use (more details in Section 4.1.3). Firstly, the calibration data points are gathered. Then, their respective non-conformity scores are computed and the quantile $\hat{q}$ is obtained, which we will later use to create prediction sets for tokens.

2. **Prediction step**. At each token decoding step we obtain a set composed of all the tokens in the token vocabulary and their respective softmax scores. Using the selected non-conformity function we can obtain the non-conformity scores of every token, and with $\hat{q}$ we can create a token prediction set.

In this work, we use the *top-k* sampling method (refer to Section 2.1.2) in LLMs to generate answers, which makes any token outside the *top-k* have a softmax score of 0. As such, each prediction set as a maximum size equal to the value of *top-k* (any token outside the *top-k* would have the maximum possible non-conformity score).

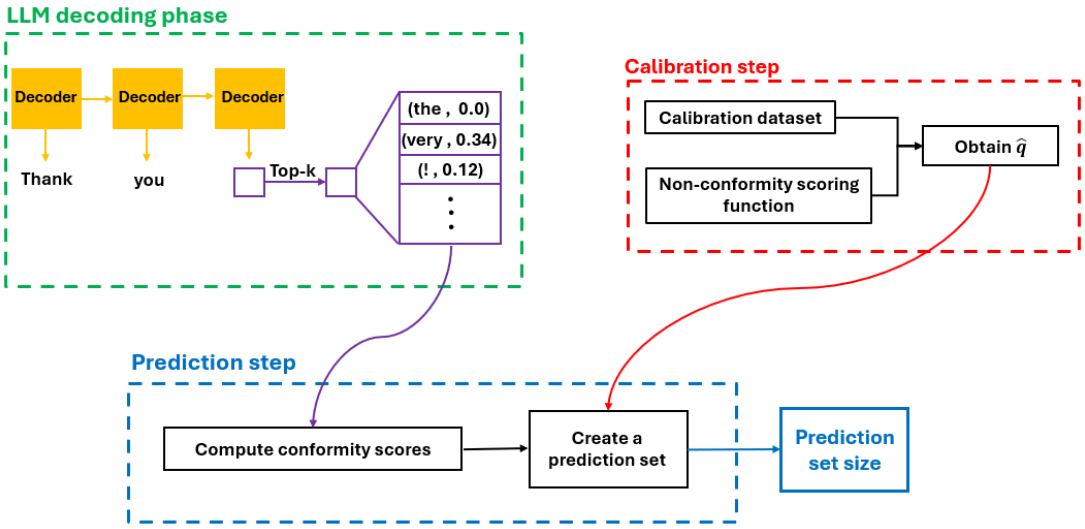


Figure 4.2: Detailed conformal prediction procedure

The adaptiveness of a conformal prediction procedure is paramount for our task since our UQ method gives proxy token-level uncertainty measurements based on the size of the prediction sets, as mentioned throughout the previous section. This adaptiveness lies on the choices for the different constituents of the overall procedure, of which we highlight: **non-conformity scoring function** and **calibration dataset**.

Non-conformity scoring functions

For the non-conformity scoring functions we have considered two options (with their theoretical foundations explained in Section 2.3.2), of which we give their formulas in regard to our proposed algorithm:

- **Least Ambiguous set-valued Classifiers (LAC)**:

$$s(x, r_{<i}, z_t) = 1 - f(z_t). \tag{4.1}$$

  For each decoding step *i* of the generation of a response *r*, we obtain a set T_i of all tokens from the token vocabulary. In order to compute the non-conformity scores for all these tokens, we must follow equation 4.1. In this equation, x is the prompt, $r_{<i}$ the response generated until the *i*-th decoding step, z_t the *t*-th token of set T_i and $f(z_t)$ its corresponding softmax score.

- **Adaptive Prediction Sets (APS)**:

$$s(x, r_{<i}, z_g) = \sum_{z_t \in T_i : f(z_t) \geq f(z_g)} f(z_t). \tag{4.2}$$

  As mentioned before in Section 2.1.4, for every decoding step *i* a token is selected from T_i – denoted as z_g (*the golden token*), for the generated response r. For computing the non-conformity score for the selected token, using APS, we must first sort descendingly all the tokens in the set T_i based on their softmax scores (the *top-k* tokens will be on the first *k* spots of the sorting). Then

we sum the softmax scores, $f(z_t)$, of all the tokens whose softmax score is greater than or equal to that of the selected token $f(z_g)$, during generation.

Calibration dataset

In this work, we considered only the bag-of-tokens calibration dataset, with other options discussed in Chapter 7.

As mentioned in Section 2.3.1, a calibration dataset for a CP procedure contains pairs of inputs and corresponding true labels. In this work, we assume that the tokens appearing in correct LLM generations (see Section 4.2.2 for a discussion of correctness) serve as the true, or ground-truth, labels for the input – which is the prompt and generated answer so far. Accordingly, the **bag-of-tokens** calibration dataset is composed of the input and the softmax score of the selected token. With each data point extracted from a batch of correct LLM generations. As such, the construction of this calibration dataset (illustrated in Figure 4.3) begins by generating an answer, evaluating its correctness, and if the answer is deemed correct, storing the softmax scores of all tokens that comprise the answer. We also keep the softmax scores of the remaining *top-k* tokens for when using the APS non-conformity function, but this is omitted in Figure 4.3.

In this work we use the same calibration dataset for all prediction steps, thereby allowing the calibration step to be performed once and before the generation phase begins (which greatly enhances the efficiency of our algorithm). Lastly, since tokens are generated in an auto regressive manner, the **bag-of-tokens** calibration dataset is non-exchangeable (refer to Section 4.1.5), thus making the CP procedure in our algorithm loose its theoretical guarantees.

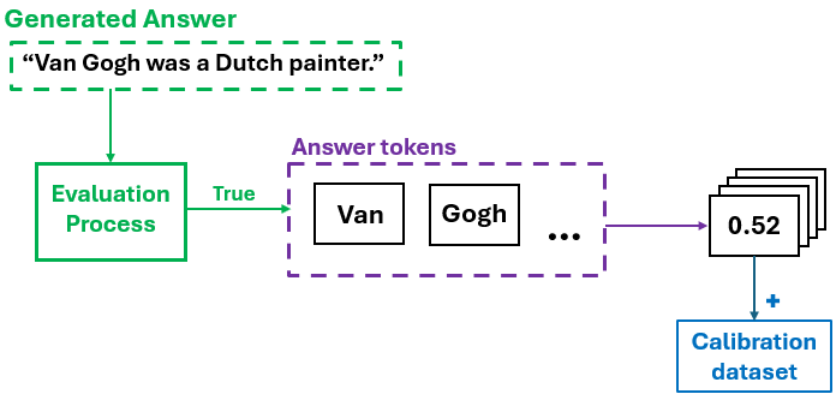


Figure 4.3: Calibration dataset creation process

4.1.4 Prediction set size analysis component

The process of both obtaining a proxy token-level uncertainty measure from a list of token-level prediction, and a global sentence-level uncertainty measure are detailed next.

Prediction set size transformation

We want to transform the sequence of prediction set sizes into values between 0 and 1, hence mimicking a confidence/uncertainty score such as the *softmax* scores of the selected tokens. As mentioned previously, a prediction set size ideally reflects the model's uncertainty. For this reason, we divide it by the maximum possible prediction set size (equal to the *top-k* value), yielding a value between 0 and 1 that represents the model's uncertainty in selecting the next token. By convention, values closer to 0 indicate higher uncertainty.

Now, let's consider the case where the prediction set size is 12 and the maximum prediction set size is 15, then we would have: $\frac{12}{15} = 0.8$. Diving these terms gives us a score close to 1, however since the prediction set size is close to the maximum prediction set size the model should be highly uncertainty (since there are many possible candidates tokens to choose), as such this score should be closer to 0 rather than to 1. Accordingly, we should instead do: $1 - \frac{12}{15} = 0.2$, which is a score more aligned with the presented ideas.

Additionally, there are two edge cases that must be taken into consideration: (1) when the prediction set size is equal to the *top-k* value and (2) when the prediction set size is equal to 0. For example, in the first case, considering $k = 15$, we have: $1 - \frac{15}{15}$, which yields a score of zero. So far it would create no problem, however if we take into consideration the follow up process – of using this score in an uncertainty quantification method, it would make the computation of uncertainty measurements requiring the logarithm impossible. In the second case, considering for example: $1 - \frac{0}{15}$, yields a score of one, which reflects the lowest possible model uncertainty. This is inconsistent with CP theory considering that in this scenario a prediction set size of zero means that no token was eligible for selection. As such the score given should alternatively be zero (hence reflecting the highest possible model uncertainty), therefore leading to the same problem as before. Accordingly, for both cases, we transform the prediction set sizes into a small value close to zero.

Hence, we propose the following function (named after the overall UQ method TokenCP), that transforms the prediction set sizes into scores between 0 and 1, considering the previous comments. These scores can be obtained through the branches of function 4.3.

$$\text{TokenCP}(c_i, c_{max}) = \begin{cases} 1 - \frac{c_i}{c_{max}} & \text{, for } c_i \neq c_{max} \wedge c_i \neq 0. \\ \epsilon & \text{, otherwise,} \end{cases} \tag{4.3}$$

where c_i denotes the prediction set size of *i*-th token, c_{max} the maximum prediction set size that can be obtained (in our case it is equal to the *top-k* value) and ϵ a small value close to 0. In this work we set ϵ to 10^{-10}.

Obtaining uncertainty measurements - TokenCP versions

As mentioned before, after having obtained a sequence of scores from TokenCP we can pass them to other UQ methods, wich will transform the scores into a single sentence level uncertainty measurement (as shown in Figure 4.4). It is this combination that we call a *TokenCP version*. Some possible

combinations are explained next:

- **TokenCP + Maximum sequence probability (MSP)**, defined as:

$$\text{MSP}(r, x) = 1 - \prod_{i=1}^{N} g_i, \tag{4.4}$$

  with *r* being the response generated, x the prompt, *N* the total number of tokens in response *r* and g_i the *i*-th score from TokenCP.

- **TokenCP + Length normalized predictive entropy (LN-PE)**, outlined as:

$$\text{LN-PE}(r, x) = \frac{1}{N} \sum_{i}^{N} -\log(g_i). \tag{4.5}$$

- **TokenCP + TokenSAR**, defined as:

$$\text{TokenSAR}(r, x) = \sum_{i}^{N} -\log(g_i) \tilde{R}_T(z_i, r, x), \tag{4.6}$$

  with $\tilde{R}_T(z_i, r, x)$ being the normalized semantic importance weights (more details in Section 3.2.2).

Every combination presented before will give one uncertainty measurement score per generated answer.

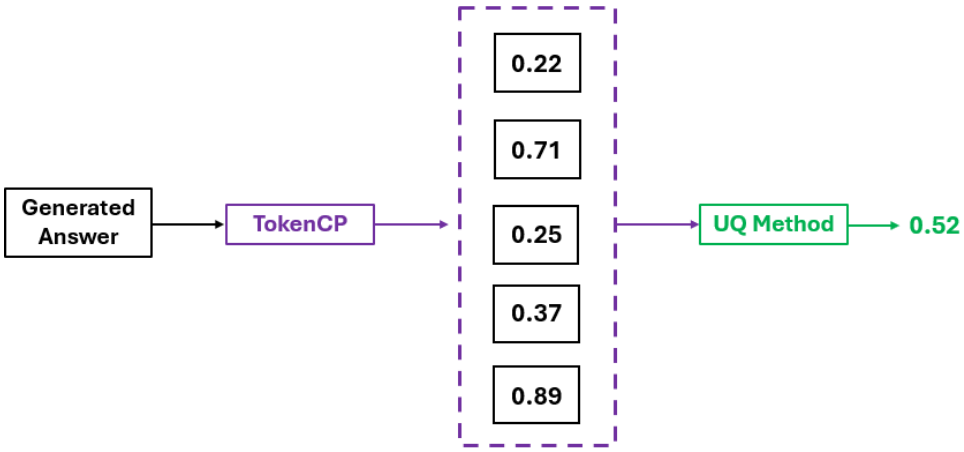


Figure 4.4: Combination of TokenCP with another UQ token-level method

4.1.5 Differences between the CP procedure in TokenCP and the standard approach

As mentioned throughout this document TokenCP makes use of a conformal prediction procedure, however, not in the traditional way:

- Firstly, CP is typically used in a classification setting where a true label exists. In our work, we make the mapping of sentence correctness into token correctness – i.e. all tokens in a correct answer are also considered correct (as stated in Section 4.1.3). However, token-level true labels do not exist – as the use of different tokens could still lead to correct generations. Consequently, there is no direct notion of coverage in our case.

- Secondly, the formal guarantees of CP require that the calibration and test data are exchangeable, which means that the joint probability of the random variables generating the data is invariant under permutations thereof (as explained by Campos et al. [9]). In accordance to this definition, the data in our case is non-exchangeable since tokens are generated in an autoregressive manner. This is a common scenario in NLP given the interdependent nature of natural language. However as shown in Section 3.3.1, the use of CP in non-exchangeable settings can still be successful.

4.2 Evaluating our approach

In this section we go over the details of the evaluation pipeline for a *TokenCP version* (TokenCP plus another UQ method). As we can see in Figure 4.5, our evaluation process follows what is typically used in the literature (see Section 3.1.2). Accordingly, we begin by choosing a task and generating answers. Subsequently, the answer correctness is assessed and an uncertainty measurement for the generated answer is obtained (using a TokenCP version). In the end, using multiple uncertainty measurements and answer correctness values, we can use a metric to study their correlation. However, we have added one small change to the assessment process of an answer – by using both sentence similarity metrics and an LLM judge. This change is further explained in Section 4.2.2.

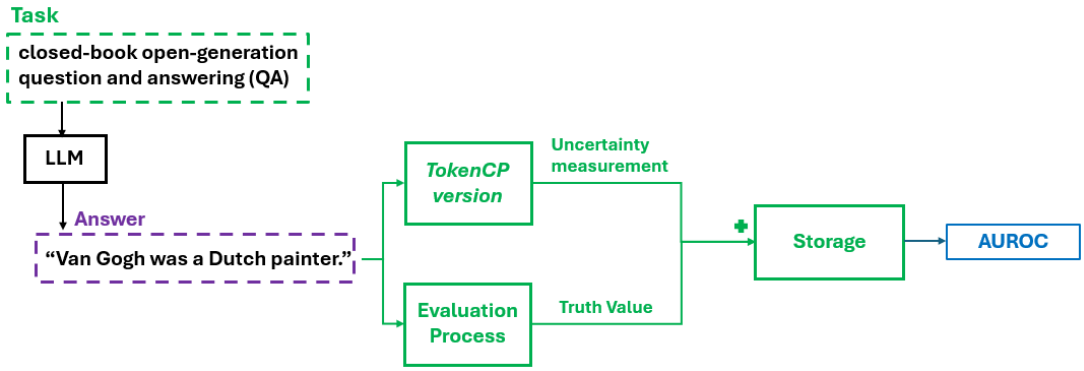


Figure 4.5: Evaluation pipeline

4.2.1 Similarities with common evaluation pipeline

As mentioned in Section 3.1, the main idea behind the evaluation of UQ methods is to study how the uncertainty measurements from UQ methods correlate with the correctness of LLM responses. So as to possibly classify an LLM answer as either correct or wrong we must choose a task that allows to do

so. For this work, we have chosen the closed-book open-generation question and answering (QA) task (explained in Section 3.1.2), since it is a simple task and widely used.

Additionally, the correlation between uncertainty measurements and answer correctness can be studied in many different ways (refer to Section 3.1.2). In this thesis we have decided to use the AUROC score as done in the work of: Duan et al. [20], Bakman et al. [31] and Aichberger et al. [34]. We made this decision because AUROC not only is an easily interpretable metric and typically used in the evaluation of UQ methods, but also that in our case would not require the transformation of binary values (e.g. answer correctness) into continuous values.

4.2.2 Changes to the typical evaluation pipeline

The last piece of the evaluation pipeline (see Figure 4.5) to be discussed is the assessment process (where an answer is classified as either correct or wrong). Our evaluation process is rooted on common practices in the literature. However we have made one change to this process, with the goal of increasing its robustness. This is crucial since answer misclassifications would contaminate the AUROC score.

Our evaluation process combines the two tests mentioned in Section 3.1.2. Passing the first test (semantic similarity using a BERT-like model) would not suffice as two answers contradicting each other, can still be considered semantically similar. Accordingly, we introduced a second test – using an LLM evaluator, that assesses if the generated answer supports the reference answer (as one plausible valid answer). We elaborate on each test below.

1. **Semantic similarity:** In this test, a generated candidate answer, for example: ”The capital of Kazakhstan is Astana.”, and a corresponding reference answer: ”Astana.”, are transformed into embeddings using a BERT [55] like model, called MPNET [56] (which produces granular sentence embeddings given its small size of 100M parameters). These two vectors are then compared using cosine similarity, for which we obtain a score between 0 and 1. If this score is bigger than 0.49 (indicating relevant similarity between these two sentences), we consider that the candidate answer passed this test. We relaxed the similarity score threshold from the usual 0.50 to 0.49 to facilitate the creation of the bag-of-tokens calibration dataset. This adjustment does not affect the validity of the evaluation, as an additional test (LLM judge) was introduced.

2. **LLM Judge:** For this last test, a *Falcon3-7B-Instruct* LLM [57] acting as a judge determines whether the candidate answer supports the reference answer. If it does, then the candidate answer passes this final test and is considered correct.

4.2.3 Model and dataset diversity

Finally, we wanted to assess the robustness of our proposed method across different datasets and models, where ideally it would maintain a similar AUROC score when varying these. As such, this led us to test our method on different LLMs and datasets, as detailed in Chapter 5. The LLMs we experimented with were the following: *Qwen2.5-7B* [58], *Llama-3.1-8B* [59] and *Gemma-2-9B* [60], having a total

number of parameters raging from 7 billion to 9 billion (therefore being small LLMs). When it comes to the datasets, we used the TriviaQA [36] and NaturalQA [37] datasets, as usually done in the literature (see Section 3.1.2).

Chapter 5

Implementation

In this chapter we discuss all the domains related to the implementation of a *TokenCP version*, from software to hardware. Besides this, we explain the structure of the overall code and the reasoning behind the models and datasets that were used.

5.1 Infrastructure

In this section we detail all the software and hardware used to implement our uncertainty quantification pipeline and to carry out the different experiments employed to evaluate our proposed approach.

5.1.1 Models

Choosing the most adequate models was essential for us, as we needed models that had a good compromise between: speed of inference (the time required to generate a response) and its performance (in our case being the ratio between correct answers and the total number of answers). The first factor had an affect on how fast we could debug and test our implementation, whereas the second had an affect on how much we could grow the calibration dataset in size. Considering that some datasets had small development (dev) and test sets, poorly performant models could not get the number of correct answers we wanted (so that the calibration dataset had a specific size).

Suitably, we constrained our LLM options to those that we could fit inside our GPUs and that were sufficiently competent (so that we could obtain the intended size for the calibration dataset). Hence, we choose small LLMs which had between 7 and 10 billion parameters (we avoided models with less than 7 billion parameters as these often answered question incorrectly). As mentioned in Section 4.2.3, we selected the following models: *Qwen2.5-7B* [58], *Llama-3.1-8B* [59] and *Gemma-2-9B* [60], with a brief description about these present in Table 5.1.

Table 5.1: LLMs description. DPO stands for Direct Preference Optimization (Rafailov et al. [61]) and RLHF stands for Reinforcement Learning From Human Feedback (Ouyang et al. [62]).

Model	Tokenizer	Token Vocabulary Size	Amount of Pretraining	Human Alignment
Qwen2.5-7B	Byte-Pair Encoding	$\approx 152,000$ tokens	≈ 18 trillion tokens	DPO
Llama-3.1-8B	SentencePiece	$\approx 128,000$ tokens	≈ 15 trillion tokens	RLHF
Gemma-2-9B	Byte-Pair Encoding	$\approx 256,000$ tokens	≈ 8 trillion tokens	RLHF

5.1.2 Datasets

As mentioned in Section 4.2.3, we chose the TriviaQA (Joshi et al. [36]) and NaturalQA (Kwiatkowski et al. [37]) as the datasets to be used in this research work. These two datasets are commonly chosen in works related to uncertainty quantification, as seen in: Bakman et al. [31], Duan et al. [20], Nikitin et al. [63], Aichberger et al. [30] and Yaldiz et al. [35]. This preference is due to both their considerable number of test examples ($>$1000) and development examples ($>$4000), and their simplicity. In this work, we used the test examples to quantify AUROC scores of UQ methods, whilst the development examples were used to create the calibration dataset used in TokenCP.

In table 5.2, we give examples of both questions and corresponding valid answers, for both datasets. Using these two datasets, we can make an LLM generate answers to questions (e.g. to those in Table 5.2), by incorporating these in the prompt presented in Section A.1.

Table 5.2: Examples of datasets' datapoints

Dataset	Question	Valid Answers
TriviaQA	What was the name of painter van Gogh's brother to whom he wrote many letters?	[Theo, theo]
NaturalQA	Who did Deion Sanders go in the hall of fame as?	[Cornerback]

Comparing the two datasets, we can see that NaturalQA appears to be more complex (meaning that it is harder for an LLM to answer questions from this dataset correctly). This is observed in Table 5.3, which presents the correctness (number of correct answers divided by the total number of answers) of each model for each dataset – evaluated using the procedure described in Section 4.2.2. Across all models, correctness is consistently lower on NaturalQA than on TriviaQA, indicating the greater complexity of the former.

Table 5.3: Percentage of correct answers by each model for a batch of 200 questions (test set).

Dataset	**Model**	**Correctness**
TriviaQA	Qwen2.5-7B	39 %
	Llama3.1-8B	46 %
	Gemma-2-9B	35 %
NaturalQA	Qwen2.5-7B	20 %
	Llama3.1-8B	18 %
	Gemma-2-9B	14 %

This difference can be caused by the fact that the questions in NaturalQA come from real anonymized, aggregated queries issued to the Google search engine (leading to significant syntactic and lexical variability between questions), whereas TriviaQA's questions come from trivia enthusiasts (hence being more structured questions). For these reasons, the TriviaQA's questions tend to be bigger in comparison to NaturalQA's. In addition, NaturalQA's valid answers are derived from Wikipedia pages, whereas TriviaQA's valid answers were written by the previous trivia enthusiasts. As a result, TriviaQA valid answers tend to be shorter and more concise than those in NaturalQA. Table 5.4, which reports the average question and answer lengths for the test sets of both datasets, shows that TriviaQA generally contains longer questions than NaturalQA, while its reference answers are shorter.

Table 5.4: Dataset statistics (test set)

Dataset	Average Question Length (strings)	Average Answer Length (strings)
TriviaQA	79	15
NaturalQA	49	27

5.1.3 Software

The main programming language used was the *Python* programming language, as it offered various useful libraries (like: *Pytorch* and *Pandas*). Additionally, it also eased the use of different LLM's and the management of data from the two QA datasets that were used.

When it comes to the LLM's employed, we used open-source models, which were available in *Hugging Face*. We prioritized using open-source models as our proposed UQ method is a white-box method, therefore requiring the softmax scores of the generated tokens.

5.1.4 Server

For the implementation of this master thesis we used an *Instituto de Telecomunicações* (IT) server, that had three *RTX 2080 Ti - 12GB* GPUs and twelve *AMD Ryzen 2920X @ 3.50GHz* CPUs, with a RAM of *128GB*. Without this hardware, we would not have been able to load these models (due to memory constraints) and to get generate answers quickly.

5.2 Code Workflow

The code workflow for the combination of TokenCP with another UQ method can be divided into three steps:

- Firstly, sets of evaluated question/answer pairs are created.
- Secondly, we create a calibration dataset and apply conformal prediction to a test set of question/responses pairs, thus obtaining prediction set sizes for all the tokens in the answers (from the

test set).

- Lastly, we transform prediction set sizes into a set of scores, thus completing the TokenCP algortihm. These scores are later transformed into a global uncertainty measurement using the choosen UQ method (more details in Section 4.1.4).

Because the code is divided into distinct components, each part of a *TokenCP version* can be modified independently. As such, this code structure allows for faster testing/debugging of new ideas and features. Lastly, the code for this work will be made public in the following link: *https://github.com/Dolfas/TokenCP*.

5.2.1 Step 1: Data preparation

It all begins with the construction of a dataset of 200 correct answers (their respective questions are also stored), which were evaluated using the procedure explained in section 4.2.2. This dataset will serve as the building block of the calibration dataset for the CP procedure in **step 2**, as at this step the tokens in each answer do not have non-conformity scores yet (we will call this dataset as **pre-calibration dataset**). Additionally, a test set of 1000 evaluated question/answers pairs is also created, using the same procedure as before. This step can be observed in Figure 5.1.

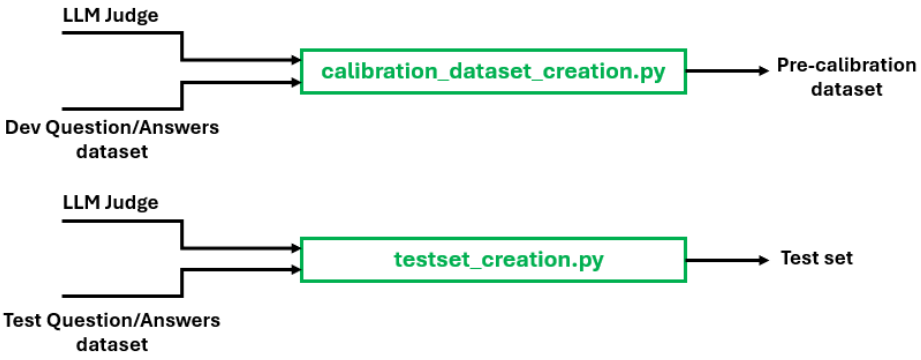


Figure 5.1: Step 1 workflow

5.2.2 Step 2: Calibration dataset creation

Next, we transform the pre-calibration dataset into a calibration dataset, by converting the softmax scores into non-conformity scores. Having a calibration dataset, we can finally apply the conformal prediction procedure to the built test set. Consequently, a refined test set is obtained, which contains evaluated question/answer pairs and prediction set sizes for all the tokens in each answer. This can be summed up in Figure 5.2.

5.2.3 Step 3: TokenCP version

Given that we have a set of question/answer pairs, with each answer having a list of prediction set sizes (one per token), we can apply the procedure described in Section 4.3 to transform these set sizes

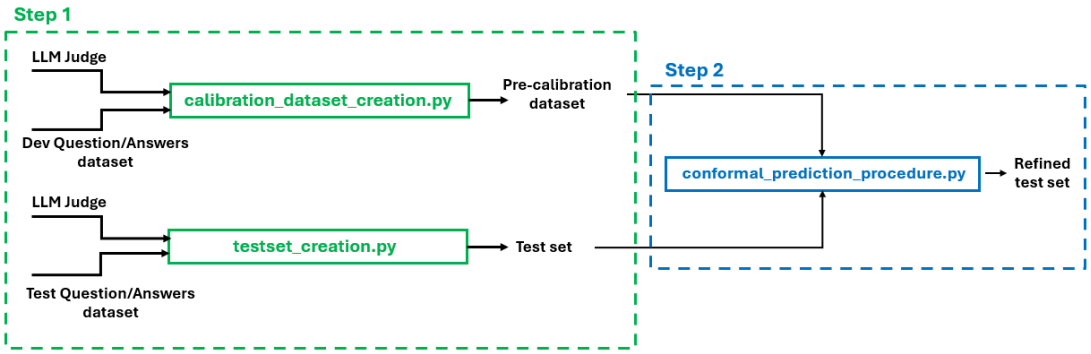


Figure 5.2: Step 1-2 workflow

into scores between 0 and 1. After choosing an UQ method to combine with TokenCP we can obtain a global uncertainty measurements (using these scores). The full workflow is shown in Figure 5.3, where we provide examples of UQ methods that can be combined with TokenCP

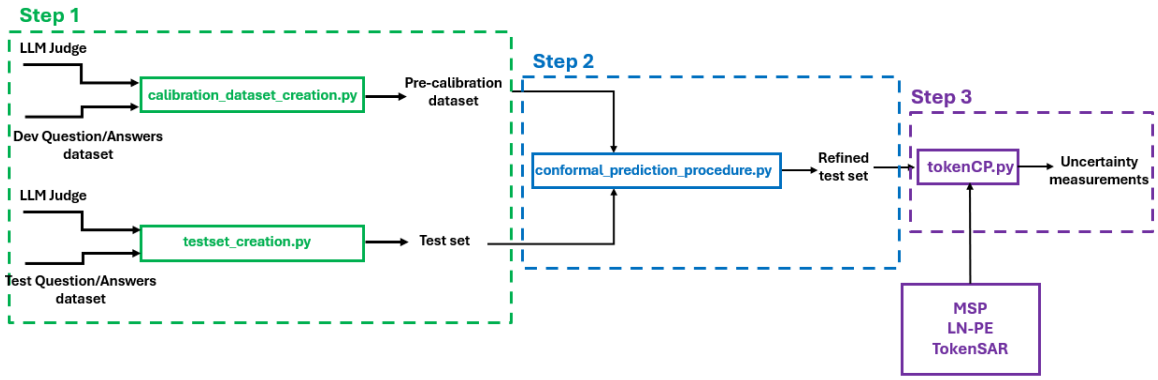


Figure 5.3: Step 1-2-3 workflow

Chapter 6

Results and discussion

In this chapter, we walk through the process of selecting the parameters that influence the performance of TokenCP. Based on these parameters, we then compare the performance of different versions of TokenCP with alternative UQ methods. In the end, we evaluate the CP procedure in TokenCP and provide a potential addition to the method.

6.1 General setting

In the following experiments, four parameters are explored, each directly or indirectly affecting the performance of a *TokenCP version*. Before describing these experiments, we first outline the default setup – fixed when tuning each parameter:

1. Non-conformity function: **LAC**.
2. *Top-k* value: **15**.
3. Subword Approach: **Single**.
4. Method to combine with TokenCP: **MSP**.

6.2 Calibration dataset sizes

We begin our experiments by examining different calibration dataset sizes, as these can affect the adaptiveness of a CP procedure. As the calibration dataset directly determines the calibration threshold $\hat{q}$, which in turn guides the creation of prediction sets. For the use of conformal prediction, Angelopoulos and Bates [10] argue that a calibration dataset of 1,000 data points is enough for most scenarios (given that the calibration and test data are exchangeable). We follow this suggestion and assess if it holds empirically for our case, where the data is non-exchangeable and the procedure is used as part of a larger pipeline. Accordingly, three sizes were considered: 1,000, 2,000 and 3,000. We considered only up to 3,000 since this was the maximum number of calibration data points all models obtained for both datasets. In Figure 6.1, the AUROC score of TokenCP for different calibration dataset sizes is shown.

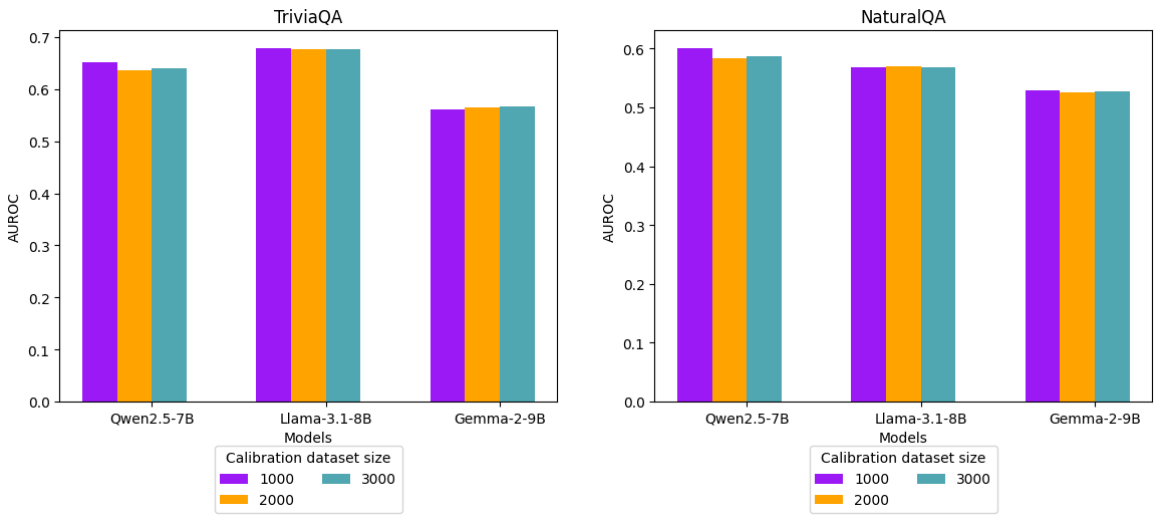


Figure 6.1: AUROC scores for different calibration dataset size

As shown in Figure 6.1, there is no significant difference in the AUROC scores when varying the calibration dataset size, indicating that the proposed calibration dataset size by *Angelopoulos and Bates* [10], works well for our case. Moreover, we hypothesize that changing the calibration dataset size (between these values) had a small impact on the adpativeness of TokenCP. Hence, reflecting no significant difference in the AUROC scores.

Further analyzing Figure 6.1, we can see that the size 1,000 stood out, by outperforming the remaining sizes in 4 of the 6 conducted experiments. Consequently, we decided to set the calibration dataset size to 1,000, which also reduce the required memory by TokenCP.

6.3 *Top-k* values

In this experiment we analyzed the impact of the choice of k in *top-k* has on TokenCP – since it defines the maximum prediction set size, as seen in Section 4.3.

The results presented in Figure 6.2 indicate no consistent correlation between the *top-k* value and the AUROC score (with the effect *top-k* appearing to be model and data dependent). Given that changing the maximum number of predicted classes did not worsen the uncertainty quantification. However, we hypothesize that even with more (or less) candidates tokens to choose the CP procedure of TokenCP still produced prediction sets that adapted to the difficulty of inputs (as there was a small difference in AUROC scores for most cases), therefore showing the robustness of TokenCP.

It is relevant to note the case of Gemma-2-9B + NaturalQA. In this case, there is a considerable difference between the AUROC score for when *top-k* is 15 and the remaining values. Furthermore, for the remaining *top-k* values (30,60,90) the AUROC score goes below 0.5, indicating that the uncertainty measurements given are no better than random guessing. It is plausible that for these values, the CP procedure in TokenCP significantly worsens in adaptiveness (given the increased candidate pool), hence making the *TokenCP version* considered (TokenCP + MSP) output high uncertainty measures for correct answers and low uncertainty measures for incorrect answers. Moreover, Gemma-2-9B is the

worst performing model in terms of UQ (as seen in Figure 6.2), which goes in accordance with other research work: Vazhentsev et al. [23] and Yaldiz et al. [35].

Nevertheless, we chose the *top-k* value of 15, since it not only outperformed all the other *top-k* values in 3 out of 6 possible cases, but also lead to faster LLM generations and less memory consumption.

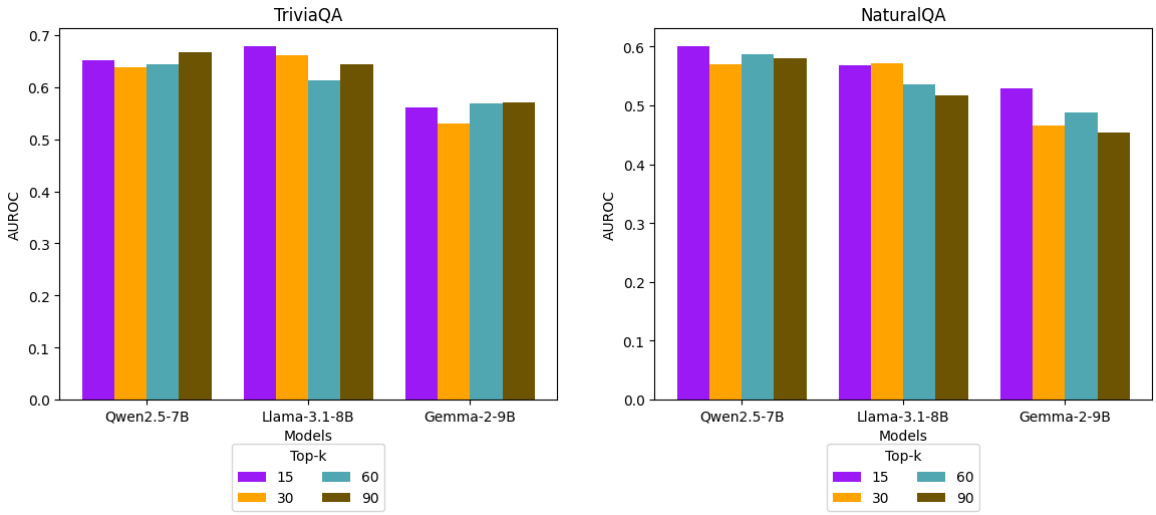


Figure 6.2: AUROC scores for different *top-k* values

6.4 Subword approaches

Given that our approach gives scores at the token level, it is necessary to understand how to approach words that are broken down in multiple tokens – i.e. words that are outside the token vocabulary of an LLM. An example of such words is: “Shakespeare”. This word when generated is separated into two tokens: “Shake” and “-speare”, (we define words that are separated into subwords as multi-token words). Taking inspiration from Bakman et al. [31], we consider two approaches when dealing with words like “Shakespeare”. The fist one called **single** treats both tokens independently – TokenCP outputs a score for each token. In contrast, the second approach called **merged** merges the tokens of words which are separated at generation time. We then made TokenCP return a single score (for the whole word), corresponding to the first token of the word – as the following tokens are just for the completion of the word. However, other options were also available— such as returning the average, maximum, or minimum of the scores, as explored by Bakman et al. [31].

To better understand the impact these two approaches have on the performance of TokenCP, we must understand how often the merges happen in the calibration dataset and testset (for all models and datasets considered). How often we have to merge tokens depends on multiple factors, including: the model’s tokenizer, the generated sentence – increasing the number of multi-token words in a sentence consequently increases the number of merges, the vocabulary size and out-of-vocabulary words.

We now present two tables. Table 6.1 lists the tokenizer used by each model, and Table 6.2 reports the percentage of merges (relative to treating tokens as **single**) in the calibration set, the test set, and in

the correct and incorrect test answers – to assess their impact on the total number of merges in the test set.

Table 6.1: Type of tokenizer used in each model and the token vocabulary size.

Model	Tokenizer
Qwen2.5-7B	Byte-Pair Encoding
Llama-3.1-8B	SentencePiece
Gemma-2-9B	Byte-Pair Encoding

Table 6.2: Percentage of merges in the calibration dataset, test set and for the correct and wrong answers in the test set.

Dataset	Model	Calibration dataset	Test set	Correct answers	Wrong Answers
TriviaQA	Qwen2.5-7B	4.12 %	11.65 %	13.93%	10.54 %
	Llama3.1-8B	4.48 %	6.10 %	6.99 %	5.60 %
	Gemma-2-9B	2.83 %	11.58 %	10.67 %	12.14%
NaturalQA	Qwen2.5-7B	3.25 %	10.39 %	10.08 %	10.45 %
	Llama3.1-8B	3.11 %	4.50 %	4.36%	4.52 %
	Gemma-2-9B	2.30 %	9.87 %	10.01%	9.84 %

As we can see in Table 6.1, Qwen2.5-7B and Gemma-2-9B have the same tokenizer, as such it is expected that these have a similar percentage of merges in the calibration dataset and testset (as the tokenizer defines the subwords to be used by the model). This does happen, as evidenced in Table 6.2 for both the calibration dataset and test set.

Inspecting Table 6.2 we can see that for the calibration dataset the percentage of merges is similar for all models and datasets. However, for the test set we see an increase in the percentage of merges, especially for the Qwen2.5-7B and Gemma-2-9B models. We initially hypothesized that the increase was caused by the presence of wrong answers in the test set. However, Table 6.2 shows that the percentage of merges in correct and incorrect answers is similar, indicating no clear link between the presence of wrong answers and a higher merge rate. Accordingly, we propose that the increase in the percentage of merges is due to the difference in the questions that are used for the calibration dataset (as these come from the validation set) and for the test set (as these come from the test set). These differences may cause a model to generate different amounts of multi-token words. Furthermore, we can see that for the Llama-3.1-8B model the percentage of merges in the test set is smaller in comparison to the remaining models. We posit that this difference is due to the model's tokenizer which lead to the use of different subwords, with respect to the other models considered.

Since the percentage of merges in the calibration dataset and test set, for all the considered models and datasets, is relatively low (at most 11%) we expect to see no significant difference in the performance of TokenCP between using the **single** or **merged** approaches. Moreover, we expect to see a small difference between these two approaches for the Llama-3.1-8B model as the percentage of merges in the calibration dataset and test set is at most 6% (for both datasets). In Figure 6.3 we present the

performance of TokenCP when we use these two approaches.

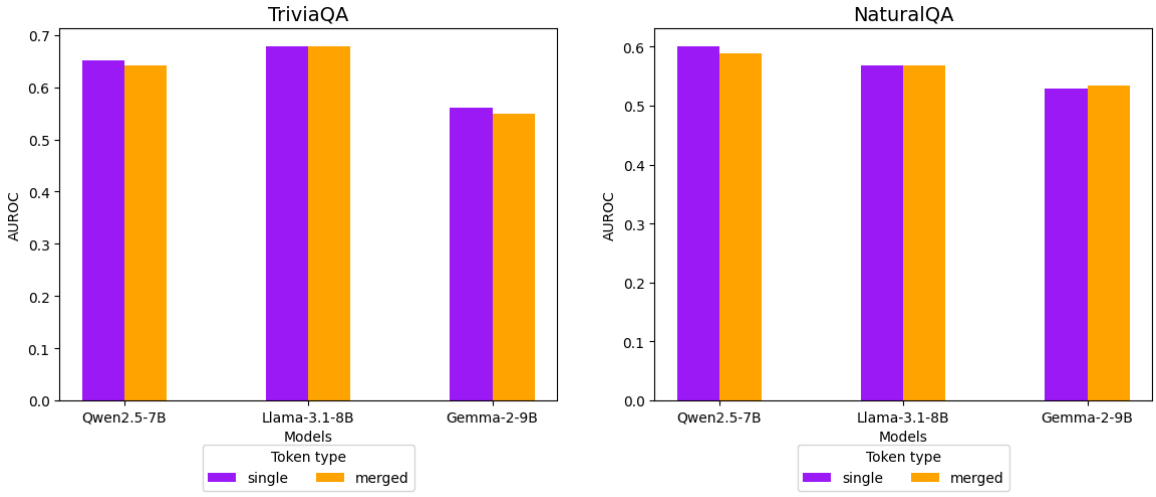


Figure 6.3: AUROC scores for different token types

As it is possible to see in Figure 6.3, there is no significant difference in the performance of TokenCP when using the **merged** or **single** approaches (for all models and datasets considered), with the difference being very small for the model Llama-3.1-8B. These results go in accordance with what was expected before.

Nevertheless, since the **single** approach surpassed the **merged** approach in 5 out of 6 cases, we therefore adopt the **single** approach in subsequent experiments.

6.5 Non-conformity functions

In this section we study the performance of TokenCP when using two different non-conformity functions: adpative prediction sets (APS) and least ambiguous set-valued classifiers (LAC). As stated previously in Section 2.3.2, LAC produces prediction sets with the smallest average size but tends to create small prediction sets for hard instances and larger ones for easy instances, whereas APS attempts to improve adaptiveness but generates larger prediction sets on average. Next, we plot in Figure 6.4 the spread of the prediction set sizes, produced by the CP procedure in TokenCP for the two non-conformity functions.

On one hand, it is possible to see in Figure 6.4 that there is a wide spread of the prediction set sizes for both the LAC and APS non-conformity functions. This is a healthy indicator of adaptiveness, pointing to the discerning capabilities of the procedure, according to Angelopoulos and Bates [10]. Comparing the spread of prediction set sizes for the two non-conformity functions, LAC produces smaller sets on average than APS. Across all models and datasets, LAC yields an average prediction set size of 3.72, while APS yields 9.42. This goes in accordance with what was mentioned before. We also note that LAC exhibits more desirable behavior. It frequently produces prediction sets of size 1–3, which are likely associated with subword continuations, function words, or punctuation. In contrast, APS more often produces larger sets (around size 13–15), suggesting that it may be overcovering token choices.

Therefore, we expect TokenCP to perform better when paired with the LAC non-conformity function than with APS non-conformity function. Subsequently, in Figure 6.5 we present the performance of TokenCP when using either of these two non-conformity functions.

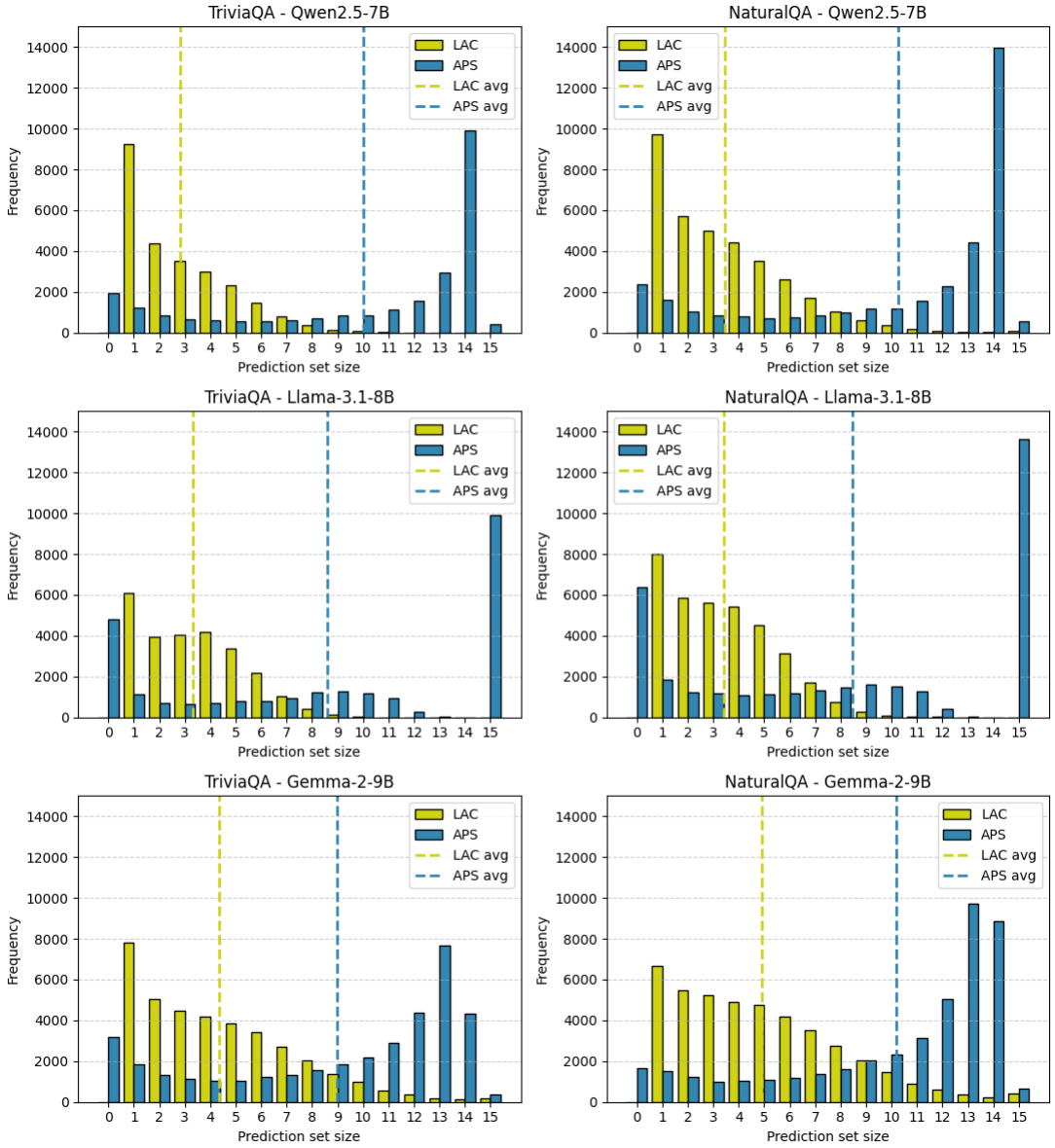


Figure 6.4: Step 1-2 workflow

As it can be seen in Figure 6.5, the use of the LAC non-conformity function in the TokenCP procedure outperforms APS in five out of the six possible cases (when it comes to the AUROC score), with the only exception being the case of: Gemma-2-9B + NaturalQA (where APS may have rendered the CP procedure more adaptive than with the use of LAC). Consequently, we made the decision of choosing LAC as the non-conformity function to be used in the CP procedure of TokenCP.

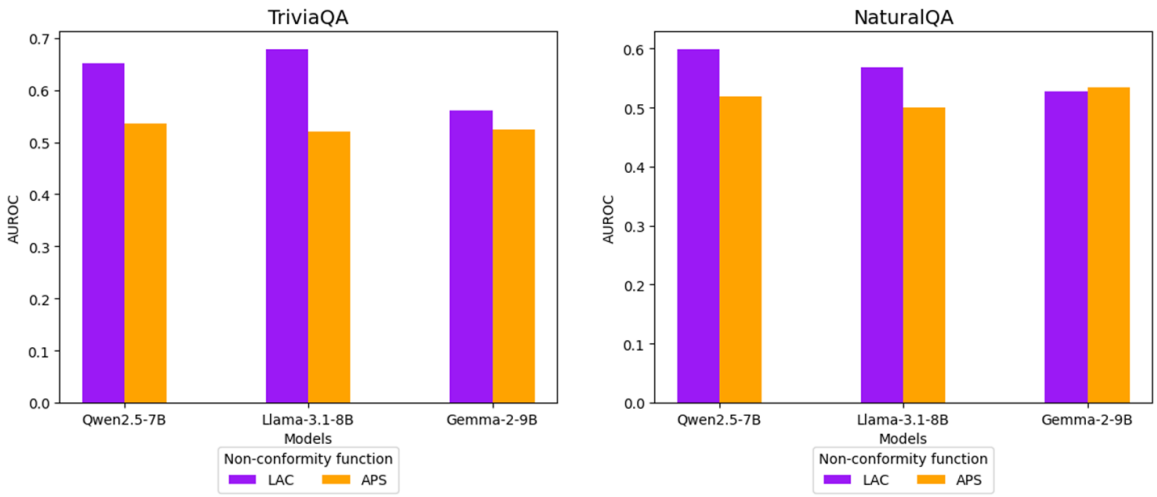


Figure 6.5: AUROC scores for different non-conformity functions.

6.6 Final results

Based on the outcomes of the previous experiments we present the optimal parameters (given the order of the experiments):

- Calibration dataset size: **1000**
- Top-k value: **15**
- Subword approach: **Single**
- Non-conformity function: **LAC**

Having the optimal setting defined, we move on to compare the different *TokenCP versions* to alternative UQ methods (discussed in Section 3.2), in terms of AUROC score. The results are presented in Table 6.3.

Examining Table 6.3, we can see that the maximum token entropy (**MTE**) UQ method achieved the highest AUROC score in 4 out of 6 possible cases, hence being the top performing UQ method overall (for the considered models and datasets). Although there is no single study that compares all of the aforementioned UQ methods using AUROC, the works of Fadeeva et al. [32] and Vashurin et al. [44] indicate that the baselines employed in this study achieve comparable or superior performance relative to state-of-the-art approaches (including TokenSAR). Therefore, these results seem consistent with the current literature. Among *TokenCP versions*, **TokenCP (LN-PE) + TokenSAR** performs best, showing the smallest average relative difference to the highest AUROC score of 4.1%. Moreover, this version surpasses all other UQ methods in one case (NaturalQA + Qwen2.5-7B). Specifically, it outperforms the state-of-the-art TokenSAR on the NaturalQA dataset, while the opposite is observed on the TriviaQA dataset. This dissimilar behavior for the two datasets, can be due to the greater complexity of NaturalQA compared to TriviaQA (meaning that a model has a harder time answering questions correctly from NaturalQA than from TriviaQA), as discussed in Section 5.1.2.

Table 6.3: AUROC scores comparison between UQ methods. Scores in bold represent the highest score for a certain model and dataset. Percentages in red or green indicate the relative difference between a score and the score in bold.

Dataset	UQ Method	Model		
		Qwen2.5-7B	**Llama-3.1-8B**	**Gemma-2-9B**
TriviaQA	MSP	0.677	0.681	0.555
	LN-PE	0.652	0.626	0.574
	MTE	**0.722**	**0.702**	0.599
	TokenSAR	0.710	0.676	**0.609**
	TokenCP (MSP)	0.652 (-9.7%)	0.678 (-3.4%)	0.562 (-7.7%)
	TokenCP (LN-PE)	0.624 (-13.6%)	0.590 (-15.9%)	0.584 (-4.1%)
	TokenCP (LN-PE) + TokenSAR	0.673 (-6.8%)	0.642 (-8.5%)	0.582 (-4.4%)
NaturalQA	MSP	0.585	0.577	0.530
	LN-PE	0.602	0.575	0.531
	MTE	0.630	**0.620**	**0.563**
	TokenSAR	0.645	0.596	0.553
	TokenCP (MSP)	0.601 (-8.9%)	0.567 (-8.5%)	0.529 (-6.0%)
	TokenCP (LN-PE)	0.635 (-3.8%)	0.566 (-8.7%)	0.556 (-1.2%)
	TokenCP (LN-PE) + TokenSAR	**0.660** (+0%)	0.598 (-3.5%)	0.556 (-1.2%)

Finally, it is worth noting that combining TokenCP with TokenSAR increased the performance of our UQ method (for the LN-PE version) by yielding an average relative difference (to the highest score) of 4.1% – smaller than 7.8% for the standalone case. In contrast, pairing TokenCP with other baselines alone (LN-PE or MSP) often does not yield improvements. We argue that this difference may stem from how the UQ baselines combined with TokenCP handle the tokens from an answer. In particular, assigning importance weights to tokens when estimating uncertainty – i.e. identifying which tokens are most relevant for uncertainty computation, seems to be a more effective strategy when integrating UQ methods with TokenCP. This hypothesis motivates the following Section 6.7, where we explore the use of a mechanism that finds the most relevant tokens in answer.

6.7 Token Selection

In the closed-book open-generation question and answering (QA) task an LLM gives an answer to a question present in the prompt. However, the LLM can not be forced to answer with the least tokens, so its answer can include semantically redundant tokens. Lets consider the previous example of the question: ”What’s the capital of Kazakhstan?” to which a model answers: ”The capital of Kazakhstan is Astana.”. In this situation, the relevant part of the answer (in regard to the question) is ”Astana”, with the remaining tokens being semantically redundant. The authors of Duan et al. [20], demonstrate that LLM answers frequently include semantically redundant tokens, and that by accounting for this redundancy the quality of the uncertainty measurements is improved. We then hypothesize that considering only the most relevant tokens in an answer – when computing sentence uncertainty, will improve the performance of the TokenCP method (and of the other considered UQ methods), as done by Bakman et al. [31] and Duan et al. [20]. In order to select the most relevant tokens in an answer, we use an LLM (*Falcon3-7B-Instruct* [57]) acting as an evaluator, which receives the prompt presented in Section A.2 and outputs

what it considers to be the most relevant tokens in an answer.

Unfortunately this approach has a limitation which can undermine its addition to UQ methods. It fails to detect any relevant token in an answer approximately on average 30% of the time. This can happen for some reasons, including: the generated answer (which can be an LLM hallucination), the LLM evaluator and the prompt used. In the scenario where the evaluator fails to identify any relevant token in an answer, we consider all the tokens in the answer as relevant. In Table 6.4 we present the results referring to the use of token selection both in TokenCP and the considered baselines.

Table 6.4: AUROC comparisons between UQ methods considering the use of token selection. UQ methods with **+ Selection** indicate the use of token selection. The percentages show the relative difference between the AUROC score of a UQ method and the same UQ method **+ Selection**.

Dataset	**UQ Method**	**Model**		
		Qwen2.5-7B	**Llama-3.1-8B**	**Gemma-2-9B**
TriviaQA	MSP	0.677	0.681	0.555
	LN-PE	0.652	0.626	0.574
	MTE	**0.722**	**0.702**	0.599
	MSP + Selection	0.666 (-1.6%)	0.670 (-1.6%)	0.660 (+18.9%)
	LN-PE + Selection	0.706 (+8.3%)	0.692 (+10.5%)	**0.67** (+16.4%)
	MTE + Selection	0.675 (-6.5%)	0.680 (-3.1%)	0.642 (+7.2%)
	TokenSAR	0.710	0.676	0.609
	TokenCP (MSP)	0.652	0.678	0.562
	TokenCP (LN-PE)	0.624	0.590	0.584
	TokenCP (LN-PE) + TokenSAR	0.673	0.642	0.582
	TokenCP (MSP) + Selection	0.612 (-6.1%)	0.633 (-6.6%)	0.635 (+13.0%)
	TokenCP (LN-PE) + Selection	**0.639** (+2.4%)	0.657 (+11.4%)	**0.637** (+9.1%)
NaturalQA	MSP	0.585	0.577	0.530
	LN-PE	0.602	0.575	0.531
	MTE	0.630	**0.620**	0.563
	MSP + Selection	0.609 (+4.1%)	0.582 (+0.9%)	0.633 (+19.4%)
	LN-PE + Selection	**0.686** (+14.0%)	0.598 (+4.0%)	**0.665** (+25.2%)
	MTE + Selection	0.605 (-4.0%)	0.541 (-12.7%)	0.563 (0.0%)
	TokenSAR	0.645	0.596	0.553
	TokenCP (MSP)	0.601	0.567	0.529
	TokenCP (LN-PE)	0.635	0.566	0.556
	TokenCP (LN-PE) + TokenSAR	0.660	0.598	0.556
	TokenCP (MSP) + Selection	0.591 (-1.7%)	0.573 (+1.1%)	0.526 (-0.6%)
	TokenCP (LN-PE) + Selection	0.680 (+7.1%)	0.594(+4.9%)	0.570 (+2.5%)

In Table 6.4 it is possible to see some differences to Table 6.3. Firstly, **LN-PE + Selection** and **MTE** emerge as the best-performing UQ methods, each outperforming all other methods in 3 out of 6 cases. Secondly, the addition of token selection improved the performance of both MSP and LN-PE, supporting our earlier hypothesis. The relative difference to their non-selection counterparts, was on average +6.7% for MSP and +13.1% for LN-PE. We attribute the negative average relative difference observed for MTE (-3.2%) to the possibility that the selected tokens were not those where the model was most unsure about choosing a token – which is central to this UQ method.

6.7.1 Impact of token selection on TokenCP

In this section, we analyze the impact token selection can have on the performance of TokenCP. We begin this analysis by considering for *TokenCP versions* two possible cases: when there is no token selection (No) and when there is token selection (Yes). The results are presented in Table 6.5.

Table 6.5: Comparison of AUROC scores for TokenCP in regard to the use of token selection. The percentages show the relative difference to the corresponding *TokenCP version* with no selection.

Dataset	Method	Selection	Qwen2.5-7B	Llama-3.1-8B	Gemma-2-9B
TriviaQA	TokenCP (MSP)	No	0.652	**0.678**	0.562
	TokenCP (LNPE)		0.624	0.590	0.584
	TokenCP (MSP)	Yes	0.612 (-6.1%)	0.633 (-6.6%)	0.635 (+13.0%)
	TokenCP (LNPE)		**0.639** (+2.4%)	0.657 (+11.4%)	**0.637** (+9.1%)
NaturalQA	TokenCP (MSP)	No	0.601	0.567	0.529
	TokenCP (LNPE)		0.635	0.566	0.556
	TokenCP (MSP)	Yes	0.591 (-1.7%)	0.573 (+1.1%)	0.526 (-0.6%)
	TokenCP (LNPE)		**0.680** (+7.1%)	**0.594** (+4.9%)	**0.570** (+2.5%)

As it is possible to observe in Table 6.5, for the *TokenCP version*: **TokenCP + LN-PE**, the use of token selection improved its score on on all experiments – averaging a relative difference to no selection *TokenCP versions* of $+6.2\%$, making it the method with the highest performance in 5 out of 6 cases. In contrast, for the *TokenCP version*: TokenCP + MSP, we do not see general improvements as in the previous case, as the use of token selection often caused a decrease in the AUROC score (in 4 out of 6 times), averaging a relative difference to no selection *TokenCP versions* of -0.2%. In general, token selection enhances the performance of TokenCP, which aligns with our previous hypothesis. However this effect seems to be dependent on the UQ method combined with TokenCP.

We hypothesize that the different impact token selection has on these two *TokenCP versions*, is due to the changing in the answer length that token selection does (as it selects only the relevant tokens). As mentioned in Section 3.2, the scores from MSP are dependent on the length of the answer, whereas the LN-PE scores are not. As such the reduction in answer length may be having an impact on the performance of TokenCP + MSP.

6.8 UQ methods sensitivity to model or dataset changes

In this section, we study the robustness of the considered UQ methods in terms of model and datasets changes. We present Table 6.6 which contains the coefficient variation (CV) of each UQ method across the considered models, alongside the relative difference between using or not token selection.

Observing Table 6.6, it is possible to see that the CV values (considering both datasets) range from 0.70% to 8.61%, with an average of 4.92%. Indicating a reasonable robustness of all UQ methods across models and datasets. Focusing on the *TokenCP versions*, the mean CV across datasets and models is 5.00%. When token selection is applied, the *TokenCP versions* become substantially more robust,

showing an average relative difference of –33.84% compared with using no token selection. Moreover, token selection yields a consistent reduction in variability for all UQ methods on TriviaQA – yielding a mean relative difference of -70.48%, suggesting that only using relevant tokens produces more model-agnostic uncertainty estimates on this dataset. For NaturalQA the impact is mixed, averaging a relative difference increase of +2.8%. In this dataset, some methods show reduced CV (improved stability), while others show small increases in CV.

We hypothesize that since token selection restricts uncertainty estimates to the most informative tokens, it inherently reduces model-to-model differences. However, this stabilizing effect seems to depend on dataset characteristics (e.g. dataset complexity).

Table 6.6: AUROC variability per UQ method measured using Coefficient of Variation (CV). Lower values indicate more consistent performance across models. Percentages in red or green show the relative difference between the CV of a UQ method using selection relative to its counterpart not using. Percentages in green indicate that the UQ method became more stable in comparison to its counterpart.

Dataset	UQ Method	CV (%)
TriviaQA	MSP	8.61%
	LN-PE	4.99%
	MTE	8.03%
	MSP + Selection	0.70% (-91.9%)
	LN-PE + Selection	2.33%(-53.3%)
	MTE + Selection	2.45% (-69.5%)
	TokenSAR	6.33%
	TokenCP (MSP)	8.18%
	TokenCP (LN-PE)	3.05%
	TokenCP (LN-PE) + TokenSAR	5.84%
	TokenCP (MSP) + Selection	1.69% (-79.3%)
	TokenCP (LN-PE) + Selection	1.27% (-58.4%)
NaturalQA	MSP	4.31%
	LN-PE	5.28%
	MTE	4.86%
	MSP + Selection	3.49% (-19.0%)
	LN-PE + Selection	5.99% (+13.4%)
	MTE + Selection	4.66% (-4.1%)
	TokenSAR	6.08%
	TokenCP (MSP)	5.09%
	TokenCP (LN-PE)	5.68%
	TokenCP (LN-PE) + TokenSAR	7.07%
	TokenCP (MSP) + Selection	4.82% (-5.3%)
	TokenCP (LN-PE) + Selection	7.33% (+29.0%)

6.9 Final comments

In summary, **TokenCP** with the proposed configuration offers a resource-efficient, robust and adaptable conformal prediction framework for token-level uncertainty quantification, with a comparable performance to other UQ methods. Additionally, token selection emerges as a promising enhancement for both performance and robustness, across all conidered UQ methods including **TokenCP**, yet its reliability and dataset dependence requires further refinement.

Chapter 7

The exploration behind TokenCP

This chapter provides a detailed account of the exploration conducted prior to finalizing the TokenCP outline, as well as the insights gained from it.

7.1 Calibration datasets

The calibration dataset plays a crucial role in any conformal prediction procedure, so we dedicated part of the thesis specifically to decide which one to use. As specified in Section 4.1.3, TokenCP employs the bag-of-tokens calibration dataset. However, before adopting it as our default choice, we also considered other alternative options:

- **Sequential bag-of-tokens**: For this option, each calibration set would be made of softmax scores of tokens in the same phase of generation. For instance, when creating a prediction set for the first token generated in an answer, the sequential bag-of-tokens would be made of only the first tokens from the considered correct generations. Unfortunately, this option did not lead to an improved performance and it slowed down TokenCP (compared to the bag-of-tokens approach).

- **Syntax bag-of-tokens**: In this option, the prediction set for each generated token would be created using a calibration dataset composed of the softmax score of tokens with the same syntactic function as the one generated. However implementing this approach would be challenging, since: tokens could be a *subword* and there are many different syntatic functions – hence requiring many different calibration datasets with a substantial number of data points.

This experimentation showed that simpler approaches can sometimes outperform more complex ones while being more efficient and easier to implement. This insight, combined with the time constraints of this thesis, led us to settle on the bag-of-tokens calibration dataset and move on to other components of the pipeline. As discussed next, this decision refined and simplified our strategy for leveraging prediction set sizes for uncertainty quantification.

7.2 How to use prediction sets for UQ

As stated in Section 1.2, the main goal of this thesis was to find a way to use the split conformal prediction procedure for uncertainty quantification in NLG.

Ideally, a conformal prediction procedure should create prediction sets whose size reflects the model's uncertainty. We therefore decided to create prediction sets for tokens, meaning that each generated answer yields a list of prediction set sizes encapsulating the model's uncertainty in each token choice. From there, we chose to focus on sentence-level UQ rather than token-level UQ, as the former is more informative. This shift naturally raised the question of how to aggregate the list of token-level prediction set sizes into a single sentence-level uncertainty measurement.

For transforming the list of set sizes into a single value, we began by using simple summary statistics: mean, median, standard deviation, and skewness. As shown in Figure 7.1, there is no significant difference between the distributions of each class for any of these statistics, meaning that the resulting sentence-level UQ measure would not help distinguish between correct and incorrect answers. This outcome was consistent across all remaining models and datasets.

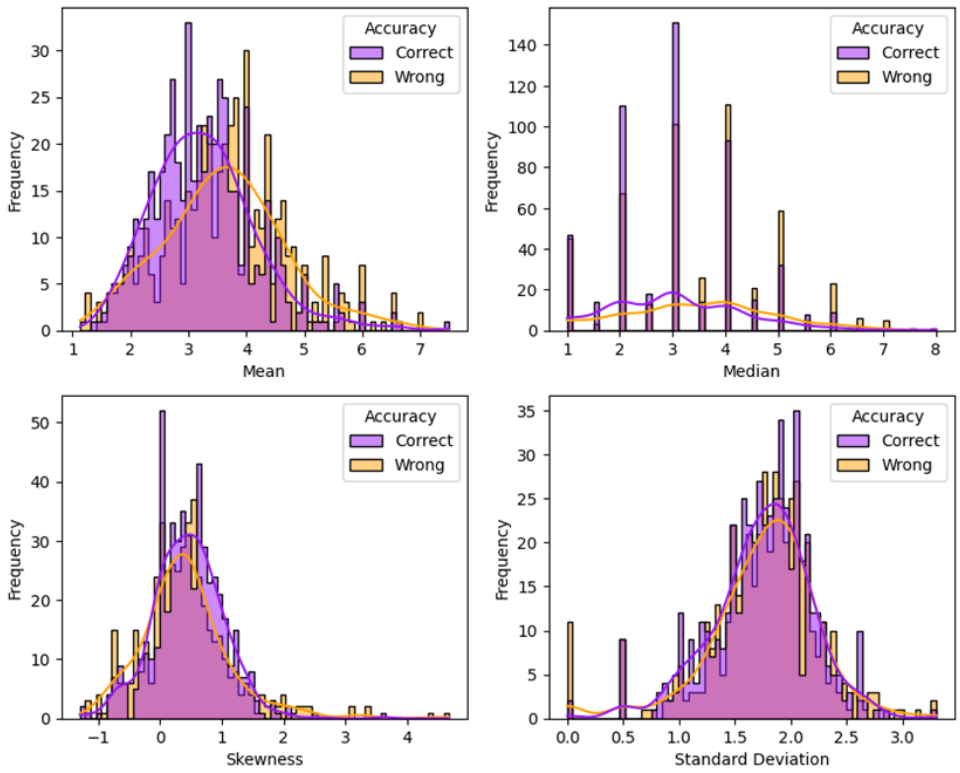


Figure 7.1: Histograms for the summary statistics, containing two classes: Correct answers and Wrong answers. The results plotted are for the Llama-3.1-8B model and the dataset TriviaQA.

Given this small difference between the distributions, we could have explored several avenues that could have lead to a bigger difference – for example, using a machine learning model to find the optimal combination of these four values. However, based on the previous insight, we instead decided to transform the list of set sizes into a list of values between 0 and 1, mimicking the softmax scores of tokens (as explained in Section 4.1.4). In this way, the final transformation from a list of values in $[0, 1]$ to a

single uncertainty measure can be handled by existing state-of-the-art UQ methods, since the list has *pseudo–softmax* scores. Therefore simplifying the overall process while still allowing easy refinement of the function that maps set sizes to values between 0 and 1. Additionally, visually analyzing several metric distributions proved difficult. It was hard both to identify differences per experiment and to compare several experiments against one another.

Chapter 8

Conclusions

Artificial intelligence has advanced significantly in recent years, unlocking a wide range of potential applications across diverse domains that do not share the same level of risk. In high-risk scenarios (such as the medical domain), a measure of an AI model's' outputs reliability is crucial for preventing bad decisions and undesirable consequences. Disappointingly, measuring the uncertainty of such models has been overlooked, and remains a relatively underdeveloped field.

This thesis aimed to contribute meaningfully to this field by examining existing solutions and introducing a new white-box uncertainty-quantification method, **TokenCP**, designed to integrate other approaches. This UQ method is based on the theoretically grounded and promising CP framework, offering new opportunities for advancing uncertainty estimation. **TokenCP** demonstrates flexibility and potential for improvement – through internal component optimization, integration with other state-of-the-art methods, and the use of token selection.

8.1 Contributions

The presented work contributes to its field in two main ways, which are presented next.

8.1.1 Insights drawn for UQ methods

In Chapter 6, some UQ methods besides TokenCP were evaluated. From this analysis, the following contributions were identified:

- **Performance:** Despite their straightforward implementation and computational efficiency, the evaluated UQ methods still demonstrated satisfactory performance. In particular, maximum token entropy (MTE) emerged as the most effective approach overall. This suggests that simple UQ methods can still provide meaningful uncertainty estimates.
- **Token Selection:** Similar to the observations made for TokenCP, restricting the uncertainty estimation to the most relevant tokens generally improved both the performance and robustness of the considered UQ methods.

8.1.2 TokenCP

In Section 1.2 we defined some objectives to guide the implementation of TokenCP. Next, we discuss how TokenCP relates to these objectives:

- **Be easily integrated into a wide range of LLMs.** Our implementation of TokenCP achieves this objective, as it can be applied to any LLM for which the logits are accessible. Furthermore, its optimal configuration (as shown in Chapter 6) results in faster generation and lower memory consumption compared to other configurations.

- **Have competitive performance with current state-of-the art approaches.** Although TokenCP is still in the early stages of development, the best-performing TokenCP version (TokenCP (LNPE) + TokenSAR) outperforms the evaluated UQ methods in one out of six cases, while achieving comparable performance in the remaining ones. Moreover, TokenCP shows significant potential for further improvement – for instance, incorporating token selection generally yielded performance gains (see Section 6.7 for details).

- **Maintain consistent performance across varying datasets and models.** Robustness is a well-known challenge for UQ methods in natural language generation. Nevertheless, the UQ methods considered in this work – including the TokenCP versions, exhibited reasonable robustness to both model and dataset variations (refer to Section 6.8). Additionally, incorporating token selection generally enhanced the stability of UQ methods under such changes.

Overall, TokenCP opens the door to integrating conformal prediction into uncertainty quantification for NLG. Even though its components are simple (e.g., the calibration dataset uses the bag-of-tokens approach), the prediction set sizes it produces show promising potential for effective uncertainty quantification. At the same time, it can be easily integrated into LLMs and remains reasonably robust across different models and datasets, showcasing its capability to being applied in real-world scenarios.

8.2 Future Work

The development of TokenCP opened many paths for possible improvements and left some questions unanswered. In this section, we discuss these in more detail.

8.2.1 Evaluating the CP procedure of TokenCP

In this work, the adaptiveness of the CP procedure employed in TokenCP was not evaluated, resulting in a lack of information regarding its implementation. As discussed in Section 4.1.5, the CP procedure used in TokenCP does not have a formal definition of coverage due to the nature of the natural language generation task. Consequently, the metrics introduced in Section 2.3.4 cannot be applied to assess its adaptiveness. Therefore, we propose an alternative evaluation approach, beyond the scope of this work, that would involve two steps:

1. **Defining the difficulty of a question.** In this first step, the difficulty of a question is assessed using a to be developed metric. This metric could be, for example, based on the Coleman-Liau Index (introduced by Coleman and Liau [64]) – which measures the U.S. grade level necessary to comprehend a text.

2. **Average prediction set size for the answer.** In the second step, the average of the prediction set sizes for all the tokens in the answer (to the previous question) is computed.

For our case, given these two steps, an adaptive CP procedure would generate on average larger prediction set sizes for harder inputs and smaller prediction set sizes for easier inputs.

8.2.2 Token Selection

As mentioned in Section 6.7, the token selection procedure sometimes fails to recognize any relevant tokens in an answer. In order to overcome this challenge, one could:

- Use a larger LLM or one with a larger token context window, to find the relevant tokens in an answer – in comparison to the one used in this work.

- Create guardrails that make sure that an LLM generates an answer with relevant tokens, for example by establishing a minimum number of strings for generation.

8.2.3 Calibration datasets

As discussed in Section 4.1.3, we considered only one calibration dataset for this work, the *bag-of-tokens* calibration dataset. An important direction for future work would be the development of new calibration datasets for TokenCP, as this could lead to an improvement on its performance.

One option would be to explore the use of token selection to create a calibration dataset (made of relevant tokens only) for the CP procedure of TokenCP. Nevertheless, this would prove challenging – unless using better performing models than those used in this work, as considering only the relevant tokens significantly reduces the number of data points one can get per correct answer (for the calibration dataset). Consequently increasing the number of questions needed for the LLM to answer, in order to create such calibration dataset.

8.2.4 Further evaluation of TokenCP

Finally, it is important to further evaluate the performance of TokenCP by:

- Using metrics beyond AUROC, such as the Pearson correlation coefficient.

- Use larger LLMs – those with more than 10B parameters, and a wider variety of tasks, including e.g., summarization and machine translation.

- Further investigation is needed to understand how *top-k* and the calibration dataset size affect TokenCP, over a wider range of values than those studied here. In particular, *top-k* values below 15 and above 90 should be explored, and calibration datasets should exceed 3,000 data points.

8.3 Final remarks

With the growing use of large language models it is essential to use these models with caution. That includes recognizing that they are not error-free and accounting for their uncertainty when generating outputs. This research work aims to promote more responsible use of large language models in real-world settings by introducing TokenCP – a promising new method for uncertainty quantification, by providing relevant insights into other state-of-the-art UQ approaches, and by motivating the adoption of theoretically grounded frameworks such as conformal prediction for reliable UQ in NLG.

Bibliography

[1] L. Qin, Q. Chen, X. Feng, Y. Wu, Y. Zhang, Y. Li, M. Li, W. Che, and P. S. Yu. Large language models meet nlp: A survey. *arXiv preprint arXiv:2405.12819*, 2024.

[2] T. Huang and Y. Wang. Can llms find fraudsters? multi-level llm enhanced graph fraud detection. *arXiv preprint arXiv:2507.11997*, 2025.

[3] D. P. Panagoulias, F. A. Palamidas, M. Virvou, and G. A. Tsihrintzis. Evaluating the potential of llms and chatgpt on medical diagnosis and treatment. In *2023 14th international conference on information, intelligence, systems & applications (IISA)*, pages 1–9. IEEE, 2023.

[4] S. Desai and G. Durrett. Calibration of pre-trained transformers. *arXiv preprint arXiv:2003.07892*, 2020.

[5] M. Shen, S. Das, K. Greenewald, P. Sattigeri, G. Wornell, and S. Ghosh. Thermometer: Towards universal calibration for large language models. *arXiv preprint arXiv:2403.08819*, 2024.

[6] C. Zhu, B. Xu, Q. Wang, Y. Zhang, and Z. Mao. On the calibration of large language models and alignment. *arXiv preprint arXiv:2311.13240*, 2023.

[7] R. Krishnan, P. Khanna, and O. Tickoo. Enhancing trust in large language models with uncertainty-aware fine-tuning. *arXiv preprint arXiv:2412.02904*, 2024.

[8] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[9] M. Campos, A. Farinhas, C. Zerva, M. A. Figueiredo, and A. F. Martins. Conformal prediction for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 12:1497–1516, 2024.

[10] A. N. Angelopoulos and S. Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[11] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[12] M. U. Hadi, Q. Al-Tashi, R. Qureshi, A. Shah, A. Muneer, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, M. A. Al-Garadi, et al. Llms: A comprehensive survey of applications, challenges, datasets, models, limitations, and future prospects.

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

[14] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[15] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[16] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[17] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[18] M. Hu, Z. Zhang, S. Zhao, M. Huang, and B. Wu. Uncertainty in natural language processing: Sources, quantification, and applications. *arXiv preprint arXiv:2306.04459*, 2023.

[19] J. Baan, N. Daheim, E. Ilia, D. Ulmer, H.-S. Li, R. Fernández, B. Plank, R. Sennrich, C. Zerva, and W. Aziz. Uncertainty in natural language generation: From theory to applications. *arXiv preprint arXiv:2307.15703*, 2023.

[20] J. Duan, H. Cheng, S. Wang, A. Zavalny, C. Wang, R. Xu, B. Kailkhura, and K. Xu. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5050–5063, 2024.

[21] L. Liu, Y. Pan, X. Li, and G. Chen. Uncertainty estimation and quantification for llms: A simple supervised approach. *arXiv preprint arXiv:2404.15993*, 2024.

[22] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W04-1013`.

[23] A. Vazhentsev, E. Fadeeva, R. Xing, A. Panchenko, P. Nakov, T. Baldwin, M. Panov, and A. Shelmanov. Unconditional truthfulness: Learning conditional dependency for uncertainty quantification of large language models. *arXiv preprint arXiv:2408.10692*, 2024.

[24] A. Malinin and M. Gales. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*, 2020.

[25] F. Ye, M. Yang, J. Pang, L. Wang, D. F. Wong, E. Yilmaz, S. Shi, and Z. Tu. Benchmarking llms via uncertainty quantification. *arXiv preprint arXiv:2401.12794*, 2024.

[26] L. Kuhn, Y. Gal, and S. Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

[27] V. Vovk. Conditional validity of inductive conformal predictors. In S. C. H. Hoi and W. Buntine, editors, *Proceedings of the Asian Conference on Machine Learning*, volume 25 of *Proceedings of Machine Learning Research*, pages 475–490, Singapore Management University, Singapore, 04–06 Nov 2012. PMLR. URL https://proceedings.mlr.press/v25/vovk12.html.

[28] R. F. Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani. Conformal prediction beyond exchangeability. *The Annals of Statistics*, 51(2):816–845, 2023.

[29] A. Angelopoulos, S. Bates, J. Malik, and M. I. Jordan. Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193*, 2020.

[30] L. Aichberger, K. Schweighofer, M. Ielanskyi, and S. Hochreiter. Semantically diverse language generation for uncertainty estimation in language models. *arXiv preprint arXiv:2406.04306*, 2024.

[31] Y. F. Bakman, D. N. Yaldiz, B. Buyukates, C. Tao, D. Dimitriadis, and S. Avestimehr. MARS: Meaning-aware response scoring for uncertainty estimation in generative LLMs. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7752–7767, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.419.

[32] E. Fadeeva, R. Vashurin, A. Tsvigun, A. Vazhentsev, S. Petrakov, K. Fedyanin, D. Vasilev, E. Goncharova, A. Panchenko, M. Panov, et al. Lm-polygraph: Uncertainty estimation for language models. *arXiv preprint arXiv:2311.07383*, 2023.

[33] O. Shorinwa, Z. Mei, J. Lidard, A. Z. Ren, and A. Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *ACM Computing Surveys*, 2025.

[34] L. Aichberger, K. Schweighofer, and S. Hochreiter. Rethinking uncertainty estimation in natural language generation. *arXiv preprint arXiv:2412.15176*, 2024.

[35] D. N. Yaldiz, Y. F. Bakman, B. Buyukates, C. Tao, A. Ramakrishna, D. Dimitriadis, J. Zhao, and S. Avestimehr. Do not design, learn: A trainable scoring function for uncertainty estimation in generative llms. *arXiv preprint arXiv:2406.11278*, 2024.

[36] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[37] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[38] S. Reddy, D. Chen, and C. D. Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

[39] D. Jin, E. Pan, N. Oufattole, W.-H. Weng, H. Fang, and P. Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

[40] P. Manakul, A. Liusie, and M. Gales. MQAG: Multiple-choice question answering and generation for assessing information consistency in summarization. In J. C. Park, Y. Arase, B. Hu, W. Lu, D. Wijaya, A. Purwarianti, and A. A. Krisnadhi, editors, *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 39–53, Nusa Dua, Bali, Nov. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.ijcnlp-main.4. URL https://aclanthology.org/2023.ijcnlp-main.4/.

[41] E. Fadeeva, A. Rubashevskii, A. Shelmanov, S. Petrakov, H. Li, H. Mubarak, E. Tsymbalov, G. Kuzmin, A. Panchenko, T. Baldwin, P. Nakov, and M. Panov. Fact-checking the output of large language models via token-level uncertainty quantification. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9367–9385, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.558. URL https://aclanthology.org/2024.findings-acl.558/.

[42] Y. A. Yadkori, I. Kuzborskij, A. György, and C. Szepesvári. To believe or not to believe your llm. *arXiv preprint arXiv:2406.02543*, 2024.

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

[44] R. Vashurin, E. Fadeeva, A. Vazhentsev, L. Rvanova, A. Tsvigun, D. Vasilev, R. Xing, A. B. Sadallah, K. Grishchenkov, S. Petrakov, et al. Benchmarking uncertainty quantification methods for large language models with lm-polygraph. *arXiv preprint arXiv:2406.15627*, 2024.

[45] A. Malinin, A. Ragni, K. Knill, and M. Gales. Incorporating uncertainty into deep learning for spoken language assessment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–50, 2017.

[46] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[47] Y. Xiao and W. Y. Wang. On hallucination and predictive uncertainty in conditional language generation. *arXiv preprint arXiv:2103.15025*, 2021.

[48] C. Zhang, F. Liu, M. Basaldella, and N. Collier. Luq: Long-text uncertainty quantification for llms. *arXiv preprint arXiv:2403.20279*, 2024.

[49] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[50] R. Vashurin, E. Fadeeva, A. Vazhentsev, L. Rvanova, D. Vasilev, A. Tsvigun, S. Petrakov, R. Xing, A. Sadallah, K. Grishchenkov, et al. Benchmarking uncertainty quantification methods for large language models with lm-polygraph. *Transactions of the Association for Computational Linguistics*, 13:220–248, 2025.

[51] M. Fomicheva, S. Sun, L. Yankovskaya, F. Blain, F. Guzmán, M. Fishel, N. Aletras, V. Chaudhary, and L. Specia. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555, 2020.

[52] A. Malinin and M. Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jN5y-zb5Q7m.

[53] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[54] D. Ulmer, C. Zerva, and A. F. Martins. Non-exchangeable conformal language generation with nearest neighbors. *arXiv preprint arXiv:2402.00707*, 2024.

[55] J. D. M.-W. C. Kenton and L. K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[56] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

[57] T. Team. The falcon 3 family of open models, December 2024.

[58] Qwen, :, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[59] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán,

F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardi, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo,

L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar, V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

[60] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[61] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[62] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[63] A. Nikitin, J. Kossen, Y. Gal, and P. Marttinen. Kernel language entropy: Fine-grained uncertainty quantification for llms from semantic similarities. *Advances in Neural Information Processing Systems*, 37:8901–8929, 2024.

[64] M. Coleman and T. L. Liau. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283, 1975.

Appendix A

Prompts

In this appendix we present all of the prompts used in this thesis. Every model that was used received one of these two prompts.

A.1 Answer generation prompt

```
Answer the following question in a clear and consice way, like a human would.
Avoid unnecessary details, repetition, or filler information.
\n Question: {question}
\nAnswer:
```

A.2 Token selection prompt

```
You are an AI evaluator.

Your task is to extract the exact portion of a response that directly
and specifically answers a given question.

### Instructions:
- Carefully read the question and the generated response.
- Identify the **minimal** span of text in the response that directly answers the question.
- Ignore surrounding context, introductions, or extra commentary.
- If the response is not answering the question, return None
- The output cannot have any punctuation

### Examples:

Example 1:
```

Question: What is the capital of France?
Response: France is a country in Europe. The capital of France is Paris. It is known for the Eiffel Tower.
Output: Paris

Example 2:
Question: Who wrote 'Hamlet'?
Response: 'Hamlet' was written by William Shakespeare, a famous playwright from England.
Output: William Shakespeare

Example 3:
Question: What is the speed of light?
Response: The speed of light in a vacuum is approximately 299,792 kilometers per second.
Output: 299,792 kilometers per second

Example 4:
Question: Who was the first king of Portugal?
Response: The third king of Portugal was D.João II.
Output: None

Example 5:
Question: In which European city was the 1968 Eurovision Song Contest held?
Response: The 1968 Eurovision Song Contest was held in the European city of Brighton, United Kingdom!
Output: Brighton

Now extract the answer:

Question: {question}
Response: {response}
Output: