Master Thesis

Développement d'une méthode de contrôle du texte généré par un modèle de langage à grande échelle (LLM) à partir de la représentation latente

---

Author:
Alba María Téllez Fernández

Directors:
PhD. Thomas Epalle

MAGIC LEMP

Supervisor:
Clement Royer

Université Paris Dauphine-PSL

MSc IASD: Artificial Intelligence, Systems, Data science

Université Paris Dauphine
Paris Sciences et Lettres, PSL

Course 2023-2024

Contents

Abstract

This project explores methods to enhance the reliability and accuracy of text generation by Large Language Models (LLMs) in sensitive domains such as healthcare and law, addressing the critical issue of "hallucination," where generated content deviates from factual information. We introduce Conformal Prediction (CP) as a framework for providing statistically reliable predictions without altering the model, and propose permutation-based strategies to mitigate the sensitivity of LLM outputs to input prompt structures. Additionally, we evaluate various decoding strategies, including a novel method called Decoding by Contrasting Layers (DoLa), to improve factual accuracy, and investigate CP without logit access (LofreeCP) for scenarios where model access is limited. Our findings suggest that while these methods can enhance prediction robustness, they may also have limitations in accuracy under certain conditions. We also propose future improvements, such as using convolutional layers for refining probability calculations. The efficacy of these approaches is demonstrated through legal document classification tasks, highlighting their potential to improve the performance and trustworthiness of AI systems in high-stakes environments.

1 Introduction

The state-of-the-art approach for various natural language processing (NLP) applications, particularly in document classification, involves training specialised models tailored for each specific task. For instance, Pretrained Language Models (PLM) like BERT [1](Devlin et al., 2019) are commonly employed, either by training the model from scratch or by fine-tuning on specific datasets to optimise performance within the target domain like done in Segonne et al. (2024) for many languages. However, such PLMs for specialised applications like legal BERT (Chalkidis et al., 2020), Juribert (Douka et al., 2021), legal CamemBERT (Louis and Spanakis, 2022) have a significant limitation: they lack the ability to generalise effectively to new categories or fields that they have not been exposed to during training. When confronted with unseen data or categories, traditional models often struggle to classify documents accurately because their performance is heavily dependent on the specific context in which they were trained. This reliance on prior knowledge limits their adaptability and makes them less suitable for dynamic or evolving datasets.

In the specific need for automatisation of document classification in the industry sector, the datasets that companies wish to classify we are working with are not symmetric and rather very specific; they do not have an equal distribution of documents across all categories. In such scenarios, training separate models for different categories or retraining models for each new classification task could be more efficient and may lead to biased or suboptimal performance. To overcome this limitation, we propose leveraging Large Language Models (LLMs) for document classification due to the success of the use of models such as GPT-4 (Brown et al. (2020), OpenAI (2022), OpenAI (2023)) in NLP tasks. Unlike traditional models, LLMs are capable of more nuanced reasoning and can dynamically adapt to a broader range of contexts. LLMs are pre-trained on vast and diverse corpora, enabling them to understand and classify documents from a wide variety of domains without the need for extensive task-specific fine-tuning. In addition, LLMs can flexibly handle varying input structures and

[1]Bidirectional Encoder Representations from Transformers Model: `https://huggingface.co/docs/transformers/model_doc/bert`

are not confined to fixed categories, we can achieve more robust and adaptable document classification across diverse datasets. This approach allows for better generalisation and a more accurate reflection of the underlying information within the documents, aligning with the need for models that can "think" more flexibly and adaptively.

Despite the impressive performance of recent LLM models, integrating LLMs into critical sectors like healthcare and law has been met with considerable apprehension, primarily due to concerns over trust, bias, and the potential for generating uncertain or misleading outputs. The challenge to companies when deploying LLMs in sensitive fields is multifaceted. On one hand, the accuracy and reliability of these models are paramount; errors in diagnosis or legal interpretation can have profound implications. On the other hand, the tendency of LLMs to "hallucinate"—a term coined to describe the generation of content that does not align with observed facts (Ji et al. (2024))—poses a significant obstacle to their widespread adoption. This phenomenon is particularly problematic in high-stakes environments, where the consequences of misinformation can be severe. In response to these challenges, this thesis introduces Conformal Prediction (CP) as a framework for enhancing the reliability of LLM outputs. CP is a statistical approach that provides a coverage guarantee for model predictions, ensuring that the actual outcome is included in the prediction set with a user-specified probability (Angelopoulos and Bates (2022)). This method is model-agnostic and distribution-free, making it suitable for any machine-learning model without needing modifications.

The significance of this research is underscored by the increasing reliance on AI in critical decision-making processes. In healthcare, for example, the ability to trust AI-generated diagnoses is essential for patient safety and treatment efficacy. Similarly, in legal settings, the veracity of AI-generated analyses is critical for the administration of justice. This thesis will delve into the application of CP within the context of LLMs, focusing on improving the interpretability of model confidence and providing a statistically reliable coverage guarantee for text generation. We will explore LLMs' sensitivity to input prompts' structure and investigate strategies to mitigate this, such as permutation-based approaches and advanced decoding strategies like Decoding by Contrasting Layers (DoLa). Moreover, we will examine the implications of LLM "hallucinations" in detail, considering the technical and ethical dimensions of this issue. Technical solutions will be proposed to minimise the risk of hallucinations. At the same time, ethical considerations will be addressed to ensure that deploying LLMs in sensitive domains does not infringe upon principles of fairness and accountability.

This thesis aims to bridge the gap between LLMs' potential and their responsible integration into critical applications. By enhancing the reliability and trustworthiness of LLM outputs, we hope to contribute to AI's safe and ethical advancement in sensitive domains, ensuring that the benefits of these technologies are realised without compromising on safety, reliability, and ethical standards.

2 Background

In this section, we aim to establish a foundational understanding of Language Models (LLMs) and provide an overview of decoding strategies, drawing upon the guidelines and using the notation offered in (Jurafsky and Martin, 2024) and (Labonne, 2023):

Tokenization Given an input piece of text (sentence, paragraph, document), i.e. *"I have a dream"*, we pass it into a **tokenizer**, an algorithm that breaks it down into smaller, more manageable units (words, subwords, characters or even bytes) called **tokens**. A tokenization scheme segments text into tokens using rules from a token learner, then maps these tokens to unique **token IDs** (numerical representations) based on a predefined vocabulary. Those sequences of N token IDs will be fed as an input vector $\mathbf{x} = (x_1, \ldots, x_{t-1})$ to the Language Model, where N is the **context length** in tokens (see Fig. 1).

Language Modeling is the task in natural language processing that involves predicting the next token or sequence of tokens in a given context or input sequence $\mathbf{x} \in \mathbb{R}^{t-1}$. To achieve this task, Language Models (LMs) generate scores assigned to every possible token in their vocabulary. We will call the unnormalized values in the final output vector $z^{[n]}$ the **logits**, the vector of scores. This output logits essentially represents a ranked list of potential next tokens in the sequence, so they are converted into probabilities using a softmax function. For example, the model assigns a probability of 17% to the token "of" being the next token after "I have a dream": $P(of|\text{I have a dream}) = 0.17$. This probability assignment provides a probability distribution over all the possible outputs, in this case, over the entire token vocabulary (see Fig. 1).

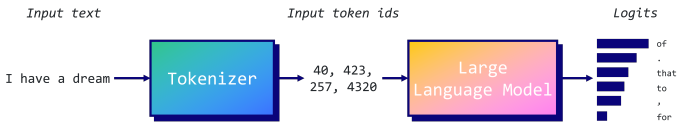


FIGURE 1: The diagram depicts how LLMs generate text by computing logits for all possible tokens. A tokenizer converts input text into token IDs, which the model (e.g., GPT-2) uses to predict the next token. Logits are then converted into probabilities via a softmax function. Retrieved from Labonne (2023).

Selecting a word to generate based on the model's probabilities is referred to as "Forward inference" or "Decoding" in a language model (refer to Appendix B for various decoding strategies). It's also possible to assign probabilities to entire sequences by breaking down the overall probability of tokens into the product of conditional probabilities of each subsequent token using the chain rule.:

$$p(\mathbf{x}) = p(x_1, \ldots, x_t) = \prod_{n=1}^{N} p(x_t \mid x_{<t}) \tag{2.1}$$

where x is a sequence of tokens, x_t is the t^{th} token, and $x_{<t}$ is the sequence of tokens preceding x_t, so that $p(x_t|x_{<t})$ represents the probability of a token x_t given all the previous tokens $x_{<t}$. Those probabilities are normalized such that for each time step t, the sum of

probabilities over all possible tokens x_t is 1:

$$\sum_{x_t} p(x_t|x_{<t}) = 1 \tag{2.2}$$

This approach of decoding from an LM to sequentially generate words by repeatedly sampling the next token using our previous choices as context is called **autoregressive language generation** or **causal LM generation**.

Each token in the sequence $\mathbf{x} = (x_i)_{i=1}^{t-1} \in \mathbb{R}^{d_{in} \times t-1}$ is first mapped to an index based on the vocabulary set $\mathcal{X}$, resulting in a sequence of N token IDs as the input to the Transformer Architecture. These token IDs are then used to retrieve **token embeddings** from an embedding matrix E of shape $[|V| \times d_{in}]$, where $|V|$ is the size of the vocabulary and d_{in} is the dimension of the embeddings. As a result, this initial token embedding provides an initial numerical representation of the token, capturing its meaning in a high-dimensional space. To incorporate the positional information of each token, **positional embeddings** are generated: vectors of dimension d_{in} that are specific to each position in the input sequence. These positional embeddings are added to the token embeddings to form combined embeddings for each position in the sequence, resulting in an input matrix $H_0 = \{h_1^{(0)}, \ldots, h_{t-1}^{(0)}\}$ of shape $[N \times d_{in}]$, where each row i represents the sum of the token embedding for the token ID at position i and its corresponding positional embedding. This matrix H_0 effectively captures both the identity and the positional information of each token, enabling the transformer to process the input sequence while maintaining the contextual relationships between tokens.

The standard neural algorithm that underlies most recent language model systems consists of an **embedding layer**, a **Transformer Architecture** (Vaswani et al., 2017) composed of N stacked transformer **blocks**, and an affine layer $\phi(\cdot)$ for predicting the next-token distribution. Each Transformer block is a multilayer network combining linear layers, feedforward networks, and self-attention mechanisms that allow the model to extract and utilize information from arbitrarily large contexts. The initial sequence of embedding vectors H_0 is iteratively processed by each transformer block, resulting in refined output vectors $H_j = \{h_1^{(j)}, \ldots, h_{t-1}^{(j)}\}$ at each j^{th} layer, of the input length.

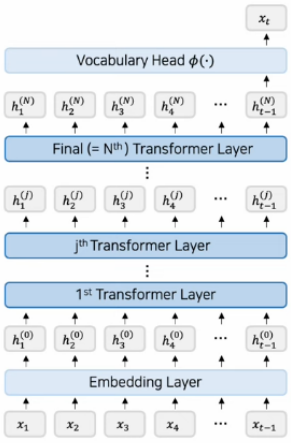


FIGURE 2: Hierarchical structure and flow of information through a Transformer model: the layered processing from initial embeddings to final token prediction.

In this approach, the token embeddings are gradually enhanced through residual connections and self-attention mechanisms, allowing the model to incorporate increasingly rich contextual information at each layer. This transforms them into complex, nuanced vectors that capture token meanings in context, enabling the transformer to build sophisticated representations essential for tasks like language modelling, translation, and question-answering.

Typically, in a transformer model, the hidden states of the final layer are passed through

the language head $\phi(\cdot)$, predicting the probability of the next token x_t over the vocabulary set $\mathcal{X}$,

$$p(x_t|x_{<t}) = \text{softmax}\phi(h_t^{(N)}))_{x_t}, \quad x_t \in \mathcal{X} \tag{2.3}$$

However, an alternative approach, known as **early-stopping**, is to apply ϕ directly to the hidden states of an intermediate layer j (where j is chosen from the set of early layers $\mathcal{J} \in \{0, \ldots, N-1\}$ instead of waiting until the final layer to compute the next-token probability:

$$q(x_t|x_{<t}) = \text{softmax}\phi(h_t^{(j)}))_{x_t}, \quad x_t \in \mathcal{X}, \quad j \in \mathcal{J} \tag{2.4}$$

It has been found to be effective even without any specialized training procedures. The reason behind this effectiveness lies in the residual connections within transformer layers, which allow hidden representations to evolve gradually rather than change abruptly. This means that even in the earlier layers, the model has already captured significant contextual information that can be used to make accurate predictions for the next token. This early exit method was first used in the field of computer vision, specifically in Convolutional Neural Networks (CNNs), and is also seen in techniques like knowledge distillation. As the name suggests, early exit involves using outputs from intermediate layers during the forward pass, not just the final layer's output. Tenney et al. (2019) demonstrated that transformer language models follow a hierarchical process in which earlier layers focus on encoding basic linguistic features, like part-of-speech tags, while more advanced semantic information is gradually integrated into the later layers. Building on this understanding, Dai et al. (2022) introduced the concept of "knowledge neurons" within the pre-trained BERT model. Their research revealed that these neurons are primarily concentrated in the topmost layers of the model, where they play a crucial role in capturing and representing complex, high-level knowledge. This suggests a layered approach to information processing within transformers, where different types of information are encoded at different depths in the model.

The self-attention mechanism plays a critical role in contextualizing token representations at each layer k by dynamically weighting and integrating information from neighboring tokens, based on their representations from the previous layer $k-1$. This process allows the model to generate nuanced, context-aware representations for each token, i.e. a contextualized meaning for each token within its specific context, which is essential for tasks such as language modeling, translation, and question-answering. In these layers, multiple self-attention heads operate in parallel at the same depth within the model, each focusing on different aspects or types of relationships within the input sequence. This allows the transformer to capture a broad range of dependencies and interactions simultaneously, leading to more robust and nuanced representations of the data. Unlike traditional models that separate encoding and decoding processes, transformers streamline this by using a unified self-attention mechanism that processes the entire input sequence at once. This approach enables the model to consider all tokens in the sequence in relation to each other, regardless of their positions, enhancing the contextual understanding of each token. As the data moves through the layers of the transformer, the self-attention mechanism continuously refines the representation of each token, effectively conveying richer and more context-aware information from lower to higher layers (Jurafsky and Martin, 2024).

2.1 Multiple Choice classification tasks

Transforming natural language processing (NLP) tasks into word prediction problems allows us to leverage the powerful capabilities of language models (LMs). This approach brings various NLP tasks together under a single framework, simplifying implementation and maximizing the utility of advanced LMs' capabilities in understanding and generating coherent text (Jurafsky and Martin, 2024). For example, we can reframe sentiment analysis by providing the model with a context such as "The sentiment of the sentence 'I like Jackie Chan' is:" and comparing the probabilities of the words "positive" and "negative" following this context. The LM then predicts the next most probable word, and the word with the higher probability indicates the sentiment, in what can be seen as a binary classification task.

Simple Label text classification can also be approached as word prediction by conditioning the model to make an option choice over the predefined categories on the prompt and then question what is the most relevant label for our text. Formally, let's denote our text T and our set of categories $c = \{c_1, c_2, \ldots, c_n\}$. Each c_i represents a distinct category relevant to the domain of the documents being classified. For example, in a medical context, categories might include "Cardiology," "Neurology," and "Oncology." To guide the language model to perform the classification, we construct a prompt that includes the list of categories and the document itself. The prompt is formulated to instruct the model to analyze the document and predict the most appropriate category. The general structure of the prompt P is tokenized using a function `tokenize()`:

$$P = tokenize(S(categories) + T + "Category:")$$

Where S(categories) is a serialized representation of the categories, typically formatted in a way that the model can understand (e.g., a numbered or bullet list; check **??** for a better explanation), and "Category:" is a textual cue indicating that the model should output one of the categories next. The sequence of tokens is passed through a large language model, which outputs a logit tensor with the shape $\text{logits} = [\text{batch\_size}, \text{sequence\_length}, \text{vocab\_size}]$, where **batch size** is the number of input sequences processed simultaneously, **sequence length** is the number of tokens in each input sequence, and the last component is the number of tokens in the model's vocabulary. This tensor represents the model's prediction for each token in the input sequence across the entire vocabulary. The classification is derived by analyzing the logits corresponding to the final token in the sequence, where the model is expected to output its category prediction. To isolate the model's output probabilities for the final token across the entire vocabulary:

$$last\_token\_logits = logits[:, -1, :]$$

To classify the document, we only need the logits corresponding to the specific tokens that represent our categories. Suppose the indices of the vocabulary tokens that correspond to the categories $c_1, c_2, \ldots, c_3$ are $i_1, i_2, \ldots, i_n$. We extract the logits corresponding to the

specific tokens representing our categories:

$$specific\_token\_logits = last\_token\_logits[:, [i_1, i_2, \ldots, i_n]]$$

To determine the predicted category, we apply a softmax function to the logits for the relevant category tokens. The softmax function converts the raw logits into probabilities that sum to 1, indicating the likelihood of each possible token being the correct category:

$$probabilities = softmax(specific\_token\_logits)$$

By doing so, we ensure that the classification is based solely on the model's confidence in the specific categories of interest. The index of the highest probability corresponds to the token predicted by the model. By mapping this token back to the original categories, we can determine the model's predicted category for the document.

On the other hand, **Multi-label text classification** refers to the task of tagging documents with relevant labels from a large label set, containing multiple labels (classes), i.e. more than one label can be selected for each document. To decide which labels to assign to a document, a threshold is often applied to the probabilities. A common threshold might be 0.5, meaning any label with a probability above 0.5 is assigned to the document. However, this threshold can be adjusted depending on the desired balance between precision (correctly predicted labels) and recall (completeness of predicted labels).

2.1.1 Evaluation Measures

To evaluate the performance of a multi-label text classification model, several evaluation metrics can be used, each providing different insights into the model's effectiveness. The most common measures are:

- **Accuracy**: In the context of multi-label classification, accuracy can be defined in several ways. One common approach is *subset accuracy*, which measures the proportion of instances for which all predicted labels exactly match the true set of labels. However, this measure can be overly strict in multi-label settings. Another approach is *example-based accuracy*, which calculates the average proportion of correctly predicted labels for each instance. Formally, for a set of instances, the accuracy is given by:

  $$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|}$$

  where Y_i is the set of true labels for instance i, $\hat{Y}_i$ is the set of predicted labels, and N is the total number of instances.

- **F1 Score**: The F1 Score is particularly useful for multi-label classification, as it balances both precision and recall. Precision measures the proportion of correctly predicted labels among all predicted labels, while recall measures the proportion of correctly predicted labels among all true labels. The F1 score, which is the harmonic mean of precision and recall, can be computed for each label and then averaged, either using a micro-average (aggregating contributions of all labels to compute the average

metric) or a macro-average (computing the metric independently for each label and then taking the average). The F1 Score is defined as:

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where:

$$\text{Precision} = \frac{|Y_i \cap \hat{Y}_i|}{|\hat{Y}_i|}, \quad \text{Recall} = \frac{|Y_i \cap \hat{Y}_i|}{|Y_i|}$$

Beyond these standard evaluation metrics, it is crucial to account for the uncertainty of the model's predictions, particularly in scenarios where decisions have significant consequences.

To quantify the model's uncertainty over the predicted output, we will employ **conformal prediction**, a framework that provides a measure of confidence in the predictions by creating prediction sets that are guaranteed to contain the true labels with a specified probability (see Section 2.2 for further details). Conformal prediction adjusts the confidence level dynamically based on the distribution of the prediction scores, offering a more robust understanding of model reliability in multi-label contexts.

2.2 Conformal Prediction (CP) with logit access

This approach paves the way for more trustworthy applications of machine learning models in critical domains such as healthcare, as demonstrated in recent studies like those by (Kumar et al., 2022).

Conformal Prediction (CP) (Gammerman et al., 2013) (Vovk et al., 2022) is a statistical framework designed to address these issues by quantifying the uncertainty in model predictions by generating prediction sets. These sets are constructed to contain the true label or correct output with a user-specified probability, such as 95% confidence. A key advantage of Conformal Prediction is model-agnostic, meaning it can be applied to any model—statistical, machine learning, or deep learning—without requiring modifications to the model or its training process. Also, CP is distribution-free, meaning it does not rely on assumptions about the data distribution. This sets it apart from other uncertainty estimation A key limitation of traditional machine learning is the need for more reasonable confidence measures for individual predictions. Models may have excellent overall performance, but not be able to quantify uncertainty for a given input reliably. Conformal prediction solves this by outputting prediction regions and confidence measures with statistical validity guarantees. This method provides reliability without relying on assumptions about data distribution or altering the base model. It offers explicit, non-asymptotic guarantees that hold regardless of the distribution or model assumptions. Conformal prediction can be applied to any pre-trained model, including neural networks, to generate prediction sets that are guaranteed to include the true outcome with a probability defined by the user.

Drawing upon previous works done by Angelopoulos and Bates (2022), Manokhin (2023) and Kumar et al. (2023), we introduce a formalism for Conformal Prediction as a method to generate prediction sets. The process of conformal prediction begins with a trained model, such as a neural network classifier, which we'll denote as $\hat{f}$. The goal is to generate

prediction sets—groups of possible labels for new instances—that are reliably accurate. To achieve this, we use a small, separate set of data, known as calibration data. This calibration step fine-tunes the prediction sets, ensuring that they meet a predefined confidence level, meaning that the true label will be included in the set with the specified probability. This method adds a layer of reliability to the model's predictions without altering the model itself.

In a multi-label classification setting, our goal is to predict the appropriate label(s) from our set of labels $Y_i \subseteq \{1, \ldots, K\}$ for a given document X_i, where K represents the total number of possible classes. Our classifier, a pretrained model $\hat{f}$, outputs the estimated probabilities (softmax scores) for each of the K classes, denoted as $\hat{f}(X) \in [0,1]^K$. These softmax scores are intended to represent the conditional probabilities of each class y on an input text x: $\mathbb{P}(Y = y|X = x)$.

However, these predicted probabilities may not always be reliable due to potential overfitting or other sources of bias and errors. To address these issues and improve the trustworthiness of the predicted probabilities, we introduce a calibration set $\{(X_j, Y_j)\}_{j=1}^m$ consisting of unseen, independently, and identically distributed (i.i.d.) pairs of texts and labels. This calibration set helps adjust the softmax outputs of our model, correcting any inaccuracies and ensuring that the predicted probabilities better reflect the true likelihood of each class given the input.

Using the pretrained model $\hat{f}$ and the calibration set, our objective is to identify the possible labels for a new test document X_{test} in such a way that most of the true labels are included in the result. To do this, we apply a threshold to the model's outputs to determine the subset of the K classes that the model considers most likely.

This **prediction set** $\mathcal{C}(X_{test})_{1-\alpha} \subseteq \{1, \ldots, K\}$ represents a subset of possible labels that we believe, with high confidence, could be the correct labels for X_{test}. The prediction set is designed to capture the true label(s) with a specified confidence level of $1 - \alpha$, being $\alpha \in (0, 1)$ the significance level that represents the probability of the prediction set not containing the true outcome. For example, if $\alpha = 0.1$, the prediction set is constructed to contain the correct label(s) with at least 90% probability. The chosen α directly influences the size of the prediction set.

Property 2.1 (Nesting Property)**.** *Smaller α corresponds in larger confidence levels that lead to larger prediction sets that contain the sets generated at smaller confidence levels. Larger prediction sets reflects more uncertainty, increasing the likelihood that the true label is within the set.*

$$\alpha_1 > \alpha_2 \Rightarrow \mathcal{C}_{1-\alpha_1}(X) \subseteq \mathcal{C}_{1-\alpha_2}(X). \tag{2.5}$$

By adjusting, α we control the trade-off between the size of the prediction set and the level of confidence that it contains the true outcome.

Theorem 2.2 (Conformal coverage guarantee)**.** *Given a user defined desired error rate $\alpha \in (0, 1)$, suppose that the calibration set $(X_i, Y_i)_{i=1\ldots n}$ and the test set as (X_{test}, Y_{test}) are independent and identically distributed (i.i.d.). C is the calibrated prediction set introduced above. $(X_{test}, Y_{test}) \approx D_{calibration}$ is an unseen test point that is drawn from the same distribution as the data used to calibrate the prediction sets. $\mathcal{C}_{1-\alpha}(X_{test})$ is a set-valued*

mapping satisfying the nesting property in Eq. 2.5. Then the following holds:

$$1 - \alpha \leq \mathbb{P}(Y_{test} \in \mathcal{C}_{1-\alpha}(X_{test})) \tag{2.6}$$

To construct the set-valued mapping $\mathcal{C}_{1-\alpha}(X) : \mathcal{X} \longrightarrow [0,1]^{\mathcal{Y}}$ using the pretrained model $\hat{f}$ and the calibration set, we first define a **nonconformity score** function $N : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. This function quantifies the degree of uncertainty or "nonconformity" of a model's prediction by measuring how much a new instance $(x, y) \in \mathcal{X} \times \mathcal{Y}$ deviates from the patterns observed in the calibration set. The nonconformity score is designed to reflect the confidence (or lack thereof) the model has in its predictions. It is chosen based on the specific machine learning task—whether it is classification, regression, or time series forecasting—and the properties of the underlying model $f : \mathcal{X} \to \mathcal{Y}$. For classification tasks, for each instance X_i and corresponding true class Y_i, is it common to define the nonconformity score, as used in least ambiguous set-valued classifiers (LAC) (Sadinle et al., 2019):

$$N(X_i, Y_i) = 1 - \hat{f}(X_i)_{Y_i} \in \mathbb{R}$$

where $\hat{f}(X_i)_{Y_i}$ represents the model's softmax score for the true label Y_i given the input X_i. This score measures how "unusual" the classifier's prediction is. A high nonconformity score signifies a large deviation from what the model considers a typical or conforming output, suggesting uncertainty or a possible error in the prediction. Conversely, a low nonconformity score indicates that the model is confident about the true label.

In order to calibrate the prediction sets to a desired coverage leve $1 - \alpha$, we estimate a threshold $\hat{q}_\alpha$ which corresponds to the $1 - \alpha$ quantile of the nonconformity scores on a calibration set. Formally, this threshold is computed as:

$$\hat{q}_\alpha = \text{Quantile}\left(\{s_1, \ldots, s_n\}, \left\lceil \frac{(n+1)(1-\alpha)}{n} \right\rceil\right), \tag{2.7}$$

where $\{s_1, \ldots, s_n\}$ are the nonconformity scores obtained from the calibration set.

At inference time, the prediction set for a new input test data point (where X_{test} is known yet, its corresponding label Y_{test} is unknown), we use the nonconformity scores to determine which labels are the most plausible given the model's predictions. The prediction set is defined as:

$$C(X_{test}) = \{y \in \mathcal{Y} : N(X_{test}, y) \leq \hat{q}_\alpha\} = \{y \in \mathcal{Y} : \hat{f}(X_{test})_y \geq 1 - \hat{q}_\alpha\} \tag{2.8}$$

This prediction set includes all classes for which the model's softmax output is high enough, specifically, those for which the softmax score is at least $1-\hat{q}_\alpha$. The assumption of exchangeability between the calibration data and the test data is crucial here. Exchangeability implies that the calibration set and the test set are drawn from the same underlying distribution and are statistically indistinguishable. This assumption ensures that the coverage guarantee $1 - \alpha$ holds, meaning that the prediction set $C(X_{test})$ will contain the true label with a probability of at least $1 - \alpha$.

2.2.1 Example: Conformal Calibration in Multiclass Classification

Let's provide an example of a multiclass setting where the label space $\mathcal{Y}$ consists of more than two classes. Consider a classifier $f : \mathcal{X} \to \Delta^{|\mathcal{Y}|}$ predicting categories $\mathcal{Y} = \{A, B, C\}$. Given a test document X_{test}, the classifier outputs softmax scores:

$$f(X_{\text{test}}) = (0.7, 0.2, 0.1).$$

The nonconformity scores are computed as:

$$S(X_{\text{test}}, A) = 1 - 0.7 = 0.3 \quad S(X_{\text{test}}, B) = 0.8 \quad S(X_{\text{test}}, C) = 0.9$$

Assume a calibration set of $n = 50$ documents with nonconformity scores $\{s_1, s_2, \ldots, s_{50}\}$. To achieve a 90% coverage ($\alpha = 0.1$), compute the quantile threshold:

$$\hat{q}_\alpha = \text{Quantile}\left(\{s_1, \ldots, s_{50}\}, \left\lceil \frac{51 \times 0.9}{50} \right\rceil\right) = s_{46} = 0.75.$$

The prediction set $C(X_{\text{test}})$ includes all categories where $S(X_{\text{test}}, y) \leq \hat{q}_\alpha$:

$$C(X_{\text{test}}) = \{A\}, \quad \text{since } 0.3 \leq 0.75 \text{ and } 0.8, 0.9 > 0.75.$$

This set $C(X_{\text{test}}) = \{A\}$ has a 90% coverage guarantee, meaning the true label will be in this set with at least 90% probability.

3 Domain specific Supervised Multiple Choice classification tasks

In our project, we focus on **legal text processing**, an emerging field in NLP with many applications but limited publicly available resources (Chalkidis et al., 2019). Accurate classification is necessary due to the complexity of the legal field in order to enable efficient legal research and analysis. In order to improve the overall usability and accessibility of the legal texts, we want to make sure that every document is appropriately classified by aligning the dataset with these preset categories. The approach to category selection varies across projects, depending on factors such as the availability of predefined categories and the nature of the dataset.

French Legal Database. We utilize a dataset of legal documents from French jurisprudence, provided in an encoded form to preserve privacy. The dataset consists of approximately 120 French legal documents supplied by a non-disclosed legal-tech source, including a variety of legal texts, such as case law, legal updates, and collective agreements. These documents are tagged with themes and subthemes in accordance with a hierarchical structure developed by subject-matter experts. The tagging process involves assigning categories, subcategories, and detailed explanations to aid the language model in achieving accurate

Content	SubThemes
"Se prévalant de ce que les troubles d'infertilité dont elle est atteinte étaient en relation directe avec la prise de distilbène par sa mère au cours de la grossesse dont elle est issue, ..."	• Droit des obligations • Droit de la sécurité sociale • Droit médical • Responsabilité civile
"... sur la demande de ... en paiement de diverses sommes et indemnités pour l'exécution et la rupture de son contrat de travail, dirigée contre la ..."	• Droit du travail
"En qualité de président directeur général de ..., nommé à cette fonction le ..., Monsieur ... percevait diverses rémunérations, et bénéficiait en outre d'un plan d'attribution d'actions ..."	• Droit commercial • Droit du travail

TABLE 1: Example of the expected text subcategories classification output

comprehension (see Appendix A, Table 8). The number of labels assigned to each document varies, with some documents remaining unclassified, while others are tagged with multiple themes and subthemes, as seen in Table 1. This variability makes the classification process more complex than straightforward categorization based solely on specific actions or content. As shown in Figure 3, the labels in this dataset are irregularly assigned, resulting in uneven sample sizes across different labels, both in single-labeled and multi-labeled cases.

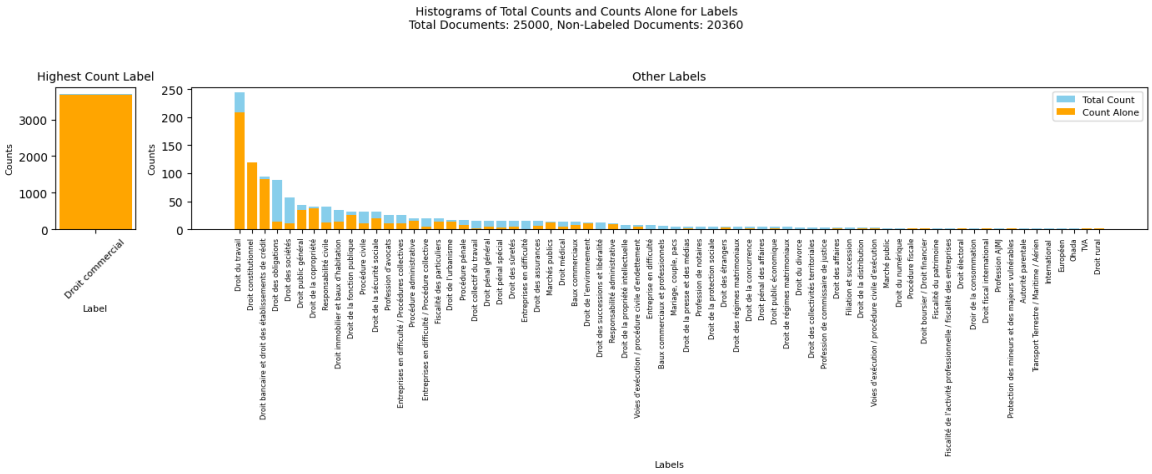


FIGURE 3: The left histogram shows the total counts for the most frequent label ("Droit commercial") across all documents. The right histogram displays the counts for all other labels. The number of times these labels appear on their own (single-labeled documents) is also included. Y-axes are independently scaled to enhance visualization. The analysis includes 25,000 documents, most of which are not labeled.

This irregular label assignment—where some labels are assigned very frequently while others

are rare—poses a challenge for traditional machine learning models due to class imbalance. Such imbalance can cause models to become biased toward the more frequent labels, reducing their overall performance. Zero-shot prompting offers a compelling alternative in this situation, as zero-shot models can leverage their generalized knowledge to make predictions without being affected by the uneven distribution of training samples. This approach makes zero-shot prompting a more suitable method for handling the variability and irregularity present in this classification task.

Calibration Dataset. To achieve our research objectives, we created a sample dataset by carefully selecting and balancing legal documents based on their themes. We specifically chose documents focused on "Droit du travail" (Labor Law) that do not mention "Droit commercial" (Commercial Law) to maintain a clear distinction between these legal domains. Additionally, we included an equal number of documents exclusively labeled as "Droit commercial" to ensure a balanced representation of both themes. This approach allowed us to create a well-rounded dataset that is both focused and unbiased, ideal for further analysis. In total, we identified 241 documents with "Droit du travail" but not "Droit commercial," and 3,669 documents with only "Droit commercial." From these, we randomly selected 241 documents from each category, resulting in an evenly balanced dataset. Furthermore, we prepare a test dataset where we included documents containing both labels to test the performance for multi-label classification tasks.

Models. In this project we will be using the models `microsoft/Phi-3-mini-4k-instruct`[2] and `meta-llama/Meta-Llama-3.1-8B-instruct`[3]. For each multiple-choice answer, we generate the corresponding logit by performing single-token prediction. These logits are then normalized using softmax to obtain valid probabilities for each option. This allows us to identify the most likely correct answer by selecting the option with the highest probability.

3.1 In-context learning and Prompt Engineering

Large Language Models (LLMs) are highly sensitive to the exact input prompt, which has led to the development of a field focused on **in-context learning** and prompt engineering, or prompt tuning (Zhou et al., 2023; Wei et al., 2023). In-context learning refers to the ability of LLMs to understand and make predictions based on the context in which the input data is presented, without updating the model weights. Prompt engineering methods vary significantly across tasks and often require extensive experimentation and reliance on handcrafted heuristics. For the current setup, model performance on classification tasks is often sensitive to the prompts used. Therefore, we experiment with several prompting strategies before finalizing the approach.

The classification task begins by constructing a system prompt that defines the categories for the model. For each domain, we create prompts tailored specifically to that field. In the context of jurisprudence, when generating predictions for the classification of legal document T, we instruct the model to assume the role of a "legal French documents expert." The prompt includes a predefined list of relevant categories, such as c_1 = "Droit commercial" (Commercial Law), c_2 = "Droit du travail" (Labor Law), and c_3 = "None" (Documents that

[2]Microsoft. Phi-3-mini-4k-instruct. `https://huggingface.co/microsoft/Phi-3-mini-4k-instruct`
[3]Meta-Llama-3.1-8B-instruct. `https://huggingface.co/meta-llama/Meta-Llama-3.1-8B-instruct`

do not fit into the specified legal categories). This directs the model to choose the most appropriate category after analyzing the provided legal text.

Zero-shot prompting In our experiments, we employ zero-shot prompting, where the model makes predictions without being provided with specific examples beforehand. In this prompt, we use a technique similar to "in-context learning," where the model is given relevant contextual information about the categories, including their names and descriptions, but without specific labeled examples. By supplying clear and concise descriptions of each category, the model leverages its internal knowledge and reasoning capabilities to infer the most appropriate category for a given legal text T. This approach enables the model to perform the classification task by understanding the context and meaning of the provided categories, allowing for accurate zero-shot predictions without explicit training examples in the target domain.

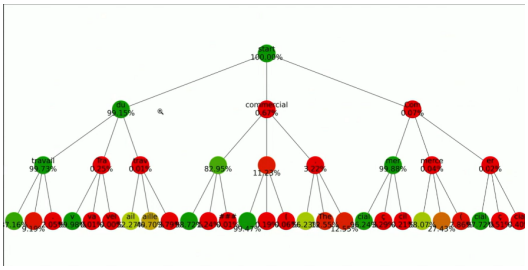
A sample prompt S that instructs the model to classify the given legal text T into one of the predefined categories might look like this:

```
You are a specialized model for classifying legal French documents.  You are
given the following list of categories:   # Categories:

  1. Commercial Law:  Legal matters related to commercial transactions and
     business law

  2. Work Law:  Legal matters related to labor laws and employment contracts

  3. None:  Documents that do not fit into the specified legal categories

The user will send you a text.  Your answer should be the number and name of
the category.
```

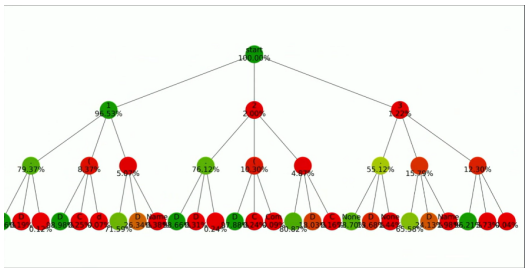
3.1.1 Importance of Enumeration in classification

In order too understand the rationale behing the enumerqtion of categories when we pass them to the LLM, we show an illustration of that using beam search as a decoding strategy in our prompt. Our LLM is going to try to generate the next n tokens that are supposed to be one of our categories. Greedy search is the strategy that considers the most probable token at the time of prediction; while beam search takes into account the n most likeley tokens, where n is the number of beams. Then, the sequence (or beam) zith the highest overqll score is chosen as the output. Here $P(w)$ is the cumulative sum of the lof probabilty of every token in the beam: We normalize this score by the seauence lenght to prevent bias towrd longer sequences. Given our prompt, we are going to generate three additional tokens.

When applying LLMs to classification tasks, we often need the model to generate tokens that correspond to specific categories. However, a challenge arises when these categories have varying token lengths or share identical initial tokens. For example, if the categories are "cat," "car," and "can," the first token ("c") is the same for all three, making it harder for the model to differentiate between them immediately. As we saw in the Figure 4b, the first token that generates is a number for the classification task and is very confident about the correct responses since the number, however, in Figure 4a, the LLM tries to do this



(a) Beamsearch prompt with no number categories



(b) Beamsearch prompt with number categories

FIGURE 4: Beamsearch prompt comparison: (a) without number categories and (b) with number categories. In both graphs; the top node stores the input tokens, so it has a 100% of probability, while the other nodes represent the token generated nodes. we computed the score for $beams^{lenght} = 3^3$ possible sequences. Each node as marked the sequence score, which represents the score of the sequence up to that point. Both cases try to generate Droit du travail, which is the expected output.

diferentiation itself in not such a efficient way.

Therefore, our logic of expecting the LLM to write the whole category name c_1, c_2, c_3. for classification, can adapted to just consider the specific token logits that correspond to the classification category indices i_1, i_2, i_3. The indices i_1, i_2, i_3. of these tokens within the model's vocabulary are identified (e.g. for tokenizer "microsoft/Phi-3-mini-4k-instruct", 29896 for "1", 29906 for "2", and 29941 for "3"). This yields a tensor of shape `[batch size, 3]`, representing the model's raw confidence scores for each category. These logits are then converted into probabilities using the softmax function.

3.1.2 Sensitivity of Model Outputs to Prompt Structure

When using a language model (LLM) as a classifier, one expects consistent and reliable outputs that accurately reflect the input data. Ideally, these models should generate outputs that map inputs into a well-structured probability space, clearly indicating the model's confidence in each category. However, our findings reveal that LLM classifiers are highly sensitive to the structure of the prompt, including the order of input features and class labels. Even minor changes, such as swapping the order of two categories, can significantly affect the model's classification performance, as demonstrated in Figures 5a and 5b. This sensitivity suggests that the model's output is influenced not just by the content of the prompt but also by how the content is structured, raising concerns about the reliability and stability of the model's predictions.

In multiclass classification, one of the primary challenges is to construct prediction sets that are appropriately sized—not too large to include too many potential labels, nor too small to exclude the true label. To address this challenge and improve the robustness of prediction sets, we propose a permutation-based strategy within the Conformal Prediction (CP) framework. This approach is particularly useful for LLMs, where model outputs are sensitive to the order of input features and class labels. For example, changing the order in which class labels $y \in \mathcal{Y}$ are presented in a prompt can influence the output probabilities of the model.

To investigate this sensitivity systematically, we analyzed the final probabilities produced by the model across various prompt configurations. By permuting the order of categories in each prompt, we quantified the extent to which these changes affect the model's outputs. In Table 2 and 3 we provide different results based on the input order for the text processed by the "microsoft/Phi-3-mini-4k-instruct" model.

Method	**microsoft/Phi-3-mini-4k-instruct**		
<|**system**|>	You are a model that classifies legal French documents. You are given the following list of categories:	You are a specialized model for classifying legal French documents. You are given the following list of categories:	You are a model for classifying legal documents in French. The available categories are listed below:
<|**system**|>	# Categories: 1. Droit commercial 2. **Droit du travail** 3. None		
<|**system**|>	The user will send you a text. Your answer should be the number and name of the category.	The user will provide a legal text. Respond with the corresponding number and name of the category.	After receiving the legal text from the user, respond with the relevant category number and name.
<|**user**|>	FAITS PROCÉDURE MOYENS ET PRÉTENTIONS DES PARTIES...		
<|**assistant**|>	Category:		
Specific tokens Logits	[19.7969, **21.8125**, 18.8125]	[21.2344, **23.3281**, 19.5156]	[17.0625, **18.3281**, 15.0938]
Softmax Logit Score $f(X)$	[0.1116, **0.8379**, 0.0417]	[0.1065, **0.8643**, 0.0191]	[0.2117, **0.7506**, 0.0296]

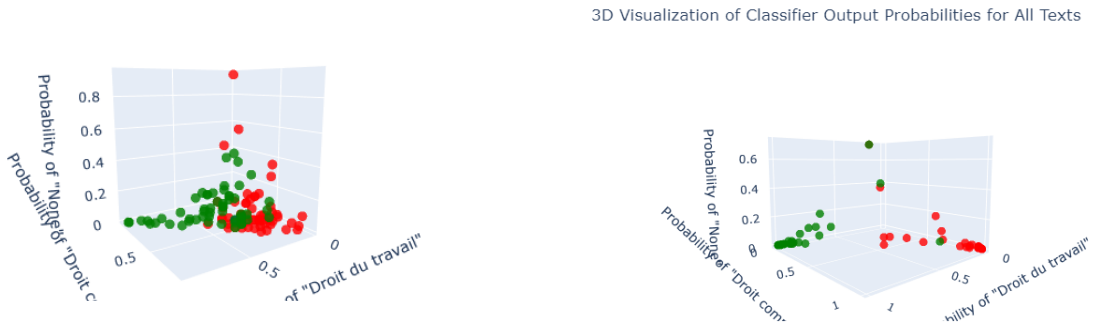
TABLE 2: Influence of Prompt Wording on Model Output in a Zero-Shot Setting Using "microsoft/Phi-3-mini-4k-instruct." This table shows the variation in output logits and softmax probabilities for the "Droit du travail" category based on three different prompt wordings provided to the model. Each column represents a different formulation of the prompt text, highlighting how the choice of prompt affects the model's prediction probabilities. The logits and softmax scores indicate that the likelihood of predicting "Droit du travail" changes with the prompt, showcasing the sensitivity of the model's output to prompt phrasing.

The analysis of these tables reveals that, although the "microsoft/Phi-3-mini-4k-instruct" model consistently classified the text into the correct category, "Droit du travail," the confidence levels of these predictions varied significantly depending on the prompt formulation and the order in which the class labels were presented. This variability underscores the sensitivity of large language models (LLMs) to both the wording and structure of input prompts. While the true label was always included in the prediction set, the fluctuating softmax scores indicate that the model's perceived certainty is highly susceptible to prompt and label order manipulation.

Method	**microsoft/Phi-3-mini-4k-instruct**		
<|**system**|>	You are a model that classifies legal French documents. You are given the following list of categories: # Categories:		
<|**system**|>	1. **Droit du travail** 2. Droit commercial 3. None	1. Droit commercial 2. **Droit du travail** 3. None	1. Droit commercial 2. None 3. **Droit du travail**
<|**system**|>	The user will send you a text. Your answer should be the number and name of the category.		
<|**user**|>	FAITS PROCÉDURE MOYENS ET PRÉTENTIONS DES PARTIES ...		
<|**assistant**|>	Category:		
Logits for specific tokens	[**25.9531**, 23.4531, 23.0000]	[19.7969, **21.8125**, 18.8125]	[22.2969, 21.5625, **25.3906**]
Softmax Logit Score $f(X)$	[**0.8757**, 0.0719, 0.0457]	[0.1116, **0.8379**, 0.0417]	[0.0423, 0.0203, **0.9330**]

TABLE 3: Impact of Class Label Ordering on Model Output in a Zero-Shot Setting Using "microsoft/Phi-3-mini-4k-instruct." This table illustrates the model's sensitivity to the order of class labels presented in the prompt. Three different label orders were tested: placing "Droit du travail" first, second, or third. The logits and resulting softmax probabilities for each specific label order indicate substantial variability in the predicted likelihoods. This demonstrates the significant influence that the order of input labels has on the model's predictions, with the highest confidence consistently aligning with the positional bias of the category within the list.

Figure 5 presents a 3D visualization of the output probabilities generated by a classifier when processing multiple texts. This visualization aims to illustrate the sensitivity of the model to different prompt configurations. In subfigure 5a, the output probabilities are shown for a specific order of input features and class labels, while subfigure 5b displays the output probabilities when the prompt order is altered. The comparison between these two subfigures provides insight into how the classifier's output distribution changes in response to variations in prompt structure.



(a) B

(b) B

FIGURE 5: 3D visualization of classifier output probabilities for all texts. Subfigure (a) demonstrates the output probabilities when the prompt order is structured in one configuration, while subfigure (b) shows the output probabilities with a different prompt order. The variation in the distribution of points between (a) and (b) illustrates the model's sensitivity to the order of input features and class labels, highlighting how minor changes in prompt structure can significantly influence the classification results.

These findings emphasize the importance of carefully designing prompts and considering label order when utilizing LLMs for classification tasks, to ensure robust and reliable prediction outcomes. Given the model's sensitivity to prompt structure, as discussed, it is crucial to select an effective method for predicting the final category for each document. To mitigate the variance in model predictions caused by prompt variability, for each test input X_{test}, we suggest generating multiple permutations of the input, such as different orderings of its features or controlled modifications that preserve the label. Additionally, generating permutations of the class labels $\mathcal{Y}$ can help mitigate the influence of class order on nonconformity scores and prediction outcomes.

In our methodology, we employed an ensemble approach that involved generating prompts using all possible permutations of three distinct categories within each prompt structure. This process resulted in $3! \cdot 3 = 18$ unique prompts for each document, covering all permutations of the three categories. For each of these 18 prompts, we obtained a softmax probability output, capturing the model's response to each distinct permutation. The final probability distribution for each document was then calculated by averaging the softmax outputs across all 18 prompts. This comprehensive ensemble strategy reduces prediction variance and enhances model robustness by incorporating a broad spectrum of model outputs, thereby minimizing the impact of any individual prompt's structure. We evaluated three methods for selecting the predicted category from the aggregated probabilities: argmax of the Sum, Sharpest Distribution, and Weighted. Each method offers a unique approach to leveraging the ensemble outputs for achieving a robust and reliable final classification.

- **Argmax of the Sum**: This method aggregates the probabilities across all permutations of the prompt categories and selects the category with the highest total probability using the *argmax* function. By considering the cumulative likelihood of each category across different prompt structures, this approach is more robust to fluctuations in individual predictions. It effectively averages out noise and identifies the most likely category based on comprehensive evidence from all prompts.

- The **Sharpest Distribution method**, in contrast, selects the category with the highest individual probability among all prompts. This method captures the model's single most confident prediction, which can be advantageous when the model demonstrates high confidence in a particular category. However, this approach may also be more sensitive to noise or artifacts introduced by the specific structure of a prompt. The sharpest distribution can sometimes reflect outlier behaviors, where a single prompt disproportionately influences the final prediction due to its structure.

- The **Weighted method** computes a weighted average of the probabilities across all permutations of the prompt categories, allowing for differential weighting of each prompt based on their perceived reliability or relevance. This method provides a nuanced prediction by giving more importance to certain prompt outputs deemed more indicative of the correct category. The weighted approach seeks a balance between high-confidence predictions and the inclusion of diverse evidence from multiple prompts. However, the effectiveness of this method depends on the proper selection and tuning of weights. If not appropriately tuned, the weights could either undervalue significant predictions or overemphasize less relevant ones, potentially leading to

suboptimal classification performance.

By comparing these three methods for selecting the predicted category from the aggregated probabilities, we aim to understand how the choice of approach affects final predictions, particularly given the model's sensitivity to prompt structure (see Figure 6). This comparative analysis will help identify which method most effectively produces accurate and robust predictions for legal document classification tasks. Understanding these differences is crucial, as it provides insights into how varying methodologies can either mitigate or exacerbate the model's inherent sensitivity to changes in prompt structure. Before presenting the experimental results, we adapted the Conformal Prediction (CP) formalism to accommodate these permutations, ensuring that the predictive intervals remain valid despite the variability introduced by different prompt structures. By incorporating the permutations into the CP framework, we account for the potential effects of prompt and category ordering on model predictions, thus providing more reliable and well-calibrated prediction intervals.

3.1.3 Split CP Steps with permutations.

One of the methods we explore is selective classification, which involves making predictions only when the model is confident enough and abstaining when it is uncertain. Within the conformal prediction framework, this is achieved by using the size of the prediction set as a threshold: a smaller prediction set indicates higher confidence, allowing the model to make a prediction, whereas a larger set indicates uncertainty, prompting the model to abstain. To further investigate the robustness of predictions, we adapt the Split Conformal Prediction (CP) procedure by introducing permutations of input features and class labels.

STEP 1. Permutation Generation:

- For each input X_{test}, generate a set of permutations $\mathcal{P}(X_{\text{test}}) = \{X_{\text{test}}^{(1)}, X_{\text{test}}^{(2)}, \ldots, X_{\text{test}}^{(m)}\}$ where each permutation represents a different ordering of its features or an augmentation that preserves the label but alters the input in a controlled manner.
- Additionally, generate a set of permutations of the class labels $\mathcal{P}(\mathcal{Y}) = \{\mathcal{Y}^{(1)}, \ldots, \mathcal{Y}^{(k)}\}$, where each $\mathcal{Y}^{(j)}$ is a different ordering of the class labels.

STEP 2. Establish Heuristic Uncertainty Notions:

- Define heuristic notions of uncertainty for the classifier, particularly in the context of input and class label permutations. This involves analyzing how variations in the input data $\mathcal{P}(X)$ and the order of class labels $\mathcal{P}(\mathcal{Y})$ affect the classifier's predictions and the resulting nonconformity scores.

STEP 3. Define the Nonconformity Measures/Score Function:

- For each instance (X, Y), define the nonconformity score function $S(X, Y) \in \mathbb{R}$.
- With permutations, the score is computed over multiple versions of the input X and the class labels $\mathcal{Y}$. Let $\mathcal{P}(X) = \{X^{(1)}, X^{(2)}, \ldots, X^{(m)}\}$ represent a set of m permutations of X, and let $\mathcal{P}(\mathcal{Y}) = \{\mathcal{Y}^{(1)}, \mathcal{Y}^{(2)}, \ldots, \mathcal{Y}^{(k)}\}$ represent a set of k permutations of the class labels.

- The aggregated nonconformity score $N(X, Y)$ is then defined as:

$$N(X, Y) = \text{Aggregate}\left(\{S(X^{(i)}, Y^{(j)})\}_{i=1,j=1}^{m,k}\right),$$

  where the aggregation function could be the mean, maximum, or another suitable measure over the set of nonconformity scores.

STEP 4. Compute $\hat{q}$ as the $\left\lceil \frac{(n+1)(1-\alpha)}{n} \right\rceil$ Quantile of the Nonconformity Scores:

- Using a calibration set, compute the nonconformity scores for each calibration example (X_j, Y_j) and their corresponding permutations $\mathcal{P}(X_j)$ and $\mathcal{P}(\mathcal{Y}_j)$.
- Calculate the aggregated nonconformity score $N(X_j, Y_j)$ for each calibration point.
- Determine the threshold $\hat{q}$ as the $\left\lceil \frac{(n+1)(1-\alpha)}{n} \right\rceil$ quantile of the sorted set of aggregated nonconformity scores $\{N(X_1, Y_1), \ldots, N(X_n, Y_n)\}$.

STEP 5. Generate Prediction Sets for New Examples:

- For a new test input X_{test}, generate a set of permutations

$$\mathcal{P}(X_{\text{test}}) = \{X_{\text{test}}^{(1)}, X_{\text{test}}^{(2)}, \ldots, X_{\text{test}}^{(m)}\}$$

  and a set of permutations of the class labels $\mathcal{P}(\mathcal{Y}) = \{\mathcal{Y}^{(1)}, \mathcal{Y}^{(2)}, \ldots, \mathcal{Y}^{(k)}\}$.
- Compute the nonconformity score $S(X_{\text{test}}^{(i)}, y^{(j)})$ for each permutation $X_{\text{test}}^{(i)}$ and each class label $y^{(j)} \in \mathcal{Y}$.
- Aggregate these scores to obtain $N(X_{\text{test}}, y)$ for each y.
- The prediction set $C(X_{\text{test}})$ is then defined as:

$$C(X_{\text{test}}) = \{y \in \mathcal{Y} : N(X_{\text{test}}, y) \leq \hat{q}\}.$$

- This prediction set includes all labels y for which the aggregated nonconformity score is less than or equal to the threshold $\hat{q}$.

This modified approach allows us to assess the impact of different prompt structures and label orderings on the model's predictions, providing a comprehensive evaluation of the model's stability and the effectiveness of each method in handling prompt sensitivity.

3.1.4 Experimental Results

The results presented in the Figure 6 illustrate the performance of three different methods—Sharpest, Average, and Weighted—applied to three separate prompts for a classification task. Each subplot shows the sorted output probabilities for each method across different prompts, providing insights into how these methods manage prediction confidence. The Sharpest Method (left column) consistently exhibits a narrower range of high-probability values, indicating more decisive predictions when the model is confident. This method's predictions tend to cluster at higher probability values, reflecting a conservative approach that avoids uncertainty. In contrast, the Average Method (middle column) displays a more

gradual increase in probability values, suggesting a balanced approach that averages the predictions across permutations, potentially leading to a more inclusive prediction set. The Weighted Method (right column) shows a similar gradual trend but with variations in the distribution characteristics, indicating that it might prioritize certain factors differently, leading to distinct probability clusters.

Consistency across prompts is evident, as all methods show relatively stable 90th percentile probabilities, accuracy, and F1 scores. However, there are subtle variations in the distribution of predictions, particularly at higher probability levels, indicating that the model's output is sensitive to the structure of the prompts. The plots highlight a thresholding mechanism at the 90th percentile, represented by a horizontal blue line, above which high-confidence predictions are clustered. This clustering varies slightly between methods, reflecting different approaches to managing prediction uncertainty. These findings underscore the need to consider method choice carefully, as each approach impacts the robustness and reliability of model predictions in tasks requiring high accuracy, such as legal document classification. Future work should focus on further refining these methods and exploring additional strategies to enhance model stability and performance under varying prompt conditions.

In an ideal scenario for a classifier, the plot would exhibit clear separations or gaps in the sorted probabilities, reflecting the model's high confidence in its predictions. A perfect plot would show distinct clusters of high-probability values separated by gaps, indicating effective discrimination between classes. This separation would imply the model's outputs are not overly sensitive to prompt structure and maintain robust performance across different prompt variations. Gaps in the probability distribution indicate that the model is not only confident in its predictions but also capable of clearly distinguishing between the likelihoods of different classes, thereby enhancing the model's effectiveness and reliability in practical applications.

Prompt	Method	Top-1 Acc.	Top-2 Acc.	Cov. Set Size 1	Cov. Set Size 2	Avg. Set Size	Acc. 95% CI	F1 Score 95% CI
1	Sharpest	0.55	0.88	0.55	0.88	1.00	(0.50, 0.59)	(0.50, 0.59)
1	Average	0.63	1.00	0.63	1.00	0.96	(0.59, 0.67)	(0.59, 0.67)
1	Weighted	0.63	1.00	0.63	1.00	0.94	(0.59, 0.67)	(0.59, 0.67)
2	Sharpest	0.55	0.94	0.55	0.94	1.00	(0.51, 0.59)	(0.51, 0.59)
2	Average	0.61	1.00	0.61	1.00	0.97	(0.56, 0.65)	(0.56, 0.65)
2	Weighted	0.61	1.00	0.61	1.00	0.96	(0.56, 0.65)	(0.56, 0.65)
3	Sharpest	0.52	0.90	0.52	0.90	1.00	(0.48, 0.56)	(0.48, 0.56)
3	Average	0.56	1.00	0.56	1.00	0.98	(0.52, 0.61)	(0.52, 0.61)
3	Weighted	0.56	1.00	0.56	1.00	0.97	(0.52, 0.61)	(0.52, 0.61)

TABLE 4: Performance metrics for each prompt and method, including Top-1 and Top-2 accuracy, coverage at different set sizes, average set size, and confidence intervals for accuracy and F1 score. Metrics for Conformal Prediction Set Sizes. The table shows the Top-1 accuracy and coverage for each prompt at different prediction set sizes. Coverage at prediction set size 2 represents the percentage of instances where the true label is among the top two predicted labels. All values are rounded to two decimal places

The table summarizes the performance of different methods (Sharpest, Average, and Weighted) across three prompts. For all prompts, the **Average** and **Weighted** methods consistently achieve higher **Top-1 Accuracy** compared to the **Sharpest** method. Both the **Average** and **Weighted** methods show perfect **Top-2 Accuracy** (1.00) across all prompts, indicating that the true label is always within the top two predicted labels when

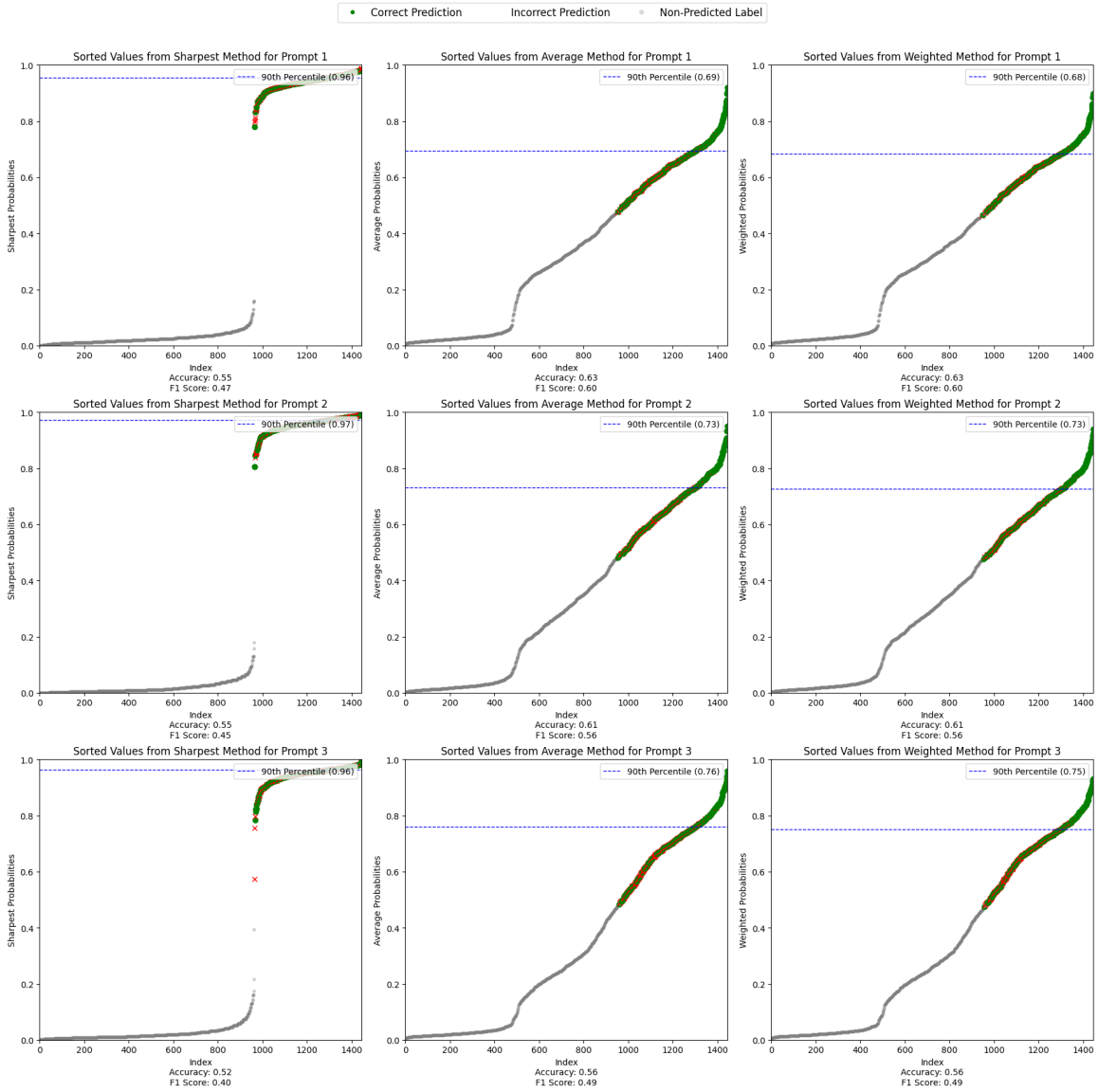


FIGURE 6: **Figure X: Sorted Prediction Probabilities for Conformal Prediction Calibration Across Methods and Prompts.** Each subplot shows sorted prediction probabilities for documents under three methods—Sharpest, Average, and Weighted—across three prompts. The x-axis represents sorted indices, and the y-axis shows probability values (0 to 1). Markers indicate prediction correctness: green circles (correct predictions), red crosses (incorrect predictions), and grey dots (non-predicted labels). The dashed blue line represents the 90th percentile threshold (1 - $\alpha = 0.90$) used for conformal prediction calibration, retaining 90% of confident predictions (with $\alpha = 0.10$ error rate). Annotations within each subplot display Accuracy, F1 Score, empirical Coverage, and Error Rate, providing insights into each method's performance. The figure highlights the distribution of probabilities and the balance between confidence and accuracy across different prediction methods and prompts.

these methods are used.

Coverage at set sizes 1 and 2 mirrors the Top-1 and Top-2 accuracies, reinforcing that **Average** and **Weighted** methods provide more reliable predictions. The **Average Set Size** is close to 1 for the **Sharpest** method, indicating it tends to provide a narrower

prediction set, while the other methods offer slightly more comprehensive sets (below 1 on average, reflecting cases where prediction probabilities are narrowly distributed).

The **confidence intervals** for accuracy and F1 score show overlapping ranges across methods within the same prompt, suggesting there is no statistically significant difference in performance between the **Average** and **Weighted** methods. However, both outperform the **Sharpest** method in terms of both accuracy and prediction coverage.

The prediction set size can be adjusted based on the desired confidence level, allowing researchers to balance between precision and coverage. Table 4 explores the relationship between prediction set size and two key metrics: Top-1 Accuracy and Coverage.e

- Top-1 Accuracy refers to the percentage of instances where the most probable label predicted by the model matches the true label.
- Coverage indicates the percentage of instances where the true label is included in the prediction set provided by the model.

This study examines how varying the prediction set size impacts these metrics across different prompts or datasets. Using three distinct prompts, the prediction probabilities were calculated across all possible categories for each instance. For each prompt, the top-1 accuracy and coverage were analyzed to provide insights into how these metrics change with prediction set sizes.

The findings demonstrate that expanding the prediction set size from 1 to 2 significantly improves coverage while maintaining the same top-1 accuracy. This outcome is expected with conformal prediction methods. When the prediction set size is 1, the model only considers the most probable single label, leading to lower coverage because it often misses the true label. However, by increasing the set size to 2, the model becomes more inclusive, achieving near-total coverage (close to 100%). This means that the true label is almost always among the top two predictions, making the prediction set more reliable. Top-1 Accuracy remains unchanged because it measures the correctness of the most probable prediction and coverage increases as the set size expands, indicating a higher likelihood of the true label being included within the prediction set.

These findings emphasize the trade-off between precision and coverage in conformal prediction settings, highlighting the importance of selecting an appropriate prediction set size based on the specific requirements of the application. Future studies should continue to explore this balance, especially in scenarios where varying levels of accuracy and coverage are crucial for effective decision-making. Furthermore, to enhance the prediction sets and achieve more confident responses, we propose introducing a novel method called Decoding by Contrasting Layers, which is explained in detail in the next section. This method aims to refine prediction confidence by leveraging multiple model layers, providing a more nuanced approach to classification tasks.

4 Decoding by Contrasting Layers

As discussed in the introduction, LLMs tend to "hallucinate", a term used to describe the production of content that strays from the facts they were trained on and not factual data. This issue arises from their training process, which aims to minimize the disparity between the training and generated data. As a result, these models may assign probability to sentences that do not align with the facts in the training data. In practical terms, this leads to models recognizing patterns in the training examples rather than comprehending and generating real-world facts. Meng et al. (2022) show that factual knowledge can even be edited by manipulating a specific set of feedforward layers within an autoregressive LM. We propose to exploit this modular encoding of knowledge to amplify the factual knowledge in an LM through a contrastive decoding approach, where the output next-word probability is obtained from the difference in logits between a higher layer versus a lower layer additionally, Meng et al. (2022) demonstrated that specific feedforward layers within an autoregressive language model can be manipulated to edit factual knowledge.

In light of the promising findings detailed in the research paper by Chuang et al. (2024), we are adopting the Decoding by Contrasting Layers (DoLa) approach to improve the factual accuracy of the language model. The paper suggests that by utilizing the layered encoding of knowledge in transformers, particularly by contrasting outputs from higher and lower layers, we can reduce the tendency of LLMs to hallucinate and produce more factually accurate content. Given that higher layers in a transformer model encode more complex and semantic information, while earlier layers capture more basic details, this method prioritizes knowledge from the top layers, where factual information is more likely to be accurate. The DoLa approach has been proven to enhance the truthfulness of responses, facilitate better factual reasoning, and improve the overall factual accuracy of generated content. Furthermore, it achieves these improvements with only a minimal impact on the decoding speed, making it a practical and effective approach. Considering these results, implementing DoLa will ensure that the model produces more reliable and accurate outputs in our applications.

Contrastive Decoding is a method designed to enhance the quality of text generation by leveraging the differences in token probabilities between a large language model (referred to as the expert model) and a smaller language model (referred to as the amateur model). This approach is based on the insight that smaller models, when used for open-ended text generation, often produce shorter, repetitive, and less relevant outputs, leading to reduced user engagement. Conversely, larger models are better at generating factually accurate and contextually appropriate outputs. The key concept behind Contrastive Decoding involves adjusting the token distribution by calculating the variance in log probabilities between the expert and amateur models as shown in Fig.7. This method eliminates tokens with significantly lower probabilities in the expert model from the candidate pool during generation, resulting in outputs with improved fluency and coherence. Unlike traditional decoding strategies like greedy decoding, top-p sampling, or top-k sampling, Contrastive Decoding does not require additional model training, making it an efficient and practical approach. However, it is important to note that Contrastive Decoding is only applicable when the expert and amateur models share the same vocabulary, as demonstrated in Chuang et al. (2024) in experimental setups using models like OPT 1B and OPT 125M for the amateur, or

utilizing GPT-2 in both roles. Additionally, this method's reliance on two separate models to produce a single output presents a significant computational overhead, which may limit its feasibility in certain scenarios.

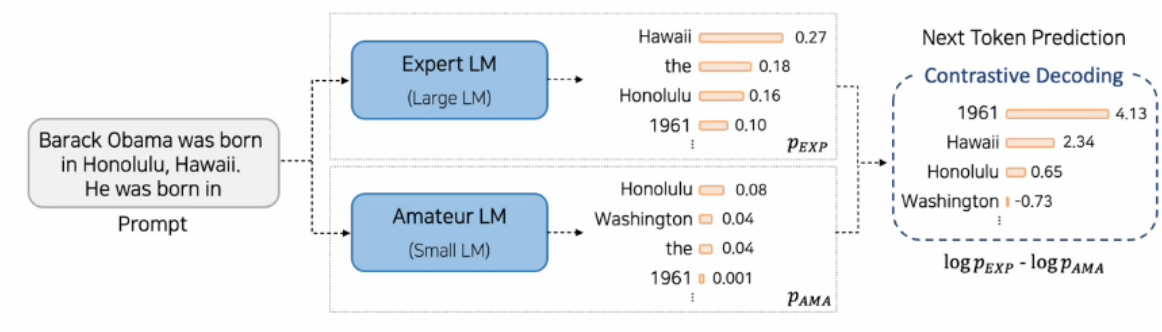


FIGURE 7: Contrastive Decoding Schema.

4.1 DOLA Methodology

As explained in Section 2.1, the basic next-token prediction method passes all transformer layers and utilizes only the final layer's output to generate the vocab distribution. However, DoLa utilizes the "early exit" that works as illustrated in Figure 8, the hidden states from different transformer layers are passed through the vocab head to generate token probabilities. The next-token probability distribution is calculated using the outputs from each intermediate transformer layer $j \in \mathcal{J}$ as we recall from Eq.2.4:

$$q(x_t|x_{<t}) = softmax\phi(h_t^{(j)}))_{x_t}, \quad x_t \in \mathcal{X}, \quad j \in \mathcal{J} \tag{4.1}$$

where $\mathcal{J} \in \{1, \ldots, N-1\}$ is the set of candidate layers for premature layer selection, X is the vocabulary set and N is the final layer, also known as the "mature layer".

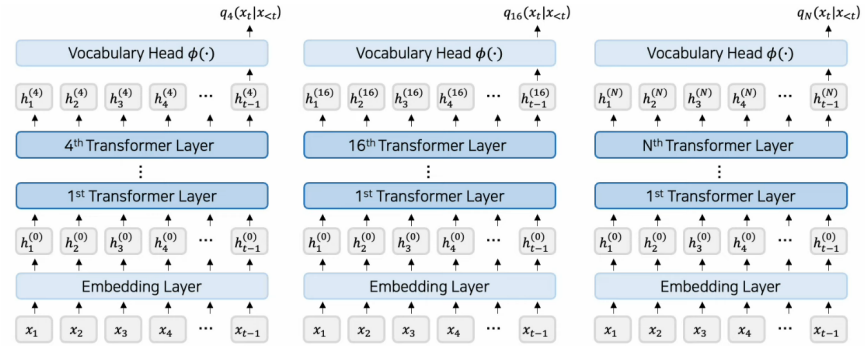


FIGURE 8: DOLA Early Exit: The fourth transformer layer are passed through the vocab head to generate token probabilities. Similarly, the process is repeated for the 16th transformer layer, as shown in the middle figure. Finally, the output of the final (nth) transformer layer is also passed through the vocab head to generate token probabilities.

DoLa compares the information from these intermediate layers with the final N^{th} layer's information to adjust the token probabilities for next-token prediction according to:

$$\hat{p}(x_t \mid x_{<t}) = \text{softmax}\left(\mathcal{F}\left(q_N(x_t|x_{<t}), q_M(x_t|x_{<t})\right)\right)_{x_t|x_{<t}}$$

$$\text{where } M = \arg\max_{j \in J} d\left(q_N(\cdot), q_j(\cdot)\right) \tag{4.2}$$

Here, layer M is the premature layer and the operator $\mathcal{F}(\cdot|\cdot)$ denotes the contrast function between the two output layers' distributions by computing the log-domain difference between two distribution. Once the token probabilities are modified through contrast, they are fed into a softmax function to obtain the vocab distribution.

A crucial aspect of this process lies in determining which premature layers M should be used for comparison. To magnify the effectiveness of contrastive decoding, the optimal premature layer should ideally be the layer most different from the final-layer outputs. The motivation for selecting the layer with the highest distance d($\cdot$, $\cdot$) is to ensure that the model would significantly change its output after that selected layer, and thus have a higher chance to include more factual knowledge that does not exist in the early layers before This way, we dynamically select in each decoding step the one with highest a distributional distance $d(\cdot|\cdot)$ between the mature layer and all the candidate layers in $\mathcal{J}$.

Premature Layer Selection Chuang et al. (2024) compares the final layer's next-token probability is contrasted with the intermediate layer that has the most different distribution, based on Jensen-Shannon Divergence (JSD). The layer with the greatest JSD difference is called the "pivot layer" and this layer is contrasted with the final layer to adjust the token probabilities. The paper states that the primary layer, which is the final layer (nth transformer layer), will be contrasted with the layer that exhibits the greatest JSD from the primary layer. As a result, when generating a specific token t, the token probability from the pivot layer will be contrasted with that from the final layer.

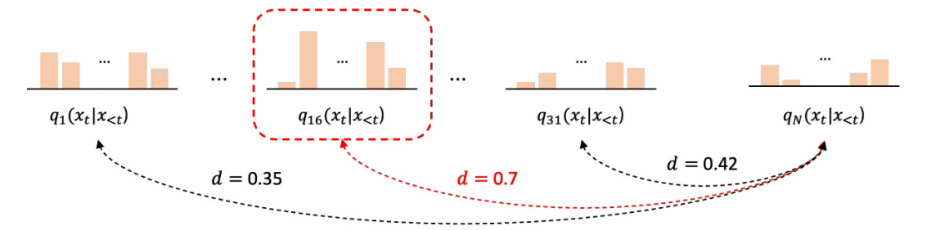


FIGURE 9: Jensen-Shannon Divergence (JSD): The 16^{th} is the "pivot layer" layer as it has the greatest JSD difference. In the example below, the 16th layer is selected for contrast because it shows the greatest JSD when compared to the 31st layer and the first layer. As a result, when generating a specific token t, the token probability from the 16th layer will be contrasted with that from the final layer.

A preliminary analysis was conducted using the `"microsoft/Phi-3-mini-4k-instruct"` model, which comprises 32 transformer layers. The input for this analysis was the prompt provided earlier in Section 3.1, aiming to generate the expected output "2. Work Law." The analysis involved calculating the JSD between the output distributions of the final layer, $q_N(\cdot|x_{<t})$, and each of the intermediate layers, $q_j(\cdot|x_{<t})$ for every next-token prediction.

In Figure 10 we present the heatmap of the Jensen-Shannon Divergence between the final 32nd layer and various early layers of a model across different tokens .The low JSD values suggest that these early layers maintain a representation quite distinct from the final representation. However, as we move to later layers, the values gradually increase, indicating

that these layers start capturing more complex features, eventually aligning closer to the final representation in deeper layers. The heatmap shows that gradient of JSD values are lower (indicating similar representations to the final layer), these layers might be capturing redundant information. In a classification task, where the objective is to map inputs to distinct categories, retaining layers that do not significantly alter the representation might not be necessary. Therefore, the model could potentially be pruned by removing or combining these early layers without a substantial loss in performance. This could lead to a more compact model that is faster and requires less memory.

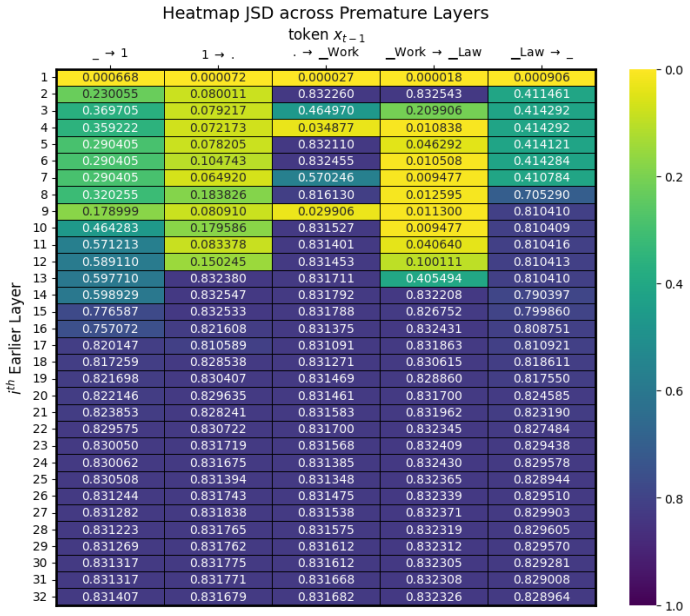


FIGURE 10: JSD (scaled by 10^5) between the final 32nd layer and the rest of the early layers. Column names are decoded tokens in each step. Row names are indices of the early layers. 0 means word embedding layer.

Different token generation show different patterns of divergence. The generqtion of 2 and *Law* have relatively consistent and lower JSD values across all layers, implying their representations evolve more gradually across layers. On the other hand, the generqtion of tokens such as *Work* show higher JSD values in certain layers, particularly around the middle layers, indicating that the model's representations for these tokens undergo more significant changes at those stages. showIing higher JSD values in the later layers indicates that the language model changes its prediction direction and **incorporates factual knowledge in the later layers**. Lower values of JSD in the intermediate layers, suggest that the model had already decided what to generate in the earlier layers, with little change in the output distribution in the later layers.

The scaled probability heatmap (Figure 11) provides additional insights, the scaled probability heatmap for the generation of token 2 shows that the probability values are relatively low across the early layers (from -32 up to approximately -15). This observation is consistent with the JSD heatmap, where early layers exhibited low divergence from the final representation. The low probabilities indicate that these layers do not contribute significantly to the generation of the token "2". A sharp increase in scaled probabilities begins

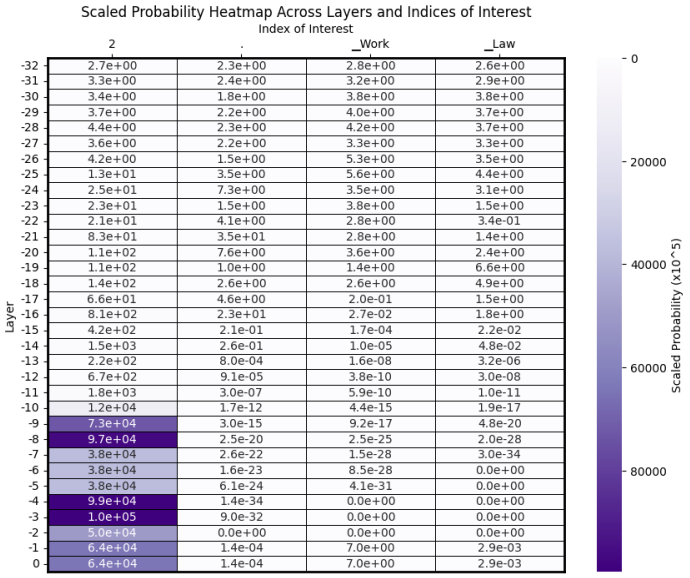


FIGURE 11: Probability heatmap for the indices of interest (scaled by 10^5) between the final 32nd layer and final one for the generation of the token 2. Row names are indices of the early layers. Columns mean word embedding layer.

around layer -14 and continues through to about layer -8. This suggests that the model's representations become more refined and discriminative during these middle layers, where probabilities start to increase significantly. These layers likely perform key transformations necessary for distinguishing between different classes or generating specific tokens, underscoring their importance. Therefore, pruning should be approached carefully around these layers, as they are critical for the model's ability to capture and refine complex features.

These heatmaps led to two conclusions: First, factual next-token predictions were more influenced by the later layers in the model, and thus, contrasting with the layer showing the highest JSD could lead to more factual outputs. Second, the pivotal layer, i.e., the layer with the highest JSD, differs for each decoding step, necessitating a dynamic pivotal layer selection process. The dynamic pivotal layer selection process is essential because the optimal layer to contrast against the final layer changes with each decoding step. The process involves measuring the distance between the token probabilities of the final layer and those of each intermediate transformer layer.

Method: **Dynamic Premature Layer Selection**Process for Selecting the Premature Layer, see Figure 12 for a graphic representation of the following steps:

- The optimal Premature Layer is the one most different from the Final Layer Output. This is done to maximize the contrast.
- In each decoding step:
  - Measure the distance between the Next Token Probabilities of two layers using Jensen-Shannon Divergence:

$$d\left(q_N(\cdot \mid x_{<t}), q_j(\cdot \mid x_{<t})\right) = \mathrm{JSD}\left(q_N(\cdot \mid x_{<t}) \parallel q_j(\cdot \mid x_{<t})\right)$$

– Select the Layer with the largest distance as the Premature Layer $\mathcal{M}$:

$$\mathcal{M} = \arg\max_{j \in \mathcal{J}} \text{JSD}\left(q_N(\cdot \mid x_{<t}) \parallel q_j(\cdot \mid x_{<t})\right)$$

– *where $\mathcal{J}$ is the set of candidate early layers for premature layer selection.*

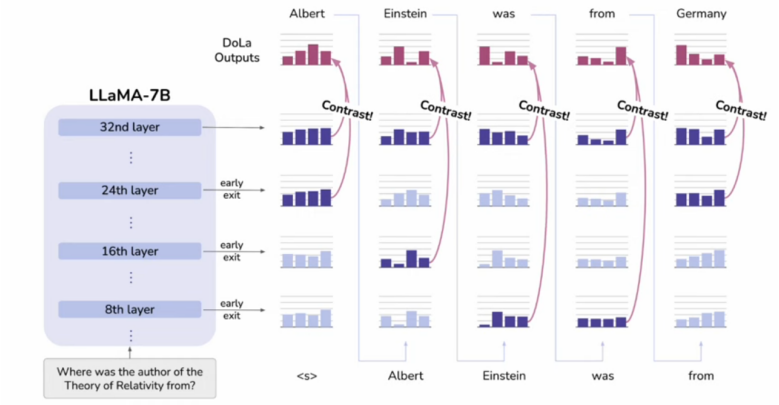


FIGURE 12: Dynamic Premature Layer Selection Scheme. Figure extracted from Chuang et al. (2024).

The layer with the highest JSD is identified as the pivotal layer. However, in our scenario, due to the numerical nature of our classification labels, we can restrict ourselves to single-token predictions for these numbers. As a result, we can terminate the process at the first step of the Dynamic Premature Layer Selection. Nonetheless, we will choose a different pivotal layer for each unique prompt, ensuring that the model dynamically adapts to the specific requirements of each input. See in Figure 13 the JSD divergence for the generation of a category number for a specific prompt, in this case, a layer between -15 to -10 would be a great option.

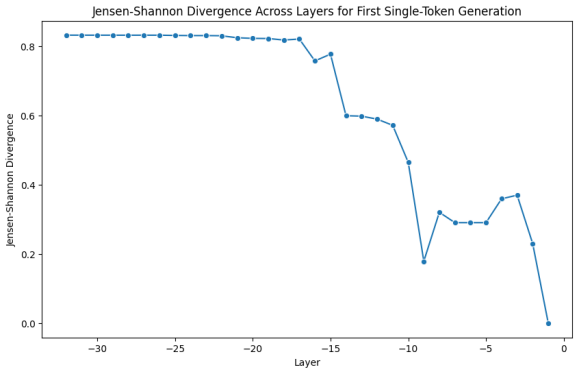


FIGURE 13: **Jensen-Shannon Divergence Across Layers for First Single-Token Generation for one prompt.** JSD values between each layer's output representation and the final token distribution. The early layers (from -32 to -15) maintain high JSD values, indicating stable yet distinct representations from the final output. A sharp decrease in JSD occurs in the middle layers (around -15 to -10), suggesting significant transformations crucial for refining the model's representations. The JSD stabilizes in the late layers (from -10 to -2), reflecting minor refinements leading to the final output, with a notable drop at layer 0, indicating alignment with the target distribution.

For efficient implementation, we select the premature layers from an optimal subset $\mathcal{J}$ of the model's total number of layers. The best bucket is chosen based on validation set performance. We could do a DoLa-static method, using a fixed pivotal layer throughout all the prompts, based on the validation performance. However, this method has some clear drawbacks, such as requiring extensive validation testing to determine the optimal pivotal layer and being highly sensitive to data distribution. DOLA-dynamic reduces the number of validation tests needed since it dynamically selects the pivotal layer for single-token prediction at each prompt. This dynamic selection also makes DOLA less sensitive to changes in data distribution, thus showing more robust performance across various tasks.

Next, let's consider the **predictions' contrast process** in DOLA. The main idea is to increase the influence of the final layer while reducing that of the pivotal layer, thus relying more on the factual internal knowledge of the language model to generate factual outputs. The formula used in DOLA is very similar to that of contrastive decoding: subtracting the log probabilities of the pivotal layer from those of the final layer. This subtraction is used to calculate the next token's probability. However, there are some potential issues with this approach:

- False Positive: A scenario where an unsuitable token ends up with a relatively high probability due to the contrastive adjustment (Figure 14).
- False Negative: A situation where the model is already confident about a certain token, but the minimal probability difference between the pivotal and final layers results in an undesirably low probability after contrastive decoding (Figure 15).

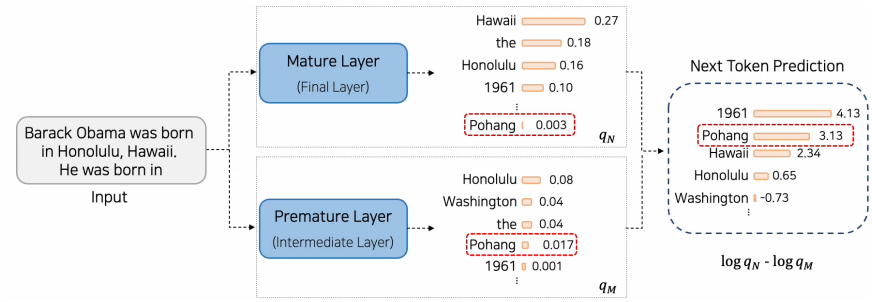


FIGURE 14: Illustration of a False Positive Case: The log probability of the final layer (Layer N) minus the log probability of the pivotal layer (Layer M) results in a higher next token probability for an inappropriate token.

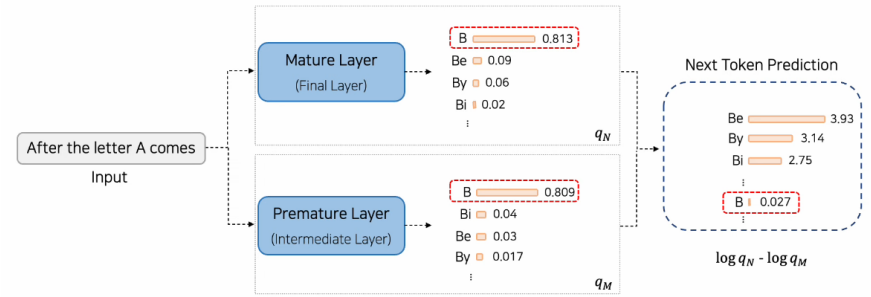


FIGURE 15: Illustration of a False Negative Case: Both models' confidence in a certain token is high, but the small difference between the final layer's log probability and the pivotal layer's log probability leads to a low next token probability after contrastive decoding.

To mitigate these issues, the method includes a technique where tokens that fall below a certain threshold are excluded from the candidate set, so that the distribution for the next-token is:

$$\hat{p}(x_t \mid x_{<t}) = \text{softmax}\left(\mathcal{F}(q_N(x_t), q_M(x_t))\right)_{x_t} \tag{4.3}$$

$$\text{where } \mathcal{F}(q_N(x_t), q_M(x_t)) = \begin{cases} \log \frac{q_N(x_t)}{q_M(x_t)}, & \text{if } x_t \in \mathcal{V}_{\text{head}}(x_t \mid x_{<t}), \\ -\infty, & \text{otherwise.} \end{cases} \tag{4.4}$$

Similar to Li et al. (2022), the subset $\mathcal{V}_{\text{head}}(x_t \mid x_{<t}) \in \mathcal{X}$ is defined as whether the token has high enough output probabilities from the mature layer,

$$\mathcal{V}_{\text{head}}(x_t \mid x_{<t}) = \{x_t \in \mathcal{X} : q_N(x_t) \geq \alpha \max_w q_N(w)\}. \tag{4.5}$$

See in Figure 16 a schematic representation of the whole process.

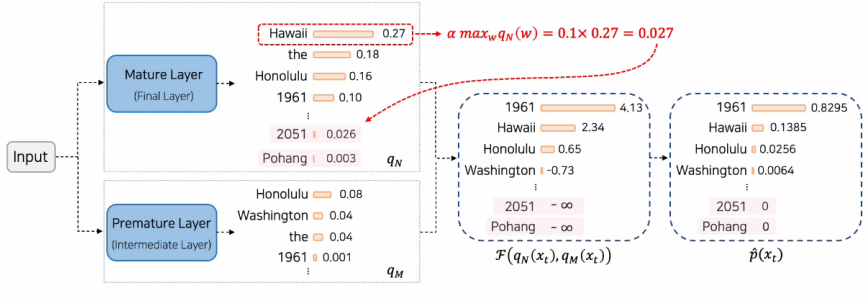


FIGURE 16: **Dynamic Output Layer Adjustment (DOLA) in Large Language Models (LLMs).** The diagram illustrates the DOLA mechanism in an LLM, where outputs from both a mature (final) layer and a premature (intermediate) layer are combined. The mature layer generates the final token probabilities (q_N), while the premature layer provides earlier layer outputs (q_M). DOLA dynamically adjusts the predictions by integrating information from both layers using a weighted combination, computed as $\alpha \cdot \max_w q_M(w) \cdot q_N(w)$, to refine the final output distribution ($\hat{p}(x_t)$). This method enhances the model's adaptability and accuracy by leveraging insights from intermediate layers along with final outputs.

4.1.1 Experimental Results

Chuang et al. (2024) analyzed the effectiveness of dynamic pivotal layer selection, demonstrating that while DOLA-static occasionally peaked higher in specific instances, it was less consistent than DORA-dynamic, which provided robust performance across different datasets. This consistency is why we chose to use the dynamic selection method for our experiments.

We employed the DOLA method for a single prompt to generate a new probability distribution, as illustrated in Figure 17. We expected that the output would correctly identify the actual label, "Labour Work," with increased confidence compared to the original distribution, given that this method aims to mitigate factual hallucinations. However, despite our efforts to reduce false positives and negatives, the DOLA method resulted in an incorrect

prediction. It produced a sharper probability distribution favouring categories 1 and 2 and the None category far away. However, it did not improve the results or clearly differentiate those two.

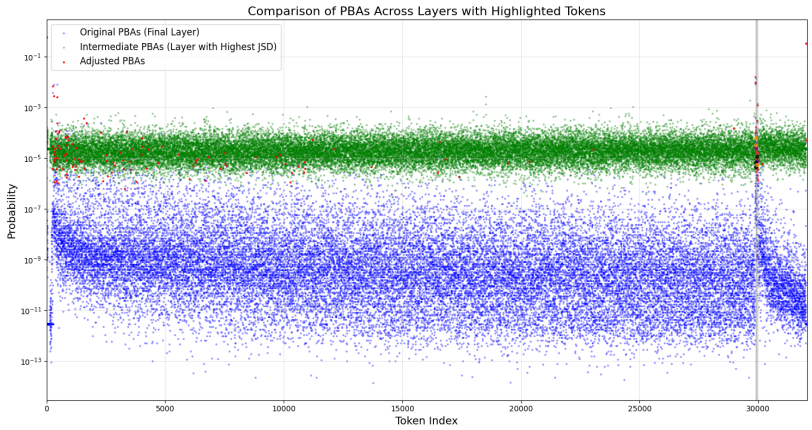


FIGURE 17: Comparison of Probability Distributions (PBAs) Across Different Layers with Highlighted Tokens: The graph displays the original PBAs from the final layer (blue), the intermediate PBAs from the layer with the highest Jensen-Shannon Divergence (green), and the adjusted PBAs (red) for a single token generation, illustrating the distribution changes across layers. The grey regions represent tokens associated with category selection numbers (1, 2, and 3). The black dots denote the probabilities for the true labels, while the orange dots indicate the probabilities according to the new distribution generated using DOLA.

We implemented this method to the same prompts and permutations as in Figure 6, hoping that the performance would improve; however, the accuracy did not improve, as seen in Figure 18. Due to the non-encouraging methods, the wide memory usage, and the computationally time-consuming nature of the method, we do not consider it appropriate for our purposes, mainly due to our approach with several permutations.

Further research could be encouraging as the aim of introducing such a method was to actually mitigate factual hallucinations without requiring additional information retrieval or model fine-tuning, making it an effective and practical approach for improving the factuality of generated text across various tasks.

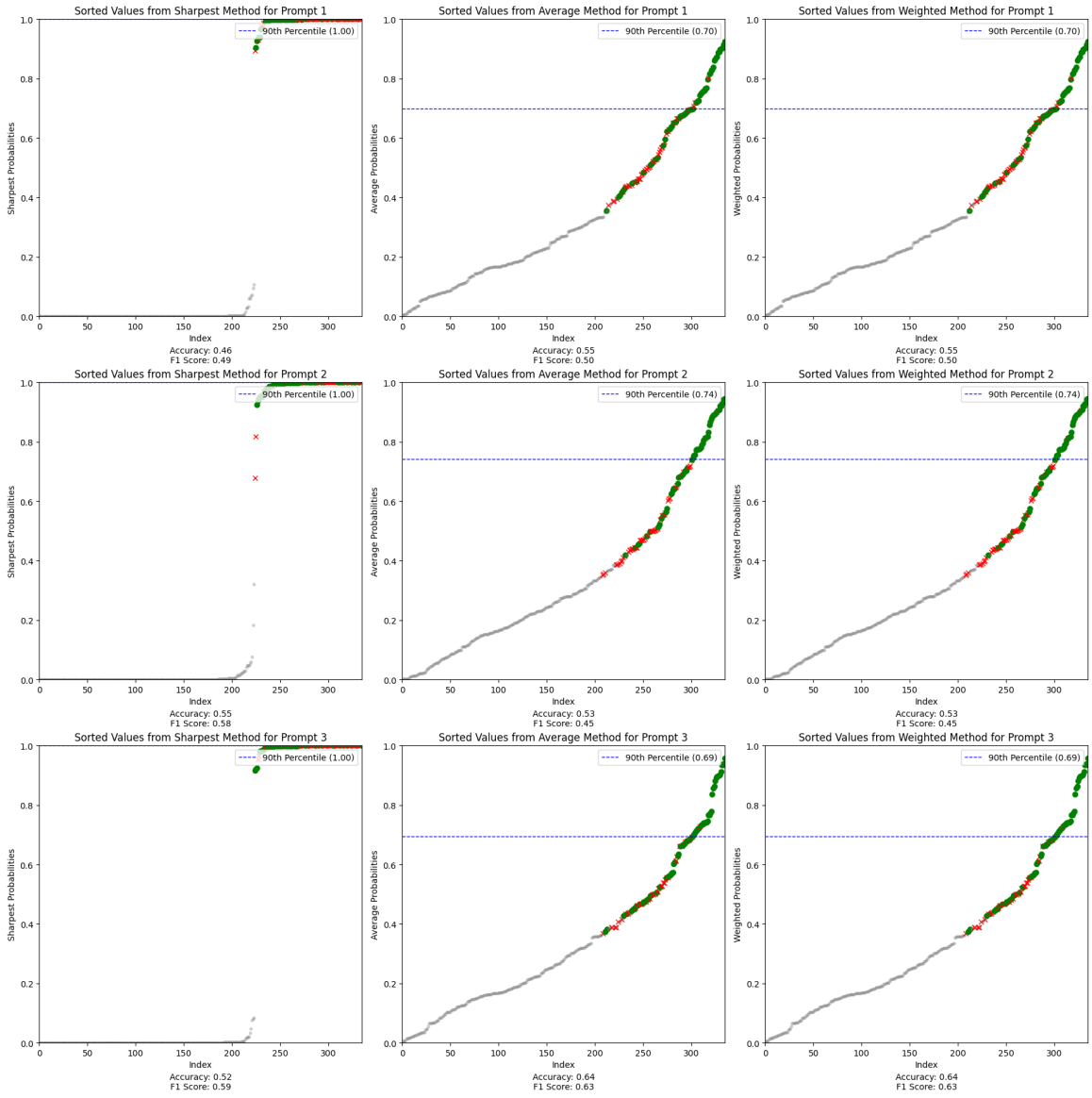


FIGURE 18: **Adjusted Prediction Probabilities for Conformal Prediction Calibration Using the DOLA Method Across Multiple Methods and Prompts.** This figure presents the sorted prediction probabilities for legal document classification using the DOLA (Distribution of Logits Adjustment) method. Three classification approaches—Sharpest, Average, and Weighted—are compared across different prompts. The x-axis represents sorted prediction indices, and the y-axis displays the adjusted probability values (0 to 1). Green circles denote correct predictions, red crosses indicate incorrect ones, and grey dots represent non-predicted labels. The dashed blue line marks the 90th percentile threshold (1 - $\alpha = 0.90$), ensuring that 90% of predictions above this confidence level are retained. Annotations within each subplot summarize key metrics: Accuracy, F1 Score, empirical Coverage, and Error Rate, highlighting the DOLA method's ability to refine predictions by enhancing lower-confidence outputs while preserving high-confidence decisions, thus optimizing the trade-off between confidence and accuracy across varying classification methods and prompts.

5 Conformal Prediction Without Token Access

Regarding the results provided in the previous sections, it is undeniable that conformal prediction helps us offer a proper idea of our model performance by providing mathematically guaranteed, well-calibrated predictions. Also, we remind that CP only has a held-out calibration dataset and an inference dataset to provide them. This is quite interesting, as we previously insisted on models that work with direct model access. Nonetheless, sometimes logits are unavailable, like in APIs like Bard, or where retraining the model is costly or impractical, like with LLMs with access unavailable through third-party or commercial APIs. Also, when available, may be miscalibrated, leading to inaccurate probability estimates and inefficient prediction sets.

To address this issue, inspired by Logit-free Conformal Prediction (LofreeCP) introduced by Kumar et al. (2024), we propose a generalized approach to conformal prediction that does not rely on access to logits. Since Conformal Prediction relies on the ranking of nonconformity scores rather than their absolute values, we could estimate the probability assigned to a category through the frequency of each response through sampling, as we did before. However, instead of considering the model's probabilities and confidence, we rely on the freQency of the answers for each input. However, this method still remains computationally intensive but reduces computational costs as shown in Lemma 5.1. By sampling from the model this way, we can approximate the underlying model-based probabilities using the frequency of these responses as a measure of coarse-grained uncertainty.

The key idea here is that consistency in the rankings of uncertainty across repeated samples can be a strong indicator of the model's confidence in its predictions. If a model consistently ranks certain predictions as more likely across multiple samples, it suggests a higher degree of certainty in those predictions. This method allows us to construct prediction sets that are still well-calibrated and provide robust performance guarantees, even in the absence of direct token-level access to the model.

5.1 Frequency as a Proxy for Rankings

In practice, approximating true predictive probabilities with sampling, rather than estimating probabilities directly, involves conducting a sufficiently large number of samplings. However, achieving a high confidence level with a low margin of error is computationally expensive, making practical implementation challenging.

Lemma 5.1 (Minimum Sample Size for Reliable Probability Estimation)**.** *Let freq(Y_i) represent the frequency of occurrence of the event Y_i in a sample of size N_{total}. Let p_i denote the desired estimate probability of the event Y_i, ϵ be the estimation error, and $0 < \delta < 1$ be the desired confidence level. To ensure that the estimation error does not exceed $\epsilon > 0$ with a confidence level of $1 - \delta$, the following condition must hold:*

$$P\left(\left|\frac{freq(Y_i)}{N_{total}} - p_i\right| \leq \epsilon\right) \geq \delta.$$

The minimum required sample size N_{total} that satisfies the above condition is then given by:

$$N_{total} \geq \left(\frac{z_{(1-\delta)/2}}{2\epsilon} \right)^2,$$

where $z_{(1-\delta)/2}$ is the quantile of the standard normal distribution corresponding to a confidence level of $1 - \frac{(1-\delta)}{2}$.

To empirically validate the affirmation that, we could evaluate how the model ranks its uncertainty, rather than focusing solely on the precise probability values. With this aim, we take the results from our previous results from Figure 6 from or different 3 prompting and permutations 3! for each of the 482 document, and we count for the frequency for each chosen label. The results are shown in Figure 19, focusing on how frequently the model response was "Labor Law" appeared across the samples and to compare them with the corresponding output probabilities provided by the LLM.

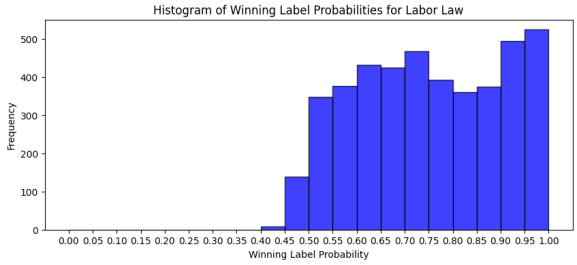


FIGURE 19: LLM response frequency vs. LLM output probability for Labor Law text classification. By analyzing the frequency of each response and correlating it with the model's output probability, we observe a positive correlation: higher response frequencies tend to correspond to higher average predicted probabilities.

The results offered in Figure 19 are consistent with the hypothesis that more frequent responses indicate higher levels of certainty. This motivates the use of response frequency as a proxy for probability ranking. We suggest utilizing response frequency as a stand-in for rating the model's probability in light of this result. We may determine the confidence level of a model without requiring exact probability calculations by concentrating on the model's frequency of producing certain answers.

5.2 Experimental Results

The frequency-based ranking based on m sampled replies for the i-th question may be stated mathematically as follows:

$$F(\hat{y}_a^{(i)}, m) = \frac{\tilde{p}[\hat{y}_a^{(i)}]}{m} \tag{5.1}$$

where $\hat{y}_a^{(i)}$ represents the a-th non-repeated sampled response for i-th prompt, and m is the total number of samples for each prompt. Here, $\tilde{p}[\hat{y}_a^{(i)}]$ is the estimated probability for response a for prompt i. In other words, the Frequency-Based Approach examines the predicted categories across all samples within a document and selects the most frequently occurring category. Based on this approach, we provide insights into the different probabilistic

and frequency-based methods in Table 5.

In previous sections, for probability-based methods (see Figure 6), combining results from different prompts was not an option because it led to worse performance. This decline in performance occurs because the probabilities generated by different prompts can vary significantly due to varying sensitivities or biases associated with each prompt. Mixing these probabilities can create inconsistent or misleading probability distributions, diluting the strength of confident predictions and skewing the final output. However, combining results from different prompts is essential for the Frequency-Based Approach. This method relies on aggregating multiple predictions to determine the most frequently predicted category for each document. Introducing a variety of prompts adds diversity and variability to the predictions, which helps mitigate the biases or errors that may result from depending on a single prompt. This diversity in predictions allows the Frequency-Based Approach to stabilize around the most frequently predicted category, which is more likely to reflect the true category.Despite these reasons for not combining prompts in the probability-based method, we have included such combined results in Table 5 to provide some numerical insights.

Method	**F1 Score (Commercial Law)**	**Accuracy (Commercial Law)**	**F1 Score (Labor Law)**	**Accuracy (Labor Law)**	**Overall Accuracy**
Probability-Based	0.53	42.02%	0.69	83.50%	62.77%
Frequency-Based	0.54	40.66%	0.73	91.27%	65.98%
Average Probabilistic	0.37	24.07%	0.70	95.00%	59.54%
Weighted Average	0.30	18.26%	0.69	95.83%	57.05%

TABLE 5: Performance Comparison of Different Methods in Predicting Legal Categories. The table reports the F1 scores and accuracies for three legal categories—Commercial Law, Labor Law, and None—across four different classification methods. The "Probability-Based" and "Frequency-Based" approaches are compared alongside "Average Probabilistic" and "Weighted Average" strategies, with overall accuracy highlighted for each method. No document was predicted as None within the selected documents, as expected.

The Frequency-Based method lets us mix predictions from all the prompts, giving us the freedom to make our dataset as varied as we want. This approach allows us to take advantage of different perspectives from multiple prompts, which helps balance out any individual biases and improves the overall accuracy. Surprisingly, even though we're not using a specialized or highly advanced model, the Frequency-Based approach has shown a noticeable improvement compared to the other probabilistic methods. This shows that simply combining diverse inputs can make our predictions more reliable and effective.

Despite this achivements, there are inherent drawbacks to using response frequency alone to gauge nonconformity. First of all, the computation cost is quite elevated, however, in order to resolve this, we introuduce VLLM to optilmice this approqch, refer to Section **??**.

6 Computational optimization by Prefix Caching in vLLM

Efficient Memory Management with PagedAttention, vLLM, is a powerful library designed by Kwon et al. (2023) to serve large language models (LLMs) efficiently. It aims to maximise the throughput and minimise the latency of LLM inference, particularly in scenarios where

multiple prompts need to be processed in parallel. The critical innovation of vLLM lies in its use of a sophisticated memory management system that optimises GPU memory utilisation. This system allows vLLM to handle larger batches of prompts and support more efficient model parallelism, thereby enhancing the scalability and performance of LLM deployments.

One of the notable features of vLLM is its support for automatic prefix caching, enabled through the –enable-prefix-caching flag. Prefix caching is a technique that improves the efficiency of LLM inference by storing the results of computations for shared prefixes across different prompts. Multiple prompts may share a common initial sequence or prefix in many natural language processing tasks before diverging into different content. For instance, when prompts are constructed from a standard template with different endings, the initial part of each prompt remains the same. By caching the results of the initial computations, vLLM avoids redundant processing for each prompt that shares the same prefix.

When prefix caching is enabled, vLLM automatically detects these common prefixes among the input prompts. For the first occurrence of a shared prefix, the model computes the necessary outputs and stores them in the cache. For subsequent prompts with the same prefix, vLLM retrieves the cached results rather than recomputing them, significantly reducing the computational load and improving response times. This is particularly advantageous when processing a large volume of data with slight variations in the prompt structure, such as different instructions or categories appended to a shared opening phrase.

By leveraging automatic prefix caching, vLLM enhances the efficiency of processing prompts that differ mainly in the latter part of their text while sharing the same initial context. This makes it ideal for use cases like text summarisation, translation, and classification, where the initial instructions or questions are often the same across multiple requests. As a result, vLLM can serve more requests in less time, providing faster inference speeds and better utilisation of computational resources, especially in batch-processing scenarios. Overall, the prefix caching capability in vLLM represents a significant optimisation for deploying large language models in production environments, allowing for scalable and high-performance language model serving.

Metric	**non-VLLM**	**VLLM**
Documents/Permutations Processed	482 & 8676	482 & 8676
Total Time Elapsed	19:15	08:03
Estimated Speed - Input	7.63 toks/s	20592.31 toks/s
Estimated Speed - Output	7.52 toks/s	16.95 toks/s

TABLE 6: Comparison of two processes for "microsoft/Phi-3-mini-4k-instruct": vLLM demonstrates a significantly higher input speed, indicating faster processing capabilities.

7 Experimental Results

Given the results from Table 6 comparing both performances from the permutations using or not VLLM, we conlcude that thanks to leveraging techniques such as automatic prefix caching in vLLM, we can optimise processing multiple prompts that share common

structures, thereby reducing redundant computations. This optimisation speeds up the overall process and ensures that we maintain statistical integrity and accuracy in the outputs. Therefore, our permutation methods now appear more viable regarding computational efficiency. This opens the door of rather using more powerful models like "meta-llama/Meta-Llama-3.1-8B-Instruct" that could understand better our requirements and thus allow better accuracy performance, which it is the case, as shown in Table 7:

Category	**Accuracy**	**F1 Score**	**Correct Documents /Total**
Overall Accuracy	0.8195	0.8820	395/482
Labor Law	0.9461	0.8048	228/241
Commercial Law	0.6929	0.8434	167/241

TABLE 7: Accuracy Results for each Different Categories using "meta-llama/Meta-Llama-3.1-8B-Instruct". With times of**12:10s, speed input: 13113.87 toks/s, output: 11.87 toks/s**]

Based on our findings, we can conclude that the enhanced model performance validates our hypothesis that more advanced models significantly improve accuracy. Notably, the Commercial Law category continues to exhibit the lowest performance, likely due to its inherent complexity, which suggests that additional targeted analysis is required to better handle this challenging category. The results presented in Table7 demonstrate that leveraging advanced models, particularly through optimization techniques such as those enabled by vLLM, can yield superior performance metrics. This is especially crucial for legal applications, where accuracy and reliability are paramount.

Building on the promising results, we propose expanding the classification framework to achieve more detailed and accurate labelling of documents. This enhanced methodology involves processing documents in batches and evaluating each document against various potential categories. We recommend including a 'None' option to account for documents that do not clearly fit into a specific category. Incorporating multiple prompts will further strengthen the classification process, allowing us to calculate each category's frequency more effectively. This method provides a flexible and adaptive approach, accommodating the complexities and overlaps often in legal texts. We suggest selecting the category with the highest frequency across all prompts to determine the final label. Two or more categories will likely be chosen frequently when a document falls into multiple categories. To address this, we need to set a clear threshold to define what counts as a significant frequency. Establishing this threshold is not straightforward; it requires thoughtful discussion with our clients to ensure it meets their needs and objectives. By customizing the threshold for each project, we can improve the model's accuracy and relevance, resulting in a more effective and context-aware classification system.

8 Conclusions and Further Work

In this thesis, we explored methods for statistically controlling the outputs of Large Language Models (LLMs) to address the instability and sensitivity of their responses to different prompts. Due to the unpredictable nature of prompt-based LLM outputs, we sought to develop techniques for better understanding and managing this variability.

To achieve this, we experimented with various permutation strategies both over the categories and the prompts themselves. We demonstrated that the permutation of prompts can sometimes lead to worsened results, indicating the nuanced nature of prompt engineering and its significant impact on model outputs. Recognizing the challenges associated with enhancing factual knowledge in LLM outputs, we applied Decoding by Contrasting Layers (DoLa) methods. However, we observed that the final accuracy decreased, suggesting limitations in the current approach to improving factual consistency in LLM-generated text.

To further understand the behavior of logits, we proposed an innovative approach analogous to "tournage noir," where we analyze the frequency of different classes instead of focusing solely on logits. This method could offer a new perspective in understanding and controlling LLM outputs, potentially leading to more reliable predictions.

One potential improvement for future research, which was not implemented in this study, involves altering the way final probabilities are computed. The tensors of size (18, 3) obtained from the outputs can be conceptualized as monochrome images or heatmaps. Traditionally, taking the mean in the first dimension or using the sharpest method can be viewed as a version of average pooling (avgpool) and maximum pooling (maxpool) respectively. This observation suggests the possibility of applying a convolutional layer to the (18, 3) tensor. The convolutional layer could be trained to optimize label fitting, potentially improving the robustness and accuracy of the predictions.

In conclusion, our findings underscore the complexity of prompt engineering and its substantial influence on LLM outputs. While current methods provide some control, significant work remains to enhance the reliability and factual accuracy of these models. Future studies should focus on exploring convolutional approaches and other advanced methods for better managing the inherent uncertainties in LLM outputs. This research is a step toward more reliable AI applications in critical fields, emphasizing the need for continuous innovation in controlling and understanding machine learning models.

Our methodology involves generating detailed statistics that reflect the frequency of category assignments across multiple prompts. These statistics provide insights into which categories are most frequently selected and under what conditions. A well-defined threshold helps differentiate between primary and secondary categories, ensuring that multi-label documents are accurately classified based on the prominence of each category's occurrence. To fully leverage the value of our results, we plan to integrate this advanced model into our production environment, enhancing our ability to process and categorize legal documents more accurately and efficiently.

Moreover, we recognize the importance of examining the documents that were classified with the lowest accuracy. This analysis, conducted in collaboration with our clients, will provide deeper insights into improving the detection and contextual understanding of complex legal texts. By understanding the nuances of these challenging cases, we can refine our model further and develop in-context learning strategies tailored to our clients' specific needs. This iterative process of analysis, feedback, and improvement will ensure that our classification system remains robust, adaptable, and highly relevant to the evolving demands of legal practice.

9 Code

In order to access the code used throughout this project, you can visit the following GitHub link: `https://github.com/amtellezfernandez/ControlGenAI`. This code repository contains all the relevant scripts and functions that were utilized in the analysis. However, it's essential to note that this code has been provided without specific data or references to the dataset.

10 Acknowledgments

I want to express my gratitude to Edouard Balzin for his invaluable support and guidance throughout this research. His insightful feedback and unwavering encouragement have been instrumental in completing this project. I also wish to thank Magic Lemp for providing the resources and environment necessary for this work.

References

Angelopoulos, A. N. and Bates, S. (2022). A gentle introduction to conformal prediction and distribution-free uncertainty quantification.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv*, arXiv:2005.14165.

Chalkidis, I., Fergadiotis, E., Malakasiotis, P., Aletras, N., and Androutsopoulos, I. (2019). Extreme multi-label legal text classification: A case study in EU legislation. In *Proceedings of the Natural Legal Language Processing Workshop 2019*, pages 78–87, Minneapolis, Minnesota. Association for Computational Linguistics.

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., and Androutsopoulos, I. (2020). LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Chuang, Y.-S., Xie, Y., Luo, H., Kim, Y., Glass, J., and He, P. (2024). DoLa: Decoding by Contrasting Layers Improves Factuality in Large Language Models. arXiv:2309.03883.

Dai, D., Dong, L., Hao, Y., Sui, Z., Chang, B., and Wei, F. (2022). Knowledge neurons in pretrained transformers. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, Volume 1: Long Papers:pp. 8493–8502.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Douka, S., Abdine, H., Vazirgiannis, M., El Hamdani, R., and Restrepo Amariles, D. (2021). JuriBERT: A masked-language model adaptation for French legal text. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 95–101, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Gammerman, A., Vovk, V., , Vapnik, and Learning, V. (2013). Learning by transduction. *CoRR*, abs/1301.7375.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Chen, D., Dai, W., Chan, H. S., Madotto, A., and Fung, P. (2024). Survey of Hallucination in Natural Language Generation. *arXiv preprint*, arXiv:2202.03629.

Jurafsky, D. and Martin, J. H. (2024). *Speech and Language Processing*. Standford. 3rd edition Draft of August 20, 2024.

Kumar, B., Lu, C., Gupta, G., Palepu, A., Bellamy, D., Raskar, R., and Beam, A. (2023). Conformal prediction with large language models for multi-choice question answering. *arXiv*, arXiv:2305.18404.

Kumar, B., Lu, C., Gupta, G., Palepu, A., Bellamy, D., Raskar, R., and Beam, A. (2024). Api is enough: Conformal prediction for large language models without logit-access. *arXiv*, arXiv:2403.01216v2.

Kumar, B., Palepu, A., Tuwani, R., , and Beam, A. (2022). Towards reliable zero shot classification in self-supervised models with conformal prediction. *arXiv preprint*, arXiv:2210.15805, 2022.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. (2023). Efficient memory management for large language model serving with pagedattention.

Labonne, M. (2023). Decoding Strategies in Large Language Models. Maxime Labonne GitHub blog post.

Louis, A. and Spanakis, G. (2022). A statutory article retrieval dataset in French. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6789–6803, Dublin, Ireland. Association for Computational Linguistics.

Manokhin, V. (2023). *Practical Guide to Applied Conformal Prediction in Python*. Packt, Birmingham–Mumbai.

OpenAI (2022). Introducing chatgpt. `https://openai.com/blog/chatgpt`.

OpenAI (2023). Gpt-4 technical report. `https://cdn.openai.com/papers/gpt-4.pdf`.

Segonne, V., Mannion, A., Alonzo Canul, L. C., Audibert, A. D., Liu, X., Macaire, C., Pupier, A., Zhou, Y., Aguiar, M., Herron, F. E., Norré, M., Amini, M. R., Bouillon, P., Eshkol-Taravella, I., Esperança-Rodier, E., François, T., Goeuriot, L., Goulian, J., Lafourcade, M., Lecouteux, B., Portet, F., Ringeval, F., Vandeghinste, V., Coavoux, M., Dinarelli, M., and Schwab, D. (2024). Jargon: A suite of language models and evaluation tasks for French specialized domains. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9463–9476, Torino, Italia. ELRA and ICCL.

Tenney, I., Das, D., and Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline. *arXiv preprint*, arXiv:1905.05950.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). *Attention is all you need*. NeurIPS, Long Beach, CA, USA. 31st Conference on Neural Information Processing Systems (NIPS 2017).

Vovk, V., Gammerman, A., and Shafer, G. (2022). *Algorithmic Learning in a Random World*. Springer International Publishing. doi: 10.1007/978-3-031-06649-8.

A French Legal Dataset

Affaires	Droit des sociétés Droit bancaire et droit des établissements de crédit Droit des suretés Droit commercial Entreprises en difficulté / Procédures collectives Droit de la concurrence Droit de la distribution Droit boursier / Droit financier Transport terrestre / maritime / Aérien Droit alimentaire Baux commerciaux et professionnels
Famille	Droit de la famille général Mariage couples pacs Droit des régimes matrimoniaux Droit du divorce Filiation Succession Autorité parentale Protection des mineurs et des majeurs vulnérables
Fiscalité	Droit fiscal général Fiscalité des particuliers Fiscalité de l'activité professionnelle / Fiscalité des entreprises Fiscalité du patrimoine Procédure fiscale TVA Droit fiscal international
International	International Européen Ohada Comparé
Pénal	Droit pénal général Droit pénal spécial Droit pénal des affaires Procédure pénale
Privé	Responsabilité civile Droit des obligations Contrat spéciaux Procédure civile Voies d'exécution / Procédure civile d'exécution Droit de la copropriété Droit Immobilier et baux d'habitation Droit rural Droit médical Droit de la consommation Droit des produits alimentaires Droit des assurances Droit des succession et libéralités Droit du sport
Profession juridique	Profession d'avocats Profession de notaires Profession de commissaires de justice Profession d'AJMJ Profession de magistrats
Propriété intellectuelle	Droit de la propriété intellectuelle Droit de la propriété littéraire et artistique Droit de la presse et des médias Droit du numérique

Public	Droit public général Droit des étrangers Droit de l'urbanisme Droit de l'environnement Droit public économique Droit des collectivités territoriales Marchés publics Droit de la fonction publique Responsabilité administrative Procédure administrative Droit électoral Sécurité intérieure
Social	Droit du travail Droit collectif du travail Droit de la protection sociale
Constitutionnel	Droit constitutionnel Droit de l'homme et des libertés fondamentales

TABLE 8: Zero-Shot

B Decoding strategies

The generation of tokens at each step will determine the result, and different decoding strategies need to be considered. For a more detailed theoretical and practical review, refer to (Labonne, 2023):

Greedy search is a simple and deterministic decoding strategy where, at each time step, the word with the highest probability given the context is selected. This means that for each word in the vocabulary, the model computes the probability $P(w|w_{<t})$, and the word with the highest probability is chosen. The process repeats until the end-of-sequence (EOS) token is generated. While this approach ensures the selection of the most likely word at each step, it often results in repetitive and generic text because it always follows the most predictable path.

Algorithm 1 Greedy Search

1: Initialize context
2: Initialize empty sequence W
3: **while** $w_t \neq \text{EOS}$ **do**
4: $\quad t \leftarrow t + 1$
5: $\quad \hat{w}_t \leftarrow \arg\max_{w \in V} P(w \mid w_{<t})$
6: $\quad$ Append $\hat{w}_t$ to W
7: **end while**=0

Random sampling introduces diversity into text generation by sampling words according to their probabilities as determined by the model. At each time step, the word is chosen by sampling from the distribution $P(w|w_{<t})$. This means that while high-probability words are more likely to be selected, low-probability words can also be chosen. The process continues until the EOS token is generated. However, this method can sometimes produce incoherent text, as it does not avoid the selection of very low-probability words, which might result in odd or nonsensical sentences.

Algorithm 2 Random Sampling

Initialize context
Initialize empty sequence W
while $w_t \neq$ EOS **do**
    $t \leftarrow t + 1$
    $w_t \sim P(w \mid w_{<t})$
    Append w_t to W
end while=0

Top-k sampling improves upon random sampling by limiting the pool of candidate words to the top k most probable ones at each step. After computing the probabilities $P(w|w_{<t})$ for all words, only the k words with the highest probabilities are retained. The probabilities of these k words are then renormalized to form a valid distribution, from which the next word is sampled. This approach balances diversity and coherence, as it allows for some variability in the selection while avoiding very unlikely words that could degrade the quality of the generated text.

Algorithm 3 Top-k Sampling

Initialize context
Initialize empty sequence W
Choose k
while $w_t \neq$ EOS **do**
    $t \leftarrow t + 1$
    Compute $P(w \mid w_{<t})$ for all $w \in V$
    Select top k words with highest probabilities
    Renormalize the probabilities of these k words
    $w_t \sim P_k(w \mid w_{<t})$
    Append w_t to W
end while=0

Nucleus or Top-p sampling adynamically adjusts the number of candidate words based on their cumulative probability. Instead of fixing k, it selects the smallest set of words V_p whose cumulative probability $\sum_{w \in V_p} P(w|w_{<t})$ meets or exceeds a threshold p. The selected words' probabilities are renormalized, and the next word is sampled from this adjusted distribution. This method adapts to different contexts by varying the size of the candidate pool, aiming to maintain a balance between generating high-quality and diverse text.

Algorithm 4 Top-p Sampling

Initialize context
Initialize empty sequence W
Choose p
while $w_t \neq$ EOS **do**
  $t \leftarrow t + 1$
  Compute $P(w \mid w_{<t})$ for all $w \in V$
  Select the smallest set V_p such that $\sum_{w \in V_p} P(w \mid w_{<t}) \geq p$
  Renormalize the probabilities of words in V_p
  $w_t \sim P_p(w \mid w_{<t})$
  Append $w_t to W$
end while=0

Temperature sampling modifies the probability distribution by scaling the logits before applying the softmax function, using a temperature parameter τ. The logits $u(w)$ are divided by τ to adjust their values: lower τ (less than 1) makes the distribution more peaked, increasing the likelihood of high-probability words, while higher τ (greater than 1) flattens the distribution, allowing for more random selections. This method provides a flexible way to control the balance between exploration and exploitation in text generation, enabling both more focused and more diverse outputs depending on the chosen temperature.

Algorithm 5 Temperature Sampling

Initialize context
Initialize empty sequence W
Choose temperature τ
while $w_t \neq$ EOS **do**
  $t \leftarrow t + 1$
  Compute logits $u(w)$ for all $w \in V$
  Adjust logits: $u'(w) \leftarrow \frac{u(w)}{\tau}$
  Compute adjusted probabilities: $P_\tau(w \mid w_{<t}) \leftarrow \text{softmax}(u'(w))$
  $w_t \sim P_\tau(w \mid w_{<t})$
  Append w_t to W
end while=0