UCLouvain

École polytechnique de Louvain

Explainability and machine learning (for text)

Author: **Romain Loncour**
Supervisor: **François-Xavier Standaert**
Readers: **Jérémie Bogaert, Christophe De Vleeschouwer, Antonin Descampe**
Academic year 2024–2025
Master [120] in Data Sciences Engineering

Abstract

Neural networks such as BERT and GPT-3 have achieved impressive performance across various natural language processing tasks, yet the reasoning behind their decisions remains opaque. As these models are increasingly deployed in sensitive domains, the need for transparent and trustworthy explanations becomes critical. This Master's thesis investigates whether these explanations are stable, meaning that different models trained under the same conditions generate similar explanations. In the first part, using synthetic datasets, this work observes that the length and the coherence of an input text impact that stability and that text that do not contain discriminant markers produce unstable explanations. In the second part, this work concludes that the difficulty of the task is decoupled from the stability of its explanations using real-world classification settings. Through this, the thesis contributes to the broader question of whether complex model decisions can be reliably explained using simple, human-understandable representations. Future research could look into the impact of the number of strong markers in a text on the explanation stability.

Acknowledgements

I want to first of all thank my girlfriend, Manon, for her continued support.

I also want to thank my family and especially my dad with whom I have had multiple interesting discussions.

I want to thank my friend Raphaël for his support and friendship.

I especially want to thank Jérémie Bogaert and François-Xavier Standaert for their insightful ideas, valuable feedback and generally for their active participation in the process of doing this Master's thesis.

Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

Contents

Part 1

Motivation & Context of the Study

Neural Networks like BERT (Devlin et al. 2019) and GPT-3 (Brown et al. 2020) have shown impressive performance on a variety of Natural Language Processing tasks, such as classification or generation of text. However, the way they reason and make decisions is an open research topic that is not fully understood to this day. Explaining how models make decisions becomes increasingly more important as Neural Networks are getting adopted in sensitive domains such as health, law, finance, and human resources. It is key that models do not make decisions based on biased features such as race, gender or age. Tracing the thought process behind a prediction is therefore critical. In this context and in the light of the new GDPR legislations (European Parliament et al. 2016), the need for transparency in the decisions made by the models becomes a priority.

The topic of this Master's thesis is a little more precise. This work is indeed interested in the explanations and whether they represent the model's thought process but more crucially, it takes a look at whether these explanations are stable, meaning that two models trained in the exact same way produce similar explanations. This Master's thesis will be discussing this need for stability, different ways to quantify where and why explanations may vary and how different parameters and tasks influence this variability. Concretely, this Master's thesis is split into two big parts. Firstly, it looks at the effect of different parameters on the variability of explanations using synthetic datasets, introducing a new visualization to observe this variability. Follows a discussion about this need for synthetic datasets and the difficulty that is inherent to it. Secondly, this work takes a look at non-synthetic use cases in order to study how the difficulty of the task impacts this variability.

This work contributes to a broader open question: *Can the decisions of complex neural networks be faithfully explained using simple representations?* Ideally, we

would like every model to provide a transparent explanation for each of its decisions. However, most methods used to explain predictions that are used have drawbacks. Developing new methods and studying current ones in order to improve their performance and to assess their effectiveness better is the approach that is followed in this work.

Part 2

Background

This chapter presents the prior knowledge necessary to understand the following work.

2.1 Transformer model

The transformer architecture was proposed by Vaswani et al. 2017 in their revolutionary paper *Attention is all you need*. According to Mienye et al. 2024:

> In 2017, Vaswani et al. 2017 revolutionized the approach to sequence processing tasks, particularly in NLP. This model diverges from traditional methods that relied on recurrent layers, instead employing self-attention mechanisms that allow for parallel processing of input data. The transformer is described by:
>
> $$\text{Query } Q = XW_Q, \qquad \text{Key } K = XW_K, \qquad \text{Value } V = XW_V \tag{2.1}$$
>
> $$\text{Attention}(Q, K, V) = \text{softmax}\Big(\text{Mask}\Big(\frac{QK^T}{\sqrt{(d_k)}}\Big)\Big)V \tag{2.2}$$
>
> where X represents the input data, W_Q, W_K, W_V are the learned weight matrices, and d_k represents the dimensionality of the keys, enhancing the model's ability to focus on different parts of the input sequence simultaneously.

From the definition of attention in Equation 2.2 flow two main families of transformers that change based on the *Mask*:

- **No Masking**: When no masking is done, every token in the input can attend every other token which is suitable for bidirectional tasks like text

translation or classification. This kind of self-attention appears in encoder-only or encoder-decoder transformer models (see more at Huda 2024). It is the mechanism powering models like BERT (Devlin et al. 2019).

- **Upper Triangular Masking**: An *upper triangular matrix* is used to obtain no attention on the upper triangle after the *softmax* operation. This prevents tokens from gaining access to the tokens that follow it. This form of causal attention is the base of autoregressive models for text generation and is usually present in decoder-only transformer models (see more at Huda 2024). It is the mechanism behind models like GPT-3 (Brown et al. 2020).

3Blue1Brown provides a great conceptual example of how self-attention works in 3Blue1Brown 2024, shown in Figure 2.1. In the sentence "`a fluffy blue creature roamed the verdant forest`", the noun `creature` is attended to by the adjectives `fluffy` and `blue` which modify its conceptual representation to create a more precise one (a blue fluffy creature in the Figure). Naturally, every other word in the input is attended to by every other word as well but maybe with less intuitive mechanisms.

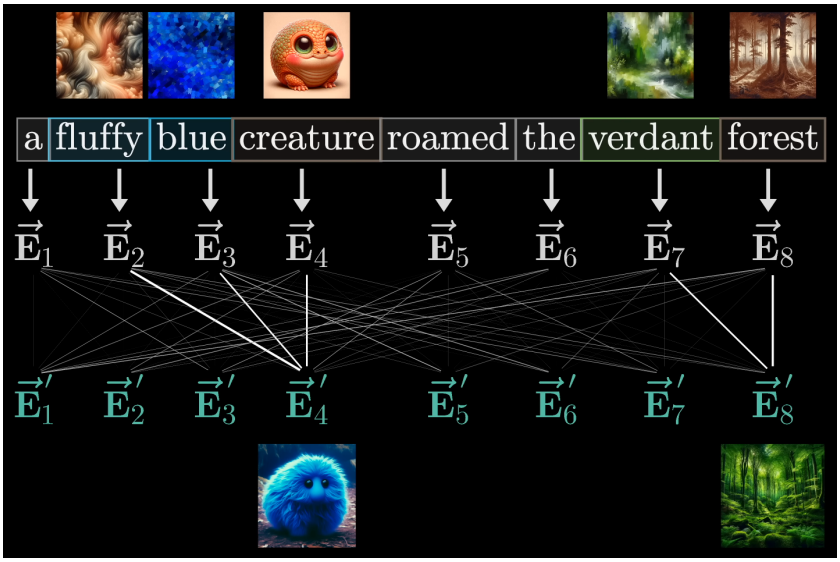

Figure 2.1: Self-Attention Example

2.1.1 BERT

In 2019, Devlin et al. 2019 introduced the **BERT** model. It is an encoder-only transformer model that can be used for sequence classification. Its architecture is defined in the paper and will not be detailed here. Usually, transformer-based

models are trained in two steps, there is a pre-training phase and a fine-tuning phase. Parthasarathy et al. 2024 mention that **pre-training** consists of training a model on a vast amount of unlabelled data in order to build general linguistic knowledge. In the case of BERT, the pre-training is a set of self-supervised operations where a big chunk of text is fed into the model to allow it to gain a good inner representation of language: how a sentence is constructed, how words are structured, etc...

Pre-training is often used as a starting point for fine-tuning, where the model is then adapted to a specific task using a smaller, task-specific dataset. Parthasarathy et al. 2024 say that "***fine-tuning*** *is a case of transfer learning which leverages the knowledge acquired during pre-training, adapting it to specific tasks with reduced computation time and resources.*"

Fine-tuned BERT models reached, at the time of publication, state-of-the-art performance on multiple benchmarks. Those benchmarked results are detailed in the paper.

2.1.1.1 RoBERTa

After BERT was introduced in 2019, Liu et al. 2019 observed that it was not optimally trained and that its performance could be greatly improved by doing better parameter searching and training it for a longer period of time. They therefore introduced the **RoBERTa** (Robustly optimized BERT approach) model which advanced the state-of-the-art on different benchmarks even further. RoBERTa shares the same architecture as BERT and contains roughly the same number of parameters[1]. Concretely, compared to BERT, they increased the size of the training set, chose different hyperparameters and removed one of the two self-supervised pre-training tasks. All these changes yielded an overall more robust model.

2.1.1.2 CamemBERT

After realizing that mono-lingual models seemed to outperform multi-lingual models on language specific tasks, Martin et al. 2020 introduced the CamemBERT model. They followed the exact same training process as Liu et al. 2019 for RoBERTa but used the French part of the Corpus OSCAR (Ortiz Suárez et al. 2019) creating a state-of-the-art model on 4 French tasks at the time.

2.1.2 Tokenization

Transformer models do not understand natural language. They only reason using numbers. Numbers are inputted into the model, they are then manipulated into

[1]There is a little difference as RoBERTa uses a larger vocabulary size which means its embedding matrix is larger.

its different layers to finally output other numbers. The tokenizer bridges the gap between the input natural language sentences and what the models can actually process as input. It transforms natural language sentences into an ordered sequence of sub-word units called *tokens* or *wordPieces* which are then converted into numbers called *ids*.

When constructing a tokenizer, the tokens are not hand-picked but the tokenizer is trained using complex algorithms such as explained in Xinying et al. 2021 for BERT's tokenizer. Tokens can be seen as sub-word units and work in pair with the transformer model. A model is trained to work hand-in-hand with its tokenizer. Once the tokenizer is trained, its output is fully deterministic meaning an input always produces the same output. Table 2.1 contains sentences tokenized using RoBERTa's tokenizer. We can observe that the whitespaces in the original sentences have been turned into the strange character '$\dot{G}$' which is placed at the beginning of some of the tokens[2]. Moreover, we can see from the second example that longer words (e.g. `Jimmie`) can be divided into multiple tokens. It is also important to note that even gibberish can be translated into tokens which means the model can operate on any input, even for example, when there is a spelling error.

Original Sentence	**Tokens**	**IDs**
`I am Jim.`	['I', 'Ġam', 'ĠJim', '.']	[100, 524, 2488, 4]
`I am Jimmie.`	['I', 'Ġam', 'ĠJim', 'mie', '.']	[100, 524, 2488, 23738, 4]
`jrcxdjk`	['j', 'rc', 'xd', 'j', 'k']	[267, 22393, 46140, 267, 330]

Table 2.1: Tokenized sentences using RoBERTa's tokenizer

2.1.3 Fine-tuning in this work

The fine-tuning process tailors a pre-trained model to a specific task using a small dataset. This section lists the different steps token to fine-tune a model and illustrate them with an example. Before fine-tuning, a task needs to be defined and a pre-trained model needs to be chosen. The task to learn is defined by a dataset containing labelled examples. The datasets used are balanced are typically contain 10.000 examples divided into 2 class in this work. Depending on the language of the dataset, the pre-trained model to fine-tune is CamemBERT for French and RoBERTa for English as my model to fine-tune. The first step of fine-tuning is to divide this dataset into 3 splits: training, validation and testing. The training set is used during the training phase so that the model learns to solve the task. The validation set is used during the training process to monitor overfitting and

[2]The '$\dot{G}$' character represents the whitespace for the RoBERTa tokenizer but it can be different for other models. For CamemBERT, the space is the underscore character.

to determine which values to assign to the hyperparameters. The testing set is used to measure the performance of our trained models on previously unseen data. Those 3 sets are then divided into batches of examples of equal size.

Mini-batch Stochastic Gradient Descent is a technique that is popular when training neural networks (Cristian 2024). Since computing the loss function of the entire dataset and then modifying model parameters is slow and ineffective for big datasets, people started dividing the dataset into smaller mini-batches of input texts, feeding it to the model, computing the loss on that subset and modifying the parameters with respect to the gradients of this loss. This process allows to make quick progress in decreasing the loss. Moreover, the randomness inherent to the selection of the texts in the batches allows to escape local minima efficiently.

This work uses a batch size of 16 examples because of hardware constraints and because smaller batch sizes tend to generalize better (Sarikaya 2024[3]). As a quick side note, the goal of this Master's that is not to obtain extremely high accuracies, therefore, finding the best set of hyperparameters is not a priority. Explaining predictions of models obtaining good accuracy is still important nonetheless because trying to explain the predictions of wrong or random models does not convey as much information. After the pre-processing of the dataset, the model is trained using the traditional training loop defined in Algorithm 1. The validation set can be used to tune the different hyperparameters: the number of epochs[4], the learning rate or parameters of the scheduler; and to monitor over- or underfitting during training. The trained model is then tested on the not-yet-seen test set using Algorithm 2 to measure its generalization performance. It is now ready to be used for inference.

More practically, model fine-tuning is done with the *Consortium des Équipements de Calcul Intensif* (CÉCI) infrastructure. The training loop is done on a GPU as it requires a lot of parallel computing, on an Nvidia Volta V100 from the *dragon2* server in Mons. The testing loop is done on CPU on an Intel Skylake CPU also from the *dragon2* server.

2.1.4 Randomness in training

When using a machine learning encoder-only model like BERT during inference, its output is deterministic, meaning there is no randomness at all in the whole inference process. Randomness only appears when training a model. This variability is something essential in that case as it allows the model not only to explore the parameter space efficiently and find good generalizable solutions but most

[3]This source surveys this issue and links to great papers.
[4]An epoch is an entire pass through the dataset.

importantly to be able to leave local minima in the non-convex loss function if it ever falls in them. This randomness comes from multiple sources:

- When constructing the mini-batches for each dataset split (train/val/test), texts are picked randomly. The texts in each split are the same but the batch in which they appear differs from one iteration to the next. This is the *Stochastic* in Stochastic Gradient Descent.
- During training, models use a regularization technique called dropout to reduce overfitting (Srivastava et al. 2014). **Dropout** consists of, during each forward pass, setting a certain proportion of weights in each layer of the model to 0, forcing nodes in the model to learn more robust features and not rely on other nodes too much as their presence is not guaranteed. Dropout only applies during training time keeping the model deterministic during inference.
- In order to specialize better to certain specific tasks, some layers can be added to the pre-trained model. In the case of classification, classification heads can be introduced. Since these newer layers are not pre-trained, the initialization of their weights is randomized.
- Due to the non-convexity of the loss function, the optimization process may converge to different local minima depending on the random initialization and training dynamics.

In programming, programs are rarely truly random. The randomness that they are using is in fact computed by a **Pseudo Random Number Generator** which produces a sequence of random integers based on an initial *random seed*. This seed is an integer that can be chosen by the programmer to manipulate the randomness in the program in order to obtain reproducible results even when the programs themselves require randomness.

2.2 Functionally Equivalent Models

The goal of this work is to observe and measure whether similar models - models that are trained in the exact same way (except for the random seed), that perform equally well on the test set and that predict concordantly on an independent set of examples (which they will have to explain next) - provide different explanations for their predictions. To achieve that goal, the notion of similar models needs to be mathematically formalized.

Functionally equivalent models are models that may differ in their architecture, in the way they are trained or in their overall parameter count but that

produce the same outputs for the same inputs. Similar to the methodology adopted in Bogaert et al. 2024, functionally equivalent models need to meet two criteria. Firstly, a set of functionally equivalent models should provide statistically similar performance on a test set. Secondly, they should all agree on their predictions on the texts they have to explain.

2.2.1 Statistically Similar Performance

As in Bogaert et al. 2024, equivalent models were generated by fine-tuning a pre-trained model n times using the same training set. Their performance was then measured on a previously unseen test set. A subset of models with the closest accuracies was selected. The goal is for the best (a) and worst (b) accuracies within this subset to be statistically similar. To assess whether the difference ($a - b$) is statistically significant, the Z-statistic is used, as described in Lehmann et al. 2008, which tests whether two proportions differ:

$$z = \left| \frac{a - b}{\sqrt{\frac{\frac{a+b}{2} * (1 - \frac{a+b}{2})}{n}}} \right| \tag{2.3}$$

Two proportions are significantly different for a z value bigger than 1.96 which corresponds to a p-value smaller than 0.025. The subset of remaining models have a performance on unseen data that is considered statistically equivalent.

2.2.2 Concordant Entries on Independent Texts

The trained models are required to produce the same predictions on the set of independent texts they have to explain. These models are used to study the stability of the explanations they generate on unseen data. Naturally, if the models produce differing predictions, the resulting explanations will also differ, but not for the right reasons. Therefore, the subset of models defined in Subsection 2.2.1 is selected to produce concordant predictions on these texts. Importantly, the predicted labels do not need to match the true labels. They need to agree with one another. rather, consistency among the models is the primary requirement. The following section presents scenarios in which many models may misclassify certain examples, and explains this is not an issue in the context of this study.

Indeed, it is possible that some texts fool all, or most, of the models. This might be the case in these two cases:

- Some texts are just extremely difficult to classify. For example, for the task of predicting if a movie review is positive or negative, the texts in Figure 2.2

is difficult to classify because the adjectives used are sometimes positive and sometimes negative. Some texts contain sarcasm (surveyed in Joshi et al. 2017), some contain double negations, etc...; which models can struggle with.

- Some texts are simply mislabelled in the dataset. Sometimes, entries in a dataset are assigned the wrong label during its construction leading to all models predicting the wrong label.

Models predicting the wrong labels is however not an issue. As previously mentioned, this work's topic of study is the stability of explanations more than the actual classification performance. All models may predict wrong but give the same reason for why they did so, which still lies in the scope of this work.

L'interprétation n'a rien d'extraordinaire mais est cependant illuminée par la présence de la trop rare et très belle Claude Farell dans le rôle de la comtesse Larish. Les premiers plans étaient prometteurs, mais l'histoire se révèle très vite inintéressante et filmée sans aucun rythme, c'est ennuyeux et vraiment pas terrible.

Figure 2.2: Text that all models struggle to classify as `positive` or `negative`

2.3 Interpretability

The use of Neural Networks is a big paradigm shift in computer science as models are not programmed, they are trained. Instead of following programmatic instructions designed by the developer, models exhibit behaviours that they developed during training while following well-defined mathematical rules to reach an optimum. This leads to two key insights. Firstly, models can represent very complicated patterns that cannot be explicitly defined by a programmer[5]. Secondly, as these models become more performant and the patterns they identify in data grow more complex, the number of parameters they require also increases. Moreover, the relationship between the model inputs and outputs becomes increasingly non-linear and difficult to grasp as well. This rise in complexity has an undesirable side effect: explaining these models' decision making becomes very difficult. So much so that they are sometimes considered as *black-box* meaning "*their inner working mechanisms are opaque, and the high complexity makes model interpretation much challenging.*" (Zhao et al. 2023). This lack of understanding can be problematic especially since these newer models can sometimes suffer from major issues:

- Sensitivity to artifacts in data, where models cheat the task by learning some undesirable patterns in the training set that are not generalizable to real-world tasks (Gururangan et al. 2018, Geva et al. 2019) or by learning statistical heuristics without understanding the language (McCoy et al. 2019);

[5] as the number of possible combinations of patterns is exponential.

- Sensitivity to adversarial attacks which is "*when we can cause the network to misclassify an image by applying a certain hardly perceptible perturbation*" as showcased by Szegedy et al. 2014;
- or, in general, bias and fairness issues (Weidinger et al. 2021).

Understanding, observing and quantifying "*may help calibrate user trust in high-stakes applications. In domains like health, law, finance, and human resources, an end user may not trust a model if it only provides a prediction but no explanation.*" according to Lyu et al. 2024.

2.3.1 Core Definitions

Interpretability has been defined by Doshi-Velez et al. 2017 as "*the ability to explain or to present in understandable terms to a human*". An interpretable model would therefore need to be able to explain its decisions. Moreover, they claim that, "*while it is not the only desiderata, as we may for example want to promote fairness, privacy or reliability of the model, it can assist in qualitatively ascertaining whether they are met.*" The needs for interpretability are multiple. Firstly, understanding these models would help provide guarantees on model performance, increasing confidence in its decisions and allowing adoption even in sensitive domains. Secondly, understanding model weaknesses would help mitigate them.

Explainability is another important concept to keep in mind as it is what this work dives in in this Master's Thesis. Both concepts look intuitively similar but there is a fundamental difference. On the one hand, "*a model is **intrinsically interpretable** if a human can understand the internal workings of the model, either the entire model at once or at least the parts of the model relevant to a given prediction. This may include understanding decision rules and cut-offs and the ability to manually derive the outputs of the model*". On the other hand, "*a model is **explainable** if we find a mechanism to provide (partial) information about the workings of the model, such as identifying influential features. We consider a model's prediction explainable if a mechanism can provide (partial) information about the prediction, such as identifying which parts of an input were most important for the resulting prediction or which changes to an input would result in a different prediction*" (definitions by Kästner 2021). In other words, explainability focuses on explaining specific model decisions while interpretability is used to explain how a model generally makes any prediction.

2.3.2 Faithfulness & Plausibility

There exist two main criteria that every good explanation needs to fulfil: faithfulness and plausibility. In their survey on the topic, Lyu et al. 2024 defines

faithfulness as "*the extent to which an explanation accurately reflects a model's reasoning process*". It is arguably the most important metric in explainability as an explanation that doesn't reflect the model's inner working is utterly useless and can be misleading. This need for faithfulness is particularly crucial in high-stakes scenarios like healthcare, finance and legal systems (Agarwal et al. 2024). Jacovi et al. 2020 defines **plausibility** as "*how convincing the interpretation is to humans*". This means that the provided explanation needs to be sufficiently easy to understand by a target audience. There is a notion of subjectivity in this concept as two people with different backgrounds may not find the same explanation plausible. Someone well-versed in the field of LLMs may be able to understand more complex explanations than someone who is not.

To better visualize the dynamics between these two concepts, let's imagine the following hypothetical task: ***predicting whether the animal in the image is a cat or not***. This is a very easy task for any human having seen a cat once in their life but it can be complicated for Neural Networks. Figure 2.3 shows three different ways to explain a correct model prediction. On the left, there is an explanation that is completely faithful but not plausible. It is the set of all model parameters. By definition, it perfectly describes how the model works[6] but it is impossible for humans to understand it, especially when the number of parameters is of the order of magnitude of a model like BERT[7]. On the right, we could have an explainer method that always outputs the same textual explanation: "*This animal has whiskers so it is a cat.*". This provided explanation is perfectly understandable by a human and is correct but does not necessarily reflect how the model reasons internally. It makes sense but it simply does not represent the inner working of the model most of the time. In the centre, we can observe a third explanation[8], an attention map that shows which regions of the image are considered important towards the classification by the classifying model. This third method is somewhat plausible as it is easy for a human to understand which parts of the image are important and it is argued to be faithful by its creators. We therefore observe a trade-off between both metrics. There exists a multitude of methods that contain that trade-off as well such as SmoothGrad (Smilkov et al. 2017), LIME (Ribeiro et al. 2016b) or Grad-CAM (Selvaraju et al. 2017). Based on this example, we conclude that a trade-off between the two metrics is necessary. Ideally, we want a method that provides faithful explanations while staying as plausible as possible.

[6]As it is the model itself, it must naturally contain the explanation as it just produced it.

[7]$\text{BERT}_{\text{BASE}}$ contains 110 million parameters and $\text{BERT}_{\text{LARGE}}$ contains around 340 million parameters

[8]This middle explanation was generated by Bach et al. 2015 using the LRP method (explained thoroughly in the case of NLP in Subsubsection 2.3.4.1)

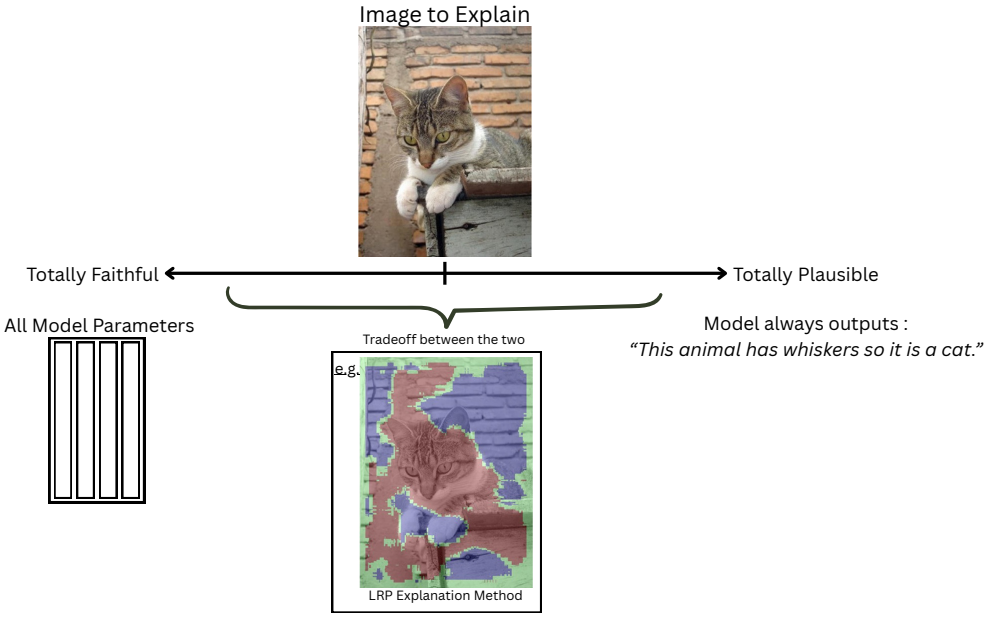

Figure 2.3: Three explanations for the classification of an image

2.3.3 Sensitivity to Randomness

The main concept studied in this Master's thesis is the notion of sensitivity to randomness introduced by Bogaert et al. 2024. The **sensitivity to randomness** can be defined as the extent to which explanations generated by models trained using the same procedure differ. Bogaert et al. have shown that changing the random seed used during model training[9] sometimes makes those models produce very different explanations even though they are functionally equivalent. They showed that, as the number of classification models trained with different seeds increases, the number of different generated explanations increases as well. They make the hypothesis that if the explanations follow a uniform distribution, selecting only one of them to illustrate a decision becomes an arbitrary choice. Each individual explanation may look plausible as some tokens that seem intuitively related to the classification task are usually highlighted but this set of highlighted tokens differs a lot from one run to the next. Figure 2.4 shows two explanations generated by functionally equivalent models but that differ significantly.

Based on this observation, they claim that this variability in explanations needs to be measured and characterized, which is my goal in this work. Still, two different models producing largely different explanations does not mean that the method used is not faithful. We could imagine that two different explanations produced

[9]See 2.1.4 for a detailed summary of where the randomness appears in training.

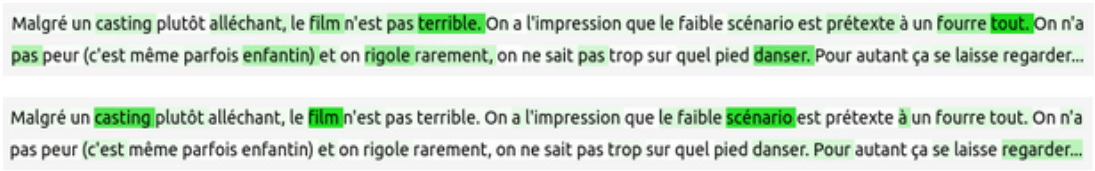

Figure 2.4: Explanations from functionally equivalent models that differ

by two different functionally equivalent models could both be faithful at the same time as there may be multiple ways to explain a single decision. Wiegreffe et al. 2019 claim in the context of using attention weights as an explanation : "*Just because there exists another explanation does not mean that the one provided is false or meaningless, and under this definition the existence of multiple different explanations is not necessarily indicative of the quality of a single one.*". For example, Figure 2.5 shows two explanations for one text, one that is based on the vocabulary employed and one that is based on the punctuation.

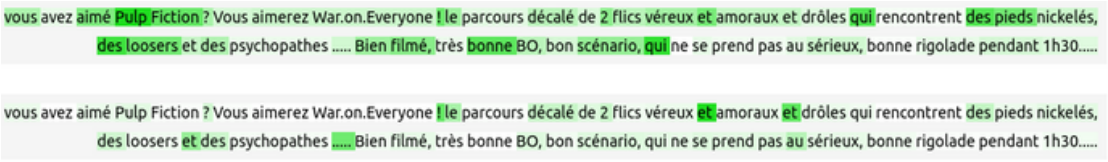

Figure 2.5: Explanations from functionally equivalent models that differ

Interestingly, Wu et al. 2021 performed, among other things, an experiment to test whether the attribution score attributed by four different explanation methods were impacted by the random seed used during model training. They only considered two random seeds and did not select functionally equivalent models but concluded that "*random initializations affect (their) results but only to a limited extent. Furthermore, consistencies in results of longer sentences seem to suffer more from random initialization compared to shorter ones.*". This experiment was not conducted at a large scale but shows that other researchers have started to look into the topic of sensitivity to randomness.

2.3.4 Attribution-Based Explanation Methods

Zhao et al. 2023 define feature attribution methods in the following way :

> Feature attribution methods aim to measure the relevance of each input feature (e.g., words, phrases, text spans) to a model's prediction. Given an input text x comprised of n word features $\{x_1, x_2, ..., x_n\}$, a

> fine-tuned language model f generates an output $f(x)$. Attribution methods assign a relevance score $R(x_i)$ to the input word feature x_i to reflect its contribution to the model prediction $f(x)$.

They then mention that there are 4 big families of attribution-based explanation methods. The focus of this work is on the one that contains **Decomposition-Based Methods** which "aim to break down the relevance score into linear contributions from the input". More precisely, the backpropagation-based method known as **Layer-wise Relevance Propagation** (LRP) is used to compute an attribution score for every input token with respect to the model's output prediction.

Giving an importance score to every part of the input makes a lot of sense intuitively as we expect the model to use some part of the input more than others. For example, for the following movie sentiment analysis task, we can easily tell that the text: "`This movie was good`" is positive and that the most important word towards that prediction is the adjective "`good`". The rest of the text intuitively shouldn't contribute towards the prediction as the tone is neutral and doesn't contain any information. In a perfect world, the attribution method would give a low score to the words "`This`", "`movie`", "`was`" and a high score to the word "`good`". It is an open question whether large model's inner mechanisms can be summarized using such a simple representation but this work makes the assumption that the attribution score distributions contain meaningful information.

2.3.4.1 LRP method

Layer-wise relevant propagation (LRP) is not an explanation method per se. It is a framework containing a set of constraints proposed by Bach et al. 2015. The general idea is to start from the output of the model and to backup through the layers to see which parts of the input were most useful towards the outputted prediction. LRP in its most general form assumes that the classifier can be decomposed into several layers of computation.

> The first layer are the inputs, (...) the last layer is the real-valued prediction output of the classifier. LRP assumes that we have a Relevance score $R_d^{(l+1)}$ for each dimension $z_d^{(l+1)}$. The idea is to find a Relevance score $R_d^{(l)}$ for each dimension $z_d^{(l)}$ for the vector z at the next layer l which is closer to the input layer such that the following equation holds:
>
> $$f(x) = ... = \sum_{d \in l+1} R_d^{(l+1)} = \sum_{d \in l} R_d^{(l)} = ... = \sum_d R_d^{(1)} \tag{2.4}$$
>
> Iterating from the last layer which is the classifier output $f(x)$ down to the input layer x consisting of input tokens yields a relevance score

distribution. Each relevance score R is the local contribution to the prediction function $f(x)$.

Notations are shown in Figure 2.6. Since LRP is a framework, any solution satisfying the set of constraints will be considered a layer-wise relevance propagation method. Multiple implementations of LRP are therefore possible.

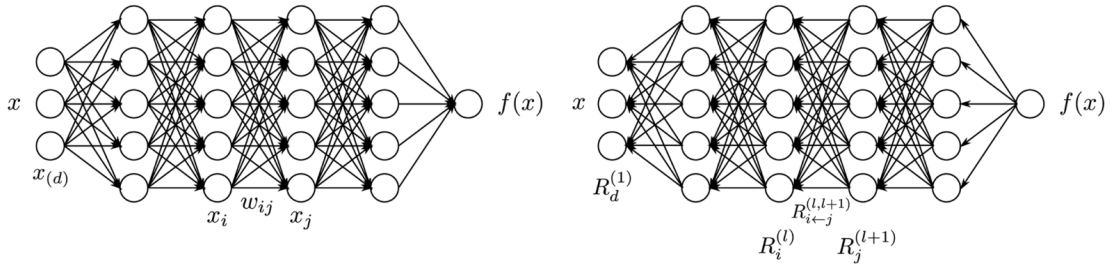

Figure 2.6: Multilayer Neural Network annotated with the different variables and indices. Left: forward pass. Right: backward pass (source: Bach et al. 2015)

To summarize, the total relevance flowing through the model is constrained to be preserved from one layer to another. The total relevance incoming to a node must be equal to the outgoing relevance. The way it is redistributed depends on the implementation chosen. In some layers, e.g. linear layers, redistribution is easy, but in others, e.g. when there is a non-linearity, it gets trickier. Since relevance is hard to propagate through non-linear layers, better ways of doing it are constantly proposed (Binder et al. 2016, Voita et al. 2019). Notably, Chefer et al. 2021 have proposed a class-specific LRP implementation for self-attention models like BERT. The method is tailored for classification tasks in vision but also for NLP. The authors provided code for BERT which Bogaert et al. 2024 ported to CamemBERT which was then adapted to RoBERTa. This implementation is used during this work when generating explanations. Moreover, LRP back-propagates down to the input tokens but looking at the relevance of entire words instead seems more appealing and intuitive. Therefore, if a word is divided into multiple tokens, their attribution scores are combined by taking their mean.

2.4 Synthetic Datasets

One of the main topics studied in this work is the impact of different parameters towards explanation stability. To try to answer this question, a very thorough experimental setup needs to be followed. As in most scientific study, the only parameter that can vary during an experiment needs to be the target parameter so it is very important to control everything else. However, when working on

NLP tasks, there are a lot of different ways sentences can differ and controlling everything is a difficult task. There is therefore a need for synthetic datasets for which most of the parameters can be controlled.

2.4.1 Data Artifacts

Neural Networks are trained to recognize statistical patterns in data. Sometimes, these patterns are not inherent to the task but what is called an **artifact**. An artifact can be seen as an undesirable hint that helps the model predict correctly. It is problematic since this hint is not present in real data and the model will therefore not be able to use it in real use cases, making the model not generalize well. Artifacts make us overestimate the quality of our model's predictions. One kind of artifacts named annotation artifacts is presented in Gururangan et al. 2018. They showed that for a Natural Language Inference task in which the model has to, given a pair of sentences, a premise p and a hypothesis h, determine whether or not p semantically entails h, a model could actually get a pretty good accuracy without seeing the second sentence. The authors say that this kind of shortcuts is common in datasets, artificially inflates model performance and needs to be carefully studied. Moreover, Gardner et al. 2021 argue that "for complex language understanding tasks, all simple feature correlations are spurious". They consider that every single hint that is too simple should be deleted from training data.

To give an example of an artifact in a dataset that is used in this work, Bogaert et al. 2023 note that the predictor considers the token `</s>` as very important when classifying one of the two classes. This intuitively doesn't make sense and is explained by the fact that input texts from that class tended to be quite long, longer than the input window and were abruptly ending with the token `</s>`. The model used the length of the text as an indicator for classification instead of understanding the input text.

2.4.2 Necessity of Synthetic datasets

Synthetic dataset generation can be difficult but is absolutely necessary in the context of this work. One of my research questions is to try to determine whether the LRP method is faithful. To do so, every single parameter in that experiment must be controlled so that the only possible variability stems from this LRP unfaithfulness or from other studiable and quantifiable parameters. However, Natural Language datasets like *RTBF-InfOpinions* or *AlloCiné* (see 3.3.1) are difficult to study systematically as texts can vary greatly from one to the next. Texts may differ by their topic, their length or by how obvious they are to classify. For example, the texts in Table 2.2 can be difficult to classify, even for humans. But a bigger problem is that, even if a program was able to get perfect prediction

accuracy on those texts, it would be hard to pinpoint why it made such predictions, what parameters contributed to it. Indeed, in those two texts, it is not clear why one is an opinion and why another is an information. Is it the presence or absence of a single word ? The length of the text ? Its structure ? The tense of the verb ? In these conditions, it is not possible to verify whether LRP is faithful as there is no verifiable ground truth to compare the model explanations to.

Sentence	**Label**
Ils reviennent sur le scandale fiscal "Panama Papers" ainsi que sur la lutte contre la fraude en Belgique.	`Opinion`
On ignore le montant emporté par les malfrats qui ont pu prendre la fuite. Aucun blessé n'est à déplorer.	`Information`

Table 2.2: Some InfOpinions texts with their labels

Since the reason for a certain prediction can be hard to definitely pinpoint, generating synthetic data is necessary to try to control every other parameter to look at the one we are interested in. In this case, whether the explanation method works well.

2.5 Metrics & Visualizations

My goal in this work is to quantify the variability of explanations. This section presents the different metrics and visualizations used in this work.

2.5.1 Metrics

2.5.1.1 Mean Correlation With the Mean Explanation (MCWME)

The main metric used in this work is the **Mean Correlation With the Mean Explanation (MCWME)** presented in Bogaert et al. 2025. As a reminder, multiple models are trained on the same task by changing the random seed in order to know how much the explanation varies from one random initialization to the other. Each explanation is a list of attribution scores representing how much each part of the input contributed to the prediction. The way the MCWME tries to quantify the variability in explanations is by measuring by how much each explanation diverges from the mean explanation and aggregating these values together. One MCWME is computed for each input text. It is estimated in the following way: After collecting the explanations from all models, each one is correlated with the Leave-One-Out mean. This means that the mean explanation does not contain the explanation it is compared to in order to avoid bias.

he goal is to compute the mean and confidence interval of this distribution. However, since the CWME distribution is typically not normally distributed, the *Fisher Z-transformation* is applied to stabilize the variance and approximate normality prior to computing summary statistics.

As a result, one correlation score is obtained per explanation which is essentially a distribution of **correlations with the mean explanation (CWME)**. The goal is to compute the mean and the confidence interval of this distribution but there is one small problem: the distribution is usually not normally distributed. The "Fisher Z transformation" (Silver et al. 1987) is therefore applied:

$$F(\hat{\rho}) = \frac{1}{2}\ln(\frac{1+\hat{\rho}}{1-\hat{\rho}}) = \operatorname{arctanh}(\hat{\rho})$$

which projects the correlation samples in a space where they are normally distributed. In this new space, the average Fisher value $\overline{F}$ is computed, as well as its sample variance $\hat{\sigma}^2$. A confidence interval on the estimation of $\overline{F}$ (e.g., 96%) is obtained by adding or removing $2\frac{\hat{\sigma}}{\sqrt{m}}$ to $\overline{F}$. Applying the inverse function $\rho = \tanh(F(\rho))$ finally leads to 96% confidence interval for the average correlation:

$$\Big[\tanh\Big(\overline{F} - \frac{2\hat{\sigma}}{\sqrt{m}}\Big); \tanh\Big(\overline{F} + \frac{2\hat{\sigma}}{\sqrt{m}}\Big)\Big]$$

The procedure of computing the MCWME per text is detailed in Appendix C. Moreover, the higher the MCWME, the better as this means all models agree with the mean, which is a proxy for measuring that they agree with each other. Initially, the goal was to aggregate the MCWME of different texts together but it didn't seem like a great idea to combine poorly estimated values with each other as the confidence intervals would become really large and uncertain. Therefore, rather than representing the results with a single, potentially poorly estimated value, the analysis focuses on identifying general trends and patterns that emerge across texts.

Since the MCWME is a mean of correlations, it lies within the $[-1, 1]$ interval. When it equals 1.0, it means that every explanation is perfectly correlated with the mean explanation. This only happens if all explanations are the same. In contrast, when it equals 0.0, it means that every explanation is completely random and does not have any structure. All of the MCWME in this work lie within this interval of $[0.0, 1.0]$ so the plots generated in this work are limited to that interval.

A critic of this metric could be the use of the correlation as its similarity measure between an explanation and the mean explanation. First of all, a linear metric may seem simplifying but it is argued that it is not necessary to look for higher-level relationships. First-order relationships are already hard to find and

looking at higher levels would most likely yield less interpretable, more noisy values. Moreover, this simplifies the intuitive meaning of the MCWME to a simple metric of explanation variability. Most importantly, this stems from the open question of whether it is possible or not to summarize predictions of models that are so complex using very simple and intuitive methods.

2.5.1.2 Top-k Word Explanations

As shown in Bogaert et al. 2025, keeping only the k highest attribution scores ($0 \leq k \leq n = \text{len}(\text{text})$) in an explanation by following Equation 2.5 before computing the MCWME can be interesting for 3 simple reasons. Firstly, providing shorter explanations seems even more plausible than longer ones as it is easier to pinpoint which words are most important. Secondly, setting all the other attributions to 0 is an easy way to remove close-to-0 noisy attribution scores. Thirdly, keeping a constant k value when comparing texts of different lengths allows for a fairer comparison.

$$a_{s,w} = \begin{cases} a_{s,w} & \text{if } a_{s,w} \in \text{TOP}_k(a_{s,:}) \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

2.5.2 Visualizations

The examples given in this subsection are fictitious. Their purpose is to illustrate how the visualizations function. They do not contain any meaningful information and no conclusion will be drawn from them in this work.

2.5.2.1 Attention Map

Attribution-based explanation methods produce floating-point values called attribution scores for every word inputted into the model. An effective way to visualize this distribution of attribution scores is with an attention map. An attention map is constructed by displaying the input text and highlighting the different words with a colour whose luminance is proportional to each word's attribution score. Figure 2.7 shows an attention map for the text "`John is going to the beach tomorrow.`". It allows to visually see that the word `beach` is considered the most important in the text while other words like `tomorrow.` and `John` are attributed only a little bit of importance.

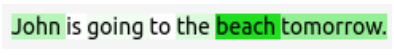

Figure 2.7: Example of an Attention Map

This visualization can be used for individual distributions or when taking the mean of multiple distributions coming from different models. This work uses the attention map visualizations from the *Captum*[10] Python library. The distribution plotted is normalized before being showed in order to see the relative importance between the highest and lowest attributions. For a distribution of attribution scores a of length n, the normalized distribution a' becomes:

$$a'_i \leftarrow \frac{a_i - \min(a_:)}{\max(a_:) - \min(a_:)} \forall i \in [0, n] \tag{2.6}$$

Sometimes, words are divided into different wordPieces by the tokenizer (see Subsection 2.1.2). Each wordPiece is given an individual attribution score by the explanation method. To obtain a plausible representation, values attributed to wordPieces are aggregated together by taking the mean of their attribution scores (AS). Table 2.3 shows an example of this for the text: `"I am Jimmie."`.

	Before	**After**
wordPieces	'I', 'am', 'Jim', 'mie', '.'	'I', 'am', 'Jimmie.'
Attribution Scores	[0.7, 0.1, 0.4, 0.6, 0.2]	[0.7, 0.1, 0.4]
Normalized AS	[1.0, 0.0, 0.5, 0.83, 0.16]	[1.0, 0.0, 0.5]

Table 2.3: Example of Aggregation of wordPieces

[10]Documentation at captum 2025.

2.5.2.2 Error Bar Graph of the MCWME per text

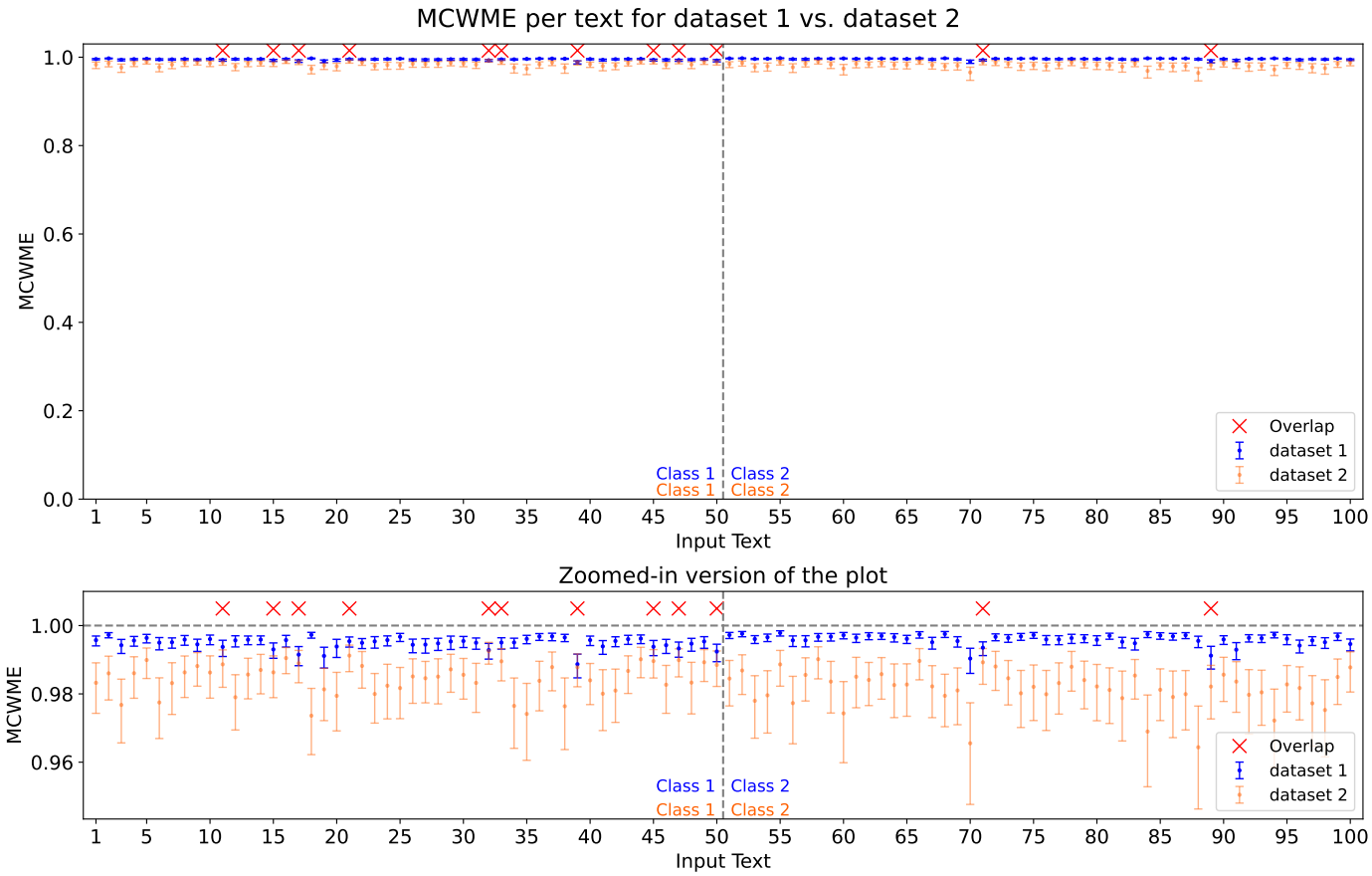

Figure 2.8: Example of an Error Bar Graph of the MCWME per text

The main visualization of this work is the **Error Bar Graph of the MCWME**, as shown in Figure 2.8. Since the MCWME is computed one text at a time, a visualization that displays a lot of them at once is necessary to see if there are some interesting trends happening in the data. The x-axis corresponds to the different texts being explained. Usually, the number of sentences to explain is 100 texts, 50 per class. A vertical dashed line separates both classes of explained texts. Sometimes, there are stark differences in the stability of explanations of one class compared to the other and marking a clear distinction between them makes interpretation clearer.

The y-axis represents the MCWME, the central metric used in this work (see Subsubsection 2.5.1.1). In all the conducted, experiments, it always falls within the $[0.0, 1.0]$ interval, which are the bounds previously chosen for this plot. This visualization allows to plot the results of multiple datasets (typically 1 or 2)

simultaneously for easy text-by-text comparisons. Before plotting, pairings of texts need to be established. Those pairs can be predefined, based on corresponding texts from both datasets, or they can be completely randomized. It will be mentioned in each experiment how the pairings are formed. At each x-coordinate is therefore plotted two confidence intervals and if they overlap, a little red cross will be drawn above them. In some experiments, the values in the graph lie on the upper part of the graph, in an interval between $[0.95, 1.0]$, making interpretation difficult. In those cases, a zoomed-in version of the plot is placed at the bottom of the graph. At times, this representation is used to show the top-k MCWME, it will be clearly mentioned when it is the case.

This representation condenses a lot of information in a very small space and it is important to keep in mind that a strong aggregation is done that hides some of the intricate patterns of the explanations. We could for example not be able to see some clusters of explanations even if there were some. To get some more precise insights, it is necessary to couple this visualization with other ones.

2.5.2.3 Scatter Plot of the Correlation With the Mean Explanation

This visualization was introduced in Bogaert et al. 2025. An example of it is shown in Figure 2.9. The number of words used in an explanation can be an interesting parameter to study to get a grasp of how the explanations are structured. In a perfect world, this metric would remove the attribution scores given to non-important words that are usually close to 0 but oscillate noisily close to it and retain only the high, relevant ones. This representation relies heavily on the *top-k words explanation* metric (see 2.5.1.2). The first step is to compute the distribution of top-*1* through top-n correlations with the mean explanation. On the x-axis is the number of words used per explanation. Naturally, there are as many x-coordinates as there are words in the text. The y-axis represents the CWME and its bounds are between 0 and 1. On the left, there is a scatter plot of all the correlations with the mean explanations for different values of k. On the right is shown the mean correlation with the mean explanation with its confidence interval. The goal of this representation is to identify outliers and to recognize patterns regarding how explanations are structured. The colours in the plot are used to differentiate one k value from the next and do not convey any information. Using this representation, only one text can be plotted at a time. That text is chosen randomly through the pool of possible ones.

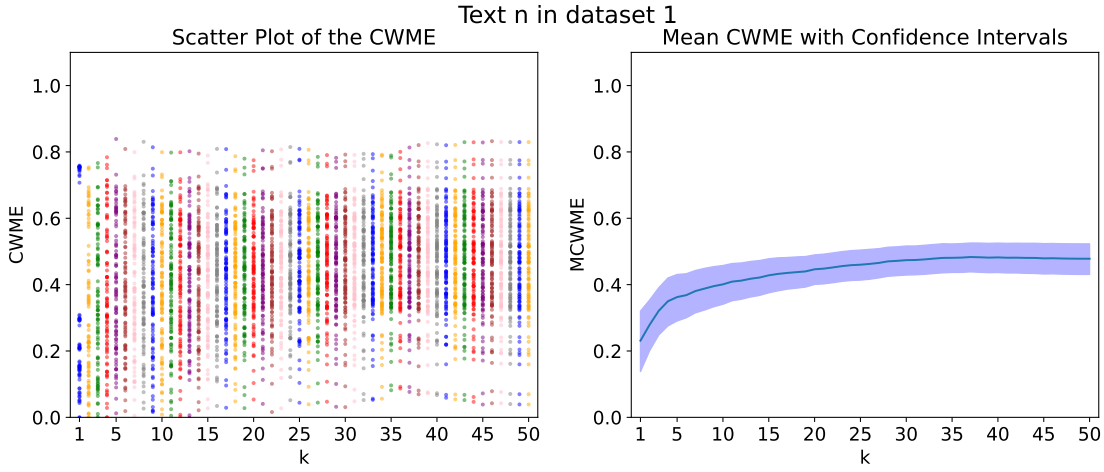

Figure 2.9: Scatter Plot of the Correlation with the Mean

Part 3

Methodology

3.1 Goal of the Work

As pointed out by Bogaert et al. 2024, when changing the random seed, models trained in the exact same way tend to produce different explanations. This behaviour seems curious and deserves to be characterized more formally in the hopes of understanding better where those variations come from. I divided this work in two parts. The first one uses synthetic datasets to gain insight into which parameters influence this stability. The second one looks at real-world datasets and examines how explanation stability differs based on the intuitive difficulty of a task. More broadly, I ask myself the two following overarching questions:

1. **Which parameters influence the stability of explanations ?**
2. **Does the variability to randomness depend on the difficulty of the task ?**

The main questions being quite broad and hard to definitely solve, I will fragment my research into a series of experiments that may provide hints towards that solution. I list here some questions whose purpose and implementation is detailed afterwards. For the first part, the different questions are:

1. What is the importance of the period in the attention distribution ?
2. Is the LRP method faithful ?
3. How does the length of the input text impact explanation stability ?
4. Does the presence of context influence the stability of explanations ?
5. How are explanations distributed when one class is defined as not having any distinctive sign ?

For the second part, the different questions are:

1. Is there a significant difference in variability of explanations between the `information` and `opinion` explanations in *InfOpinions* ?
2. Is there a significant difference in variability of explanations between the `positive` and `negative` explanations in *AlloCiné* ?
3. Is there a significant difference in variability of explanations between the *InfOpinions* and the *AlloCiné* tasks ?

The results of those experiments are presented and analysed in the Results Chapter.

3.2 Pipeline of Operations

In this section, I present step-by-step the pipeline of operations that is used for every experiment conducted in this work. It is greatly inspired by Bogaert et al. 2024.

When conducting an experiment, the first thing to do is to define a classification task where the goal is to predict a label based on an input text. I therefore choose an existing dataset or build one. The way they are chosen or built depends on the task and will be detailed for every experiment. The datasets I am working with only have two classes but this process could totally be extended to more classes without loss of generality. This dataset is divided into an 80/10/10 training/validation/testing split. The training set is used for training. The validation set is used to monitor that the models train properly and the testing set is used to compute the model's accuracy on unseen data. Once the dataset has been split, I can take a pre-trained encoder model from HuggingFace[1] such as RoBERTa (see 2.1.1.1) or CamemBERT (see 2.1.1.2) and fine-tune it on the current task using the *train* split. The training is comprised of 1 epoch of going through the entire *train* set using Stochastic Gradient Descent. I am using Adam as optimizer and a linear scheduler[2]. The hyperparameters have been chosen arbitrarily and do not matter that much. I want the models to perform well but them having a very high accuracy is not critical. This fine-tuning procedure is applied n different times to the pre-trained model with different random seeds to produce n fine-tuned models. The testing accuracy of all models is also computed. This process of model training is shown in Figure 3.1.

[1] HuggingFace is a platform which hosts a lot of machine learning models, datasets, etc... [`https://huggingface.co/`]

[2] Using a scheduler with only 1 training epoch does not do anything but I used more epochs beforehand so I kept it.

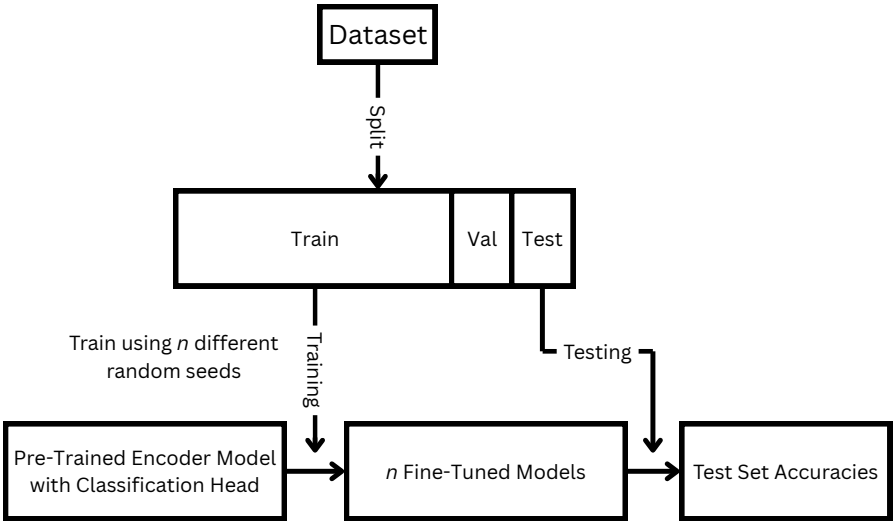

Figure 3.1: Fine-tuning n models

Based on the first criterion defined in Section 2.2 which states that models should have statistically similar accuracies, I only keep a set of m models with statistically similar performance on the test set as shown in Figure 3.2.

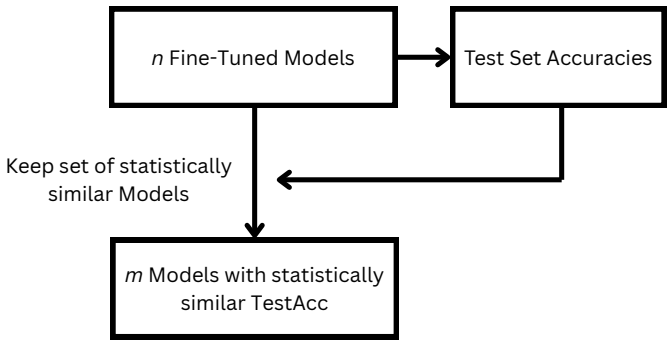

Figure 3.2: m models with Statistically similar Test Accuracy

With each of these m models, I then produce an explanation for each of the α texts to explain. These texts have been sampled and removed from the dataset before the train/val/test split and have therefore not yet been seen by the models. The number of sampled texts, their size and their label depends on the task and will be detailed in each dataset's description. My m models are used for prediction on these texts and the LRP explanation method (2.3.4.1) is then applied to explain each model's prediction. Based on the predictions on these unseen texts, I only keep the subset of models that all predict the same labels. This set of labels doesn't necessarily need to be the true labels, I just want the models to agree on

their output. Indeed, I am more interested about gaining insights into how the models reason than about whether they have perfect accuracy. This is the reason why I do not deploy some very complicated training mechanisms. Accuracy is almost secondary in this context. I obviously want to obtain models that perform significantly better than a random choice baseline[3] as otherwise their reasoning wouldn't be interesting. I therefore end up with l functionally equivalent models that predict the same labels on the α texts to explain. This set of explanations can be freely compared together. This whole step of determining concordant entries on the sentences to explain is shown in Figure 3.3

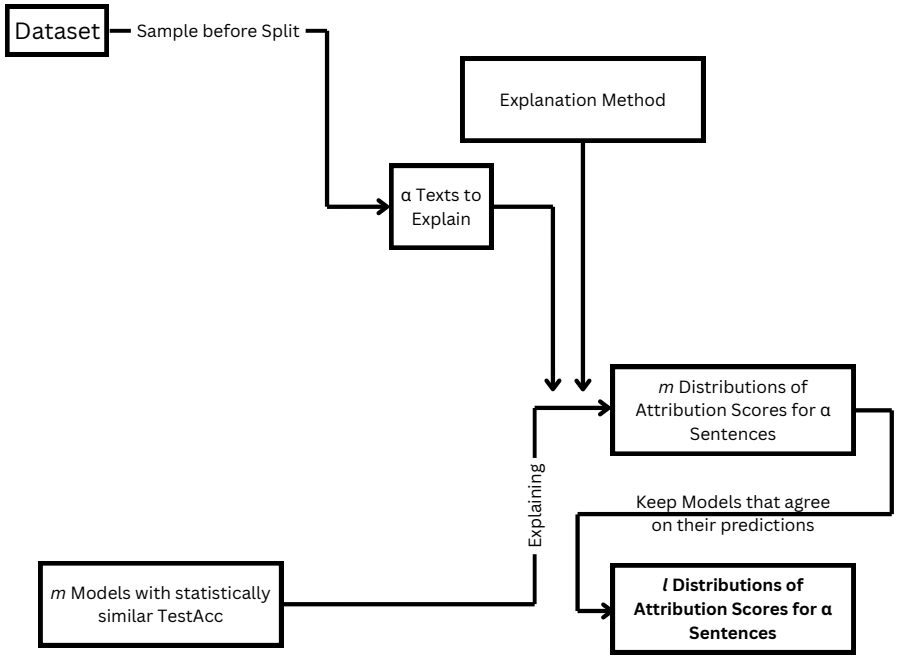

Figure 3.3: l functionally equivalent models

The whole pipeline of operations comprised of the 3 previous parts is shown in Figure 3.4.

[3]which yields a 50% test set accuracy.

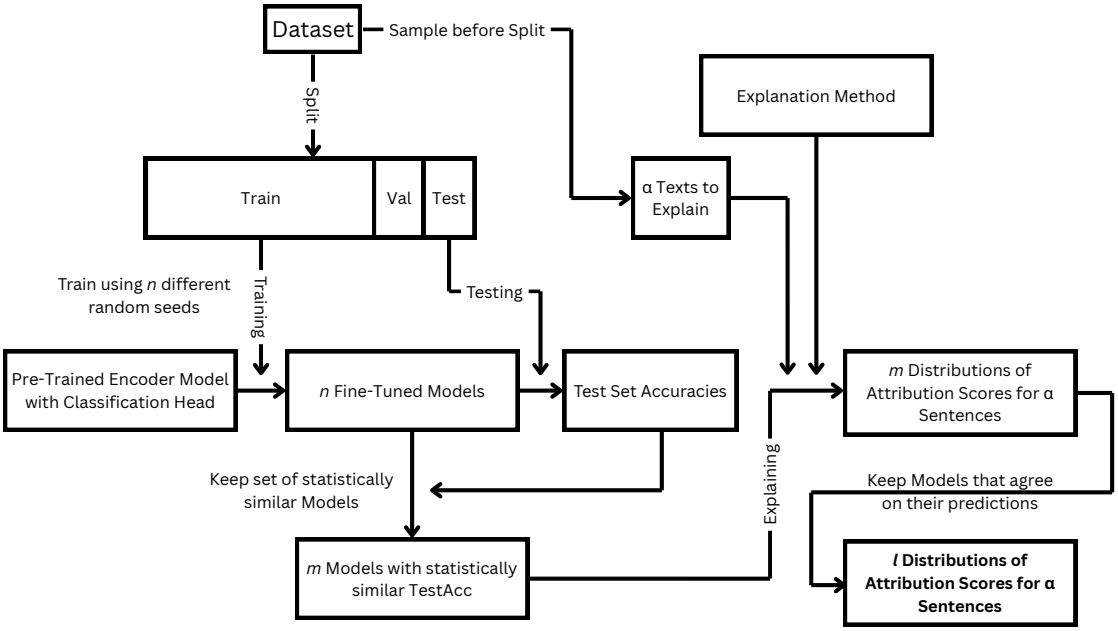

Figure 3.4: Pipeline of Operations

Across different tasks, I want to have the same number of functionally equivalent models to estimate the variation in exactly the same way. Increasing the number of models will by construction decrease the confidence interval. This variability is something that I use to draw conclusions so keeping it constant is important. Therefore, I keep 100 functionally equivalent models per task.

I would like to note that I am using both CamemBERT and RoBERTa in this work, depending on the language of the task. I do not change my methodology and consider both models to provide explanations and predictions of similar quality. Moreover, their number of parameters are very close and, since their architectures are essentially identical, I am using the same explanation method implementation for both. The only practical difference between them is the tokenizer, which does not manage whitespaces in the same way.

3.3 Datasets

Datasets are the basis of this work, they define the task that has to be solved by the Neural Networks. All the datasets in this work have been designed for classification, they are a collection of labelled Natural Language sentences. In this work, I refer to an *input text* as a collection of words given as input to a model and that can be composed of one or multiple sentences. It is therefore possible for a single sentence to be an input text. I am using two kinds of datasets in my experiments:

- Datasets downloaded from online sources or from previous papers
- Synthetically generated Datasets

They both have their uses and will be detailed in their respective sections. Moreover, I am using some of the downloaded datasets to generate synthetic ones.

3.3.1 Downloaded Datasets

I list here the different datasets I am using in this work. I only detail information that is necessary to understand the rest of my work. However, most sources contain additional details about the datasets.

- **RTBF-InfOpinions**: This dataset, created by Bogaert et al. 2023, consists of 10.000 articles in French published by the RTBF[4] between 2008 and 2021. It contains 5.000 articles classified as `Opinion` and 5.000 articles classified as `Information`. Both classes tackle similar topics. The articles have been cleaned of potential artifacts (which are defined in Subsection 2.4.1). As we will see later, the length of the input texts seems to be an important factor in the stability of explanations. I have therefore decided to truncate the different texts to 100 tokens and then to remove the last sentence in the text if the truncation split it[5]. This allows to reduce the size of of the input texts without altering them and without adding data artifacts. Indeed, this truncation should not add any bias to the task.
- **AlloCiné**: This dataset is a large-scale sentiment analysis dataset of movie reviews in French from the AlloCiné[6] website scraped by (Blard 2020). It contains two classes of reviews: `Negative` and `Positive`.
- **Wikipedia Sentences**: This dataset contains 7.8 million sentences collected and preprocessed lightly by Ortman 2018. I am using this dataset to generate some synthetic datasets.
- **Frequent English Words**: This dataset, created by Tatman 2017, contains frequent English words with their frequency of appearance in a big text corpus. I am sampling words from it when constructing some synthetic datasets.
- **Frequent English Names**: This dataset, created by Norvelle 2013, lists first and last names for different cultural groups and countries. I am using American first names when constructing some synthetic datasets. Some names

[4]RTBF stands for *Radio-télévision belge de la Communauté française* (`https://www.rtbf.be/`)
[5]An example of truncation is shown in Annex D
[6]`https://www.allocine.fr/`

that are also common nouns have been deleted from the corpus, specifically names like *January*, *May* or *April*.

3.3.2 Synthetic Dataset Generation

I detail here how the datasets I am using have been constructed but not why I need them. This is detailed in the following sections when explaining the experiments I run. The following datasets only have two classes but this could be extended without loss of generality.

3.3.2.1 English Sentences Word Giveaway Anonymized

Synthetic datasets are useful to measure the influence of specific parameters whilst keeping the others as constant as possible. The idea here is to consider that one word in the input text is enough to determine the class. I call this word the **discriminant keyword**. Using first names as discriminant keywords has multiple advantages. Firstly, they frequently appear in text, providing a large pool of texts to draw from when constructing synthetic datasets. Secondly, two different names can easily be swapped in a text while it still makes sense.

Regarding the dataset, it is comprised of texts coming from the *Wikipedia Sentences* dataset. The way it was built is simple. Firstly, I collected a subset of texts that contained names. The names came from the *Frequent English Names* dataset. After collection, I replaced each name present in these texts by a special string: `<name>`; and counted their occurrences in each of them. I now have a dataset in which I can substitute the token `<name>` with any name I choose in any given text. A typical input text in this dataset is of the form "`<name> is going to the beach tomorrow`".

I am using this dataset multiple times in this work by substituting the name and the number of names in the different input texts. The goal is however to always limit the bias in the dataset which means that if a text appears in a class, it must also appear in the other with the other keyword. Specifically, I have sampled a set of 5.000 texts for training and 50 to explain for two different lengths of texts: 10 and 50 words. These anonymized texts are then filled in to create the different tasks. I use the same anonymized texts across every dataset.

3.3.2.2 John/James Random Sentences Word Giveaway

This dataset contains two classes. Each class contains one discriminant word which determines the class in its entirety. This means that belonging to a class only depends on the presence of that word. The discriminant words I have chosen are `John` and `James` as they are first names that are interchangeable in most contexts.

This is an arbitrary choice on my part, I could have chosen any pair of first names as long as they share the same gender, the same connotation (John and James are frequent Englishman names) and that they both correspond to 1 token in the tokenizer. Each class contains 5.000 input texts with one occurrence of the name. Both classes contain the same texts with the names interchanged. This dataset is constructed from the Anonymized dataset. An example showcasing the texts' structure is shown in Table 3.1.

Anonymized	*<name>* is going to the beach tomorrow
Class 1 (John)	*John* is going to the beach tomorrow
Class 2 (James)	*James* is going to the beach tomorrow

Table 3.1: *John/James Random Sentences*' structure example

Following the principles defined beforehand, I defined two different datasets which contain texts of different lengths. One contains texts which are comprised of 10 words and the other contains texts of a length of 50 words. I name them respectively **John/James Random Sentences 10 (*JJRS10*)** and **John/James Random Sentences 50 (*JJRS50*)**[7]. All texts have been stripped of their finishing point for reasons that are explained in 3.4.1.

3.3.2.3 John/James Random Sentences Word Giveaway dots

This dataset is rigorously the same as *JJRS10* described in 3.3.2.2 except that each text still ends with a period. I name this dataset **John/James Random Sentences Dots 10 (*JJRSdots10*)**. An example of what the structure of input texts in this dataset looks like is shown in Table 3.2. The finishing point may have an impact on the attribution score distribution as explained in 3.4.1.

Anonymized	*<name>* is going to the beach tomorrow
Class 1 (John)	*John* is going to the beach tomorrow.
Class 2 (James)	*James* is going to the beach tomorrow.

Table 3.2: *John/James Random Sentences with Period*'s structure example

3.3.2.4 John/nothing Random Sentences Word Giveaway

This dataset contains two classes. The first class contains one discriminant keyword which determines the class in its entirety. The other class is characterized as not having any distinctive sign. Its particularity is that it doesn't contain any

[7] I am using these acronyms in the different experiments and plots.

discriminant keyword. The discriminant keyword for the first class is the first name `John` in order to stay consistent with the previous datasets. Each class contains 5000 input texts.

Version 1 The way the first version of this dataset is built is simple. It is based on the 5.000 random input texts from the Anonymized dataset. For the first class, I replace the `<name>` token by the keyword `John` and for the second class, I replace it by a random frequent English word from the Frequent English Words dataset. Texts from this dataset follow a structure similar to the one shown in Table 3.3.

Anonymized	*<name>* is going to the beach tomorrow
Class 1 (John)	*John* is going to the beach tomorrow
Class 2 (Random Word)	*be* is going to the beach tomorrow

Table 3.3: *John/nothing Random Sentences Version 1*'s structure example

Following the previous principles, I define 2 datasets. One containing texts comprised of 10 words and another containing texts comprised of 50 words. Respectively, I name them **John/nothing Random Sentences 10 v1 (*JNRS10v1*)** and **John/nothing Random Sentences 50 v1 (*JNRS50v1*)**. All texts have been stripped of their finishing point for reasons that are explained in 3.4.1.

Version 2 I have also defined another dataset that is very similar but, instead of replacing the `<name>` token by a random word - which is an action that breaks the context and creates a small bag-of-words in that specific spot of the text - I simply delete it. This also generates some bias as this means one class contains texts that are shorter than the other. However, I argue that for longer texts, the model will likely struggle to use that hint since there is already some non-negligible variability in the number of tokens within texts of the same length. Figure 3.5 shows the histogram of the number of tokens from both classes. I argue that the distributions are not very different and that the bias it introduces is quite minimal. I create only one dataset containing texts of 50 words that I name **John/nothing Random Sentences 50 v2 (*JNRS50v2*)**. Texts from this dataset follow a structure similar to the one shown in Table 3.4.

Anonymized	*<name>* is going to the beach tomorrow
Class 1 (John)	*John* is going to the beach tomorrow
Class 2 (nothing)	is going to the beach tomorrow

Table 3.4: *John/nothing Random Sentences Version 2*'s structure example

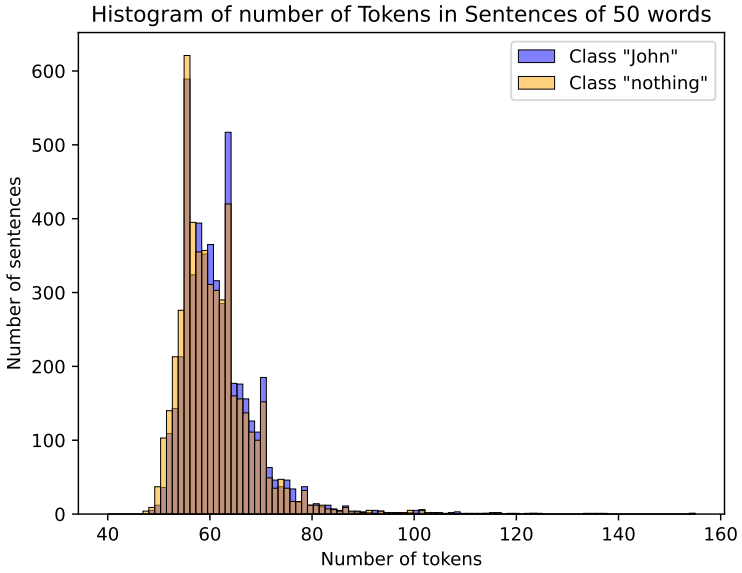

Figure 3.5: Histogram of the number of Tokens per Sentence of 50 words for "*John*" and "*nothing*"

Since both versions of the dataset introduce a form of bias, I will use both of them and observe whether one of them seems to be less biased than the other. The conclusions drawn from the experiments will always take into account the method employed to construct the datasets.

3.3.2.5 John/James Bag-of-Words Word Giveaway

The idea behind this dataset is to test whether context is necessary for a model to make its prediction and whether the explanations used to justify this prediction are stable. The goal is therefore to have one target word that defines the class in its entirety and the rest of the words do not matter at all. This dataset contains two classes. Each class contains one discriminant word which determines the class in its entirety. Belonging to a class only depends on the presence of that word. To reduce bias when comparing this task to the *JJRS10* task, the texts are the same as in *JJRS10* but shuffled. In this work, I sometimes call this ordered bag-of-words a sentence even though this is not linguistically valid. Each class contains 5.000 texts. I generate one dataset with texts of size 10 named **John/James Bag-of-Words 10 (*JJBoW10*)** and another with texts of size 50 named **John/James Bag-of-Words 50 (*JJBoW50*)**. They are respectively based on *JJRS10* and *JJRS50*. An example of such texts' structure is shown in Table 3.5

JJRS10	*John* is going to the beach tomorrow
Class 1 (John)	tomorrow going is *John* beach to the
Class 2 (James)	tomorrow going is *James* beach to the

Table 3.5: *John/James Bag-of-Words*'s structure example

3.4 Parameter Impact on Explanation Stability

The first part of this work concerns the question: **What parameters influence the stability of explanations ?** From this overarching question flow different sub-questions presented in Section 3.1. I will first dive into two more technical questions about the importance of the end period and the faithfulness of the explanation method. Afterwards, I will study how explanation stability differs when modifying different parameters using synthetic datasets. Conducting the aforementioned experiments will both raise practical questions about how to test hypotheses in NLP rigorously and spark a discussion about the importance of context in NLP sentences. It is however necessary to note that the results presented in this work are limited and their validity is very constrained. They are not general in any way, shape or form.

Every time I mention that I am using a dataset in the experiments detailed underneath, I follow the pipeline of operations described in Section 3.2 in order to end up with 100 functionally equivalent models at the end. The tasks in this section are mostly trivial which means that only 100 models need to be fine-tuned as all models will likely have perfect accuracy.

The results of the experiments showcased in this section are presented in Section 4.2.

3.4.1 Importance of the Period

I realized when generating explanations using the *John/James Random Sentences Dots* dataset that the last word in my input texts was consistently receiving a higher attribution score than it intuitively should. As mentioned in Subsubsection 2.5.2.1, each word is divided into wordPieces by the tokenizer and wordPieces belonging to the same word are aggregated together. The attribution score given to the last word contains the attributions of every wordPieces constituting and of the final period's token. Clark et al. 2019 studied this phenomenon and observed that attention heads in the deeper layers of the network seemed to attend to periods and commas. They concluded that this is due to the fact that not all attention heads in the model are relevant to every task and that when their intervention is not needed, they attend to the period at the end of the text as a sort of "*no-op*".

The goal of this question is to see whether there is a statistical difference between the attribution given to the last word when there is a period and when there is none. Whether there is a statistical difference or not, the periods of every sentence in every synthetic dataset will be removed, except when specified otherwise. All of the synthetic datasets consist of texts of only one sentence so only the finishing period would be removed. If there is a statistical difference, being able to remove this added variability is interesting to ensure that the behaviours observed when testing for a parameter are due to that parameter's impact and not to anything else. If there is no statistical difference, then I make the arbitrary decision to remove them as well to stay consistent.

To test for this difference, I use two datasets:

- JJRS10: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`. Only one keyword is necessary to determine the target class which means that the explanation provided by the different models should be completely targeted at the keyword.
- JJRSdots10: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`. Only one keyword is necessary to determine the target class which means that the explanation provided by the different models should be completely targeted at the keyword.

With each of them, I train 100 models in order to explain 100 texts. I want to see if the attribution score given to the last word of the text statistically differs from one experiment to the other. The other words are not interesting to me in this case.

I am using a t-test with unequal variances to verify that the two distributions of attributions significantly differ.

The results of this experiment are presented in Subsection 4.2.1.

3.4.2 LRP Method Sanity Check

An assumption made in Bogaert et al. 2024 is that the explanation method they are using, Layer-wise Relevance Propagation (LRP) is faithful. They expect that the explanations it produces represent faithfully what parts of the input the model used when doing its prediction. Proving that a method is faithful is impossible to do empirically since the true reasoning behind a prediction is unknown and impossible to determine, especially when models are as complex and non-linear as the ones I am currently using. There is no ground truth to compare with so it is difficult to see whether LRP is faithful or not. There exist however some scenarios where it could be possible to prove that LRP is not faithful. In this experiment, I will therefore perform a sanity check of the LRP method to see if it performs as

intended. This is the question I am asking myself here: **How faithful is the LRP method ?** We saw in Subsection 2.3.2 that finding a perfectly faithful method is essentially impossible. Indeed, it seems unreal to be able to condense the reasoning of a 100M parameter Neural Network into only a few attribution scores. I also try to answer the following subsidiary question simultaneously : "*Are explanations produced on a trivial task uniformly distributed?*". Bogaert et al. 2024 have showed that training an arbitrary large number of models on the InfOpinions dataset lead to an arbitrary large number of different explanations. The goal of this question is to show that explanations produced by LRP are not unstable in all scenarios. This would mean that the explanations produced on *InfOpinions* are diverse because the task is difficult and not because the methods are flawed.

The dataset I am using for this experiment is :

- JJRS10: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`.

The goal of this experiment is to disprove the faithfulness of LRP if the results are poor. Let's imagine that the trained models predict perfectly, which will likely be the case since the task is trivial. If, when looking at the explanations, the discriminant keyword is not highlighted, then we can assume one of two things. Either the explanation method is unfaithful as that keyword is the only way to predict correctly. Either the model guessed randomly and won the 50% chance. This second conclusion becomes exponentially less likely as the number of models trained increases so I reject it. It is imperative that the only way for a model to correctly classify is by looking at the presence of the keyword. Removing bias was therefore critical when building the dataset as any small hint count get picked on by the model and ruin the results. Moreover, to answer the subsidiary question, I only need to observe whether the mean of the explanations provided by all models differs significantly from a uniform distribution.

I expect all models to give a very high attribution score to the keywords and a very low attribution score to all of the other words. More precisely, I expect the top-1 MCWME to be 1.0 meaning that the most highlighted word needs to always be the same for every model.

The results of this experiment are presented in Subsection 4.2.2.

3.4.3 Impact of Text Length on Explanation Stability

Longer input texts contain more tokens and receive more attribution scores from the explanation method. The question I am asking here is : *Are explanations of longer input texts more unstable than shorter ones ?* The LRP method seems to

always attribute a non-zero attribution score to every word due to some sort of noise. Increasing the number of words could maybe increase this noise and the variability in the produced explanations.

The datasets I am using for this experiment are:

- JJRS10: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`. This dataset contains texts of 10 words.
- JJRS50: The task to solve when using this dataset is trivial: *predict whether the input texts contains the keyword* `John` *or* `James`. This dataset contains texts of 50 words.

Both tasks should provide very simple distributions of attribution scores: one very high attribution for the discriminant keyword and very low attribution scores for the other words. The metrics used will be the MCWME to measure overall stability and the top-MCWME to hypothesize about where it could stem from.

The results of this experiment are presented in Subsection 4.2.3.

3.4.4 Importance of Context on Explanation Stability

Context is an integral part of attention-centred neural networks such as BERT. Indeed, due to the bi-directional self-attention mechanism, every word in the input is attended to by every other word. The relationship between the different words of the input should therefore necessarily matter towards stability. The question asked in this experiment is the following: *Are explanations of coherent texts more stable than those of Bag-of-Words ?* This question may seem far-fetched but it came from the following observation. I was training models on the trivial task of predicting whether the texts contained the keyword `John` or `James`. The texts were bag-of-words. I expected the explanation to be very stable, only one big attention score given to the keyword and the other words being attributed very low scores. Instead, I saw some non-negligible attributions to tokens I deemed unimportant to the task and the whole explanation suffered from non-trivial variability. It was as if the model tried to use the broken context towards the classification.

The datasets I am using are the following:

- JJRS10: The task to solve when using this dataset is trivial: *predict whether the input contains the keyword* `John` *or* `James`. This dataset contains texts of 10 words.

- JJBoW10: The task to solve when using this dataset is trivial: *predict whether the input texts contains the keyword* **`John`** *or* **`James`**. This dataset contains texts of 10 words. The texts in this dataset are the same as in *JJRS10* except that they are shuffled. Texts in both classes of the dataset are shuffled in the same way to prevent any sort of bias.

After looking at short texts of 10 words, I do the exact same comparison with longer texts to see if the same behaviour is still happening. In that case, I am using the following datasets:

- JJRS50: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* **`John`** *or* **`James`**. This dataset contains texts of 50 words.

- JJBoW50: The task to solve when using this dataset is trivial: *predict whether the input texts contains the keyword* **`John`** *or* **`James`**. This dataset contains texts of 50 words. The texts in this dataset are the same as in *JJRS50* except that they are shuffled. Texts in both classes of the dataset are shuffled in the same way to prevent any sort of bias.

If the behaviour I observed previously still appears during this experiment, it would mean that models use the context of the surrounding words to generate the explanation for their predictions, even if those contextual words are not necessary towards the classification task per se. It would also indicate that models cannot make abstraction of the context, even when it is not necessary towards the task. In the greater scheme of things, it could mean that models need structure and context to work properly.

I need to briefly mention that the task these hypotheses are tested on is very simplistic and not representative of reality. The behaviours appearing in my results may be the product of the way the task is constructed and not inherent to NLP tasks as a whole. I however argue that using NLP datasets is a necessity based on how difficult it is to isolate parameters in real-world tasks. This experiment could maybe be extended to real-world datasets in the future.

The metrics used for this question are the MCWME and the top-*1* MCWME. Even if some variability appears in the Bag-of-Words task, I still expect the most attributed-to word to be the discriminant keyword for all models.

The results of this experiment are presented in Subsection 4.2.4.

3.4.5 No Discriminant Words

Since the LRP method's goal is to pinpoint which words in the input contribute to the classification, we can assume that some sort of discriminant word or some

combination of them needs to be apparent for the method to work correctly. However, in real-world datasets, it is possible for a class to be defined as not containing any distinctive signs. For example, in the sentiment analysis task of defining if a sentence is neutral or not. The neutral class is defined not by the presence of specific linguistic features, but by the absence of strong sentiment (Ribeiro et al. 2016a). Typically, in movie reviews whose scoring ranges from 0 to 10, reviews that are attributed the grade of 5 are considered neutral. In this scenario, I hypothesize that the LRP method is ineffective and that the provided distribution should resemble a uniform distribution where every token receives a relatively similar random attribution. I hypothesize that this task is effectively out of the scope of usage of the explanation method as it cannot not select anything. We can imagine that the method is still faithful and may also be plausible sometimes but its output is useless. The question here is therefore: *Are explanations stable when there is no discriminant word in an input text ?*

I am dividing this experiment in two parts using two different pairs of datasets since both provide interesting results and spark interesting discussions.

Version 1 The first pair of datasets is the following:

- JJRS10: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`. This dataset contains texts of 10 words.
- JNRS10v1: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or not.* This dataset contains texts of 10 words. In the second class of texts, the keyword `John` has been replaced by a random frequent English word.

I expect the explanations to be very stable for the first class as only one keyword needs to be highlighted. The top-1 MCWME will likely be perfect as well. For the second class, I expect the explanations to be very unstable overall and to resemble the uniform distribution with every token being given a similar attribution score.

I conduct the same comparison but with input texts containing 50 words to see if the same conclusions can be drawn for longer texts using the following datasets:

- JJRS50: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`.
- JNRS50v1: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or not.* In the second class of texts, the keyword `John` has been replaced by a random frequent English word.

Since the conclusions I draw from this experiment are in the same vein as the ones presented for shorter texts, I put the related plots and analyses in Appendix A.3.2.

Version 2 The second pair of datasets is the following:

- JJRS50: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`. This dataset contains texts of 50 words.
- JJRS50: The task to solve when using this dataset is trivial: *predict whether the input text contains the keyword* `John` *or* `James`. This dataset contains texts of 50 words in its first class but of 49 words in its second as the keyword `John` has been simply erased in the second class. I justify this decision and the minimal bias it introduces in Subsubsection 3.3.2.4.

The goal of the task overall is to see what parts of the input the model uses when there is nothing to base its decision on. Removing everything that can be used as a hint towards classification is therefore essential in order to obtain a dataset as unbiased as possible. Both tasks will likely still contain some bias but that will be developed when presenting the results in Subsection 4.2.5.

3.5 Task Difficulty & Explanation Stability

The second part of this work raises the question: **Does the variability to randomness depend on the difficulty of the task ?** From this overarching question flow different sub-questions presented in Section 3.1. I will first look at differences in stability of different classes within two real-world tasks and then compare these two tasks together, measure if one of them is harder than the other based on model accuracy and then observe if the stability of explanations is correlated with their respective performance. Both tasks are in French so I select the pre-trained CamemBERT model as my model to fine-tune.

Following, I make the hypothesis that a class containing strong markers will likely correlate with high explanation stability. Indeed, since the LRP method targets the words of most impact towards the classification, I suppose that polarizing words or groups of words that aid immensely towards the prediction will be highlighted by most models. We will see if this hypothesis holds true for the coming experiments and will link why it may or may not with the results of the synthetic experiments.

The results of the experiments showcased in this part are presented in Section 4.3.

3.5.1 InfOpinions : Information vs. Opinion

Explanations of texts from *InfOpinions* are very unstable, as surveyed by Bogaert et al. 2024. One of the two classes contains journalistic `opinion` pieces which tend to contain stronger word choices and more expressive punctuation than the other class, journalistic `informations`, which could be defined as lacking any `opinion` markers. In this experiment, I look at both classes of the dataset to see if there is a difference in stability. Intuitively, I expect the `opinion` class to be more stable than the `information` class as it would likely contain more expressive words easier to pinpoint by the LRP method.

I train 200 models with 1 epoch on the truncated InfOpinions dataset. Since the task is non-trivial, models start to make mistakes when predicting, meaning that some models are discarded to only keep functionally equivalent ones. The goal is to keep 100 models to estimate sufficiently well my confidence intervals.

The results of this experiment are presented in Subsection 4.3.1.

3.5.2 AlloCiné : Positive vs. Negative

The AlloCiné dataset contains polarized positive and negative movie reviews of 50 words. I expect explanations from both classes of text to have a similar level of stability. Indeed, both classes should have strong markers that the models should rely on when classifying. My hypothesis is that the models will manage to effectively find polarized markers in texts and be stable.

I train 200 models with 1 epoch for this task. Since the task is difficult, models will make mistakes and not obtain perfect accuracy on the test set which means that the set of functionally equivalent models is smaller than 200. I only keep 100 models.

The results of this experiment are presented in Subsection 4.3.2.

3.5.3 InfOpinions vs. AlloCiné

The goal of the experiment here is to look at whether one of the tasks is more unstable than the other. The question I am asking here is : *Does the stability of explanations depend on the difficulty of the task ?* As will be formalized later, on synthetic examples, it seems like they are decorrelated as it was possible to imagine a task on which models performed perfectly but that had very unstable explanations. This experiment looks at whether this kind of behaviours appears as well for real-world datasets.

In classification, some tasks are more complicated than others. The first part of this experiment is to measure whether one of the two tasks is harder than the

other using the testing accuracy. Next, I look at the stability of both tasks and compare them. Using the two previously-generated results, I look into whether there is a connection between model performance and explanation stability.

This experiment uses the following datasets:

- Truncated InfOpinions: The goal is to predict whether the text is a journalistic `information` or `opinion`. This dataset contains sentences of 50 words.
- AlloCiné: The goal is to predict whether the movie review is `positive` or `negative`. This dataset contains sentences of 50 words.

To answer the aforementioned question, I am training 200 models and keeping 100 functionally equivalent ones and use the MCWME to measure the variability in explanations.

The results of this experiment are presented in Subsection 4.3.3.

Part 4

Results & Analysis

In this chapter, I present and analyse the results of the experiments designed and explained in the Methodology chapter. The scope of validity of certain analyses is quite limited as the datasets used are synthetic and the tasks specific but they may hold some relevance outside of that scope. I will first touch upon the topic of synthetic dataset generation.

4.1 Difficulty of generating Synthetic Datasets

So, the goal of synthetic data is to control every outside parameter to be able to measure a single isolated parameter. The question is now whether it is possible or easy to do such a thing. I will walk you through different methods that I tried and explain why there is still some bias.

Let's take an hypothetical task of predicting whether texts in a class contain a specific word or not. This example does not represent real-world use cases and certainly doesn't require a 100M parameter model to solve. However, the goal is to see how the LRP method works so I argue it is useful. Each class contains a certain specific keyword: $[k1, k2]$. My first instinct was to sample texts containing each word from a big dataset and create my two classes. The problem with this method, in this particular context, is that other words in the text may appear more often in a class than in another. Let's say $k1$ is the word `positive` and $k2$ is the word `negative`, then there will likely be a sadder lexical field in the second one. This difference in contextual words can be used for the prediction and adds bias to the search.

Another idea I had was to swap using the positive/negative keywords but to use first names as they should appear in similar contexts. I chose the first names `John` and `James` and sampled texts containing them from a big dataset. It was first of all difficult to get enough texts of exactly the same length as datasets are

finite. I need 5.000 texts containing 10 words with one of them being the first name. Secondly, since the texts in which the first names appear differ from one dataset to the next. It is inevitable that some words appear more frequently in one than the other, introducing some bias.

My final idea was to sample 5000 anonymized texts of length 10 and to replace the anonymous token by one or the other name. This means that the only thing that differs between the two datasets is the keyword. Everything else is exactly the same: the number of times a word appears, the order of words in texts, the index of the word in texts... This is the version of the dataset that I deem to have the least amount of bias for this trivial task. Though this is the least I can achieve, there is still some bias because of the way models are pre-trained. Some words tend to appear more alongside other words in the pre-training step making some connections more likely than others. There will always be some bias when testing hypotheses.

4.2 Parameter Impact on Explanation Stability

The results and analyses detailed here correspond to the experiments presented and explained in Section 3.4.

The models trained on the input texts of this part all have essentially perfect testing accuracy ([0.99, 1.0]) and all agree on their predictions of the texts to explain. Therefore, all 100 models that are trained are functionally equivalent.

4.2.1 Importance of the Period

Figure 4.1 shows that there is a qualitative difference in the mean attribution given to the last word between the two datasets. Indeed, it seems like when a period is added at the end of the sentence, the model automatically gives it a higher attribution score than when it is not present. This behaviour is strange as the last word is not necessary for prediction at all, only the discriminant keyword is.

Figure 4.2 shows an histogram of the mean attribution given to the last word of each text in both datasets. We can visually see that both distributions are very different and that the dataset with no period receives more frequently lower attribution scores. To make sure that the distributions are different, I conduct a Welch's t-test (Welch 1947), which does not assume equal population variance. The null hypothesis H_0 states that the mean of the two distributions is the same. The p-value is $1.09 * 10^{-47}$ which allows to reject the null hypothesis and conclude that the two distributions are different.

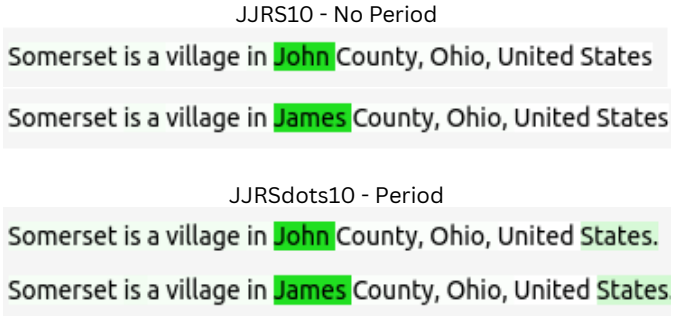

Figure 4.1: Attention Maps of mean Explanations of *JJRS10* and *JJRSdots10*

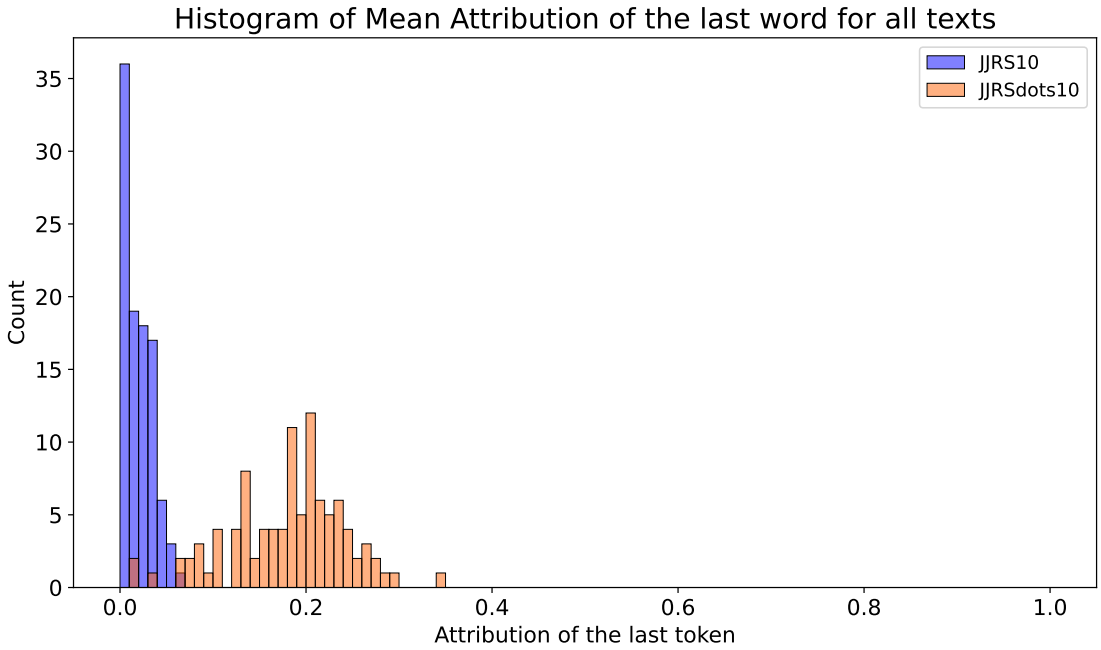

Figure 4.2: Histogram of the Mean Attribution Scores for *JJRS10* and *JJRSdots10*

Since there seems to be a statistical difference in the attribution given to the last token. I have removed all the periods from every other dataset. This removes one undesirable variability-inducing behaviour, which allows to better characterize the importance of the different parameters on explanation stability. More generally, this experiment shows that unexpected behaviours can appear when working in NLP.

4.2.2 LRP Method Sanity Check

Figure 4.3 shows in blue the MCWME with its confidence interval and in pink the top-*1* MCWME with its confidence interval for 100 texts from the *JJRS10* dataset. The MCWME is close to 1.0 for every single text meaning most explanations generated by the 100 models are very similar. A very high MCWME is a sign of stability. Moreover, I can see that the confidence intervals are very low which is another sign of stability. Something remarkable is the fact that the top-*1* MCWME, which is the MCWME computed by taking only the single highest attribution score per text, is equal to 1.0 for every single text. This is only possible if all models generate an explanation are equal to the mean explanation. This means that the discriminant word is selected as most important in 100% of texts.

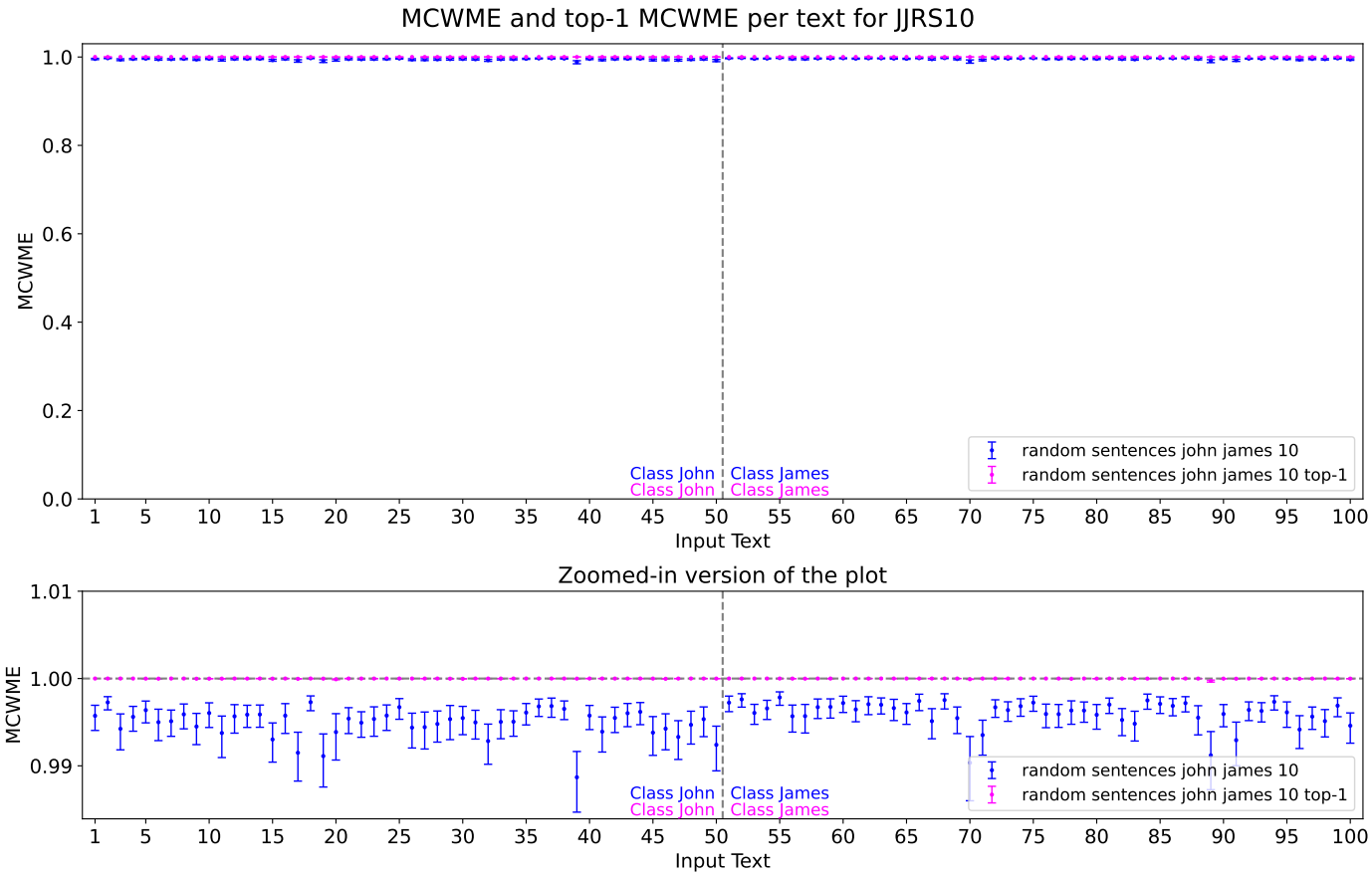

Figure 4.3: MCWME and top-*1* MCWME per Text for *JJRS10*

Interestingly, while the MCWME is very high, it has not reached 1.0 for any input text. This is likely due to some noisy connections made between the different

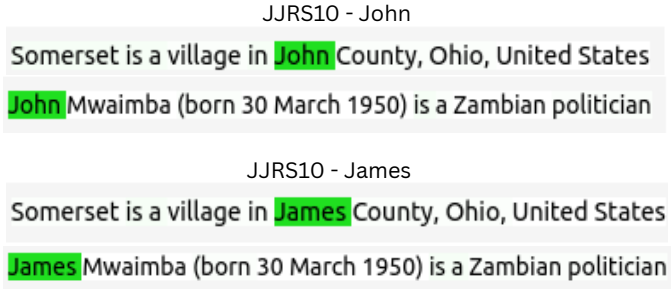

Figure 4.4: Attention map of Mean Explanations for *JJRS10* texts

words. It seems like every word in each explanation is attributed a random close-to-zero attribution score regardless of its necessity towards the task. This is certainly due to how the self-attention mechanism works in transformer models, constructing links between every word in the input. I therefore think that LRP making these connections visible is a sign of its faithfulness.

Qualitatively, we can observe from the mean explanations of texts randomly sampled from the dataset shown in Figure 4.4 that the models considered the discriminant keyword as most important and did not attribute anything substantial to the other words. This is intuitively what should happen as the keyword present in each input text is the only reason for prediction. Due to this stability, I conclude that the LRP method seems to not produce trivially unfaithful explanations and I make the assumption that it is the case for every other experiment coming after this one. Indeed, the explanations provided do not seem unfaithful but it is not possible to definitely prove that the method actually is faithful. Moreover, looking at the MCWME across all texts, I conclude that the explanations provided on this task are stable meaning that the results in Bogaert et al. 2024 are most likely due to the difficulty of the task and not the ineffectiveness of the explanation method.

4.2.3 Impact of Text Length on Explanation Stability

Figure 4.5 shows the error bar graph with the MCWME of the *JJRS10* (in blue) and of the *JJRS50* datasets (in orange). It is important to note that the pairs of texts have been randomly constituted. There is no one-on-one correspondence between texts from both tasks. There seems to be a difference in stability between shorter and longer texts but quantifying it is difficult. The shorter texts have overall a higher MCWME and smaller confidence intervals while the longer texts perform consistently lower with a wider confidence interval. Even though there are no one-on-one comparisons, the overlaps indicate that longer texts are usually significantly different than shorter one. Indeed, only 12 out of 100 pairs of texts have an overlap. Based on these 3 facts and the general trend of the graph, I

conclude that the input text length is indeed impacting the explanation stability.

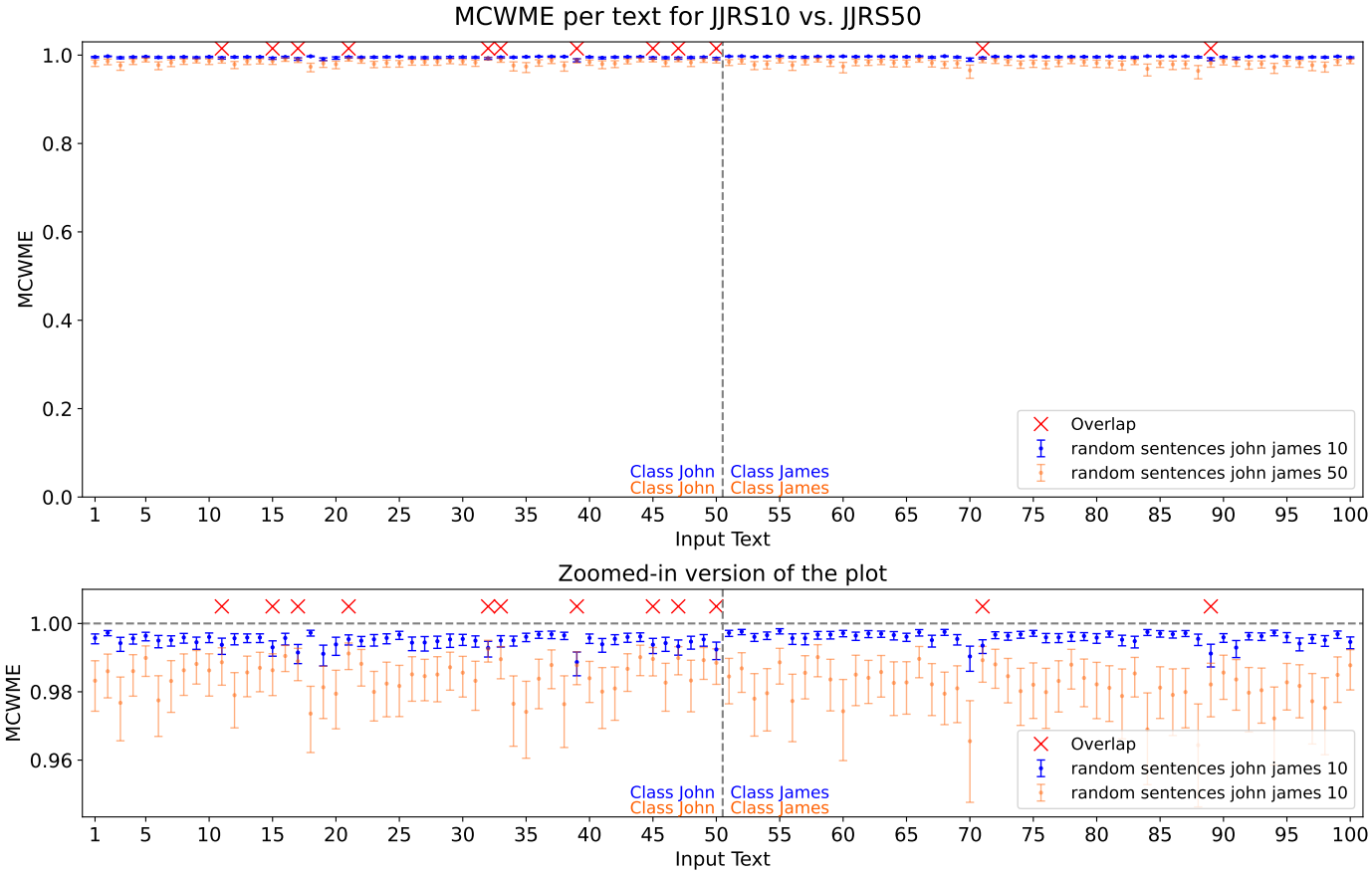

Figure 4.5: MCWME per Text for *JJRS10* vs. *JJRS50*

The following question arises: *Where does this variability come from ?* Knowing that that variability exists is important but locating its origin is even more important. Figure 4.6 shows the scatter plot of correlations with the Mean explanation for different values of the top-k CWME. This graph shows the results on one text chosen randomly. I make the arbitrary assumption that it is representative of the other texts. We can see that the top-*1* correlations equal 1.0. This means that one word is highlighted as most important all the time, the discriminant keyword. For higher values of k, more variability is observed. There are some outliers that diverge from the mean explanation but there is also just an overall variability that looks almost intrinsic to the task. Adding more words in each explanation seems to, at first, decrease its stability, but it goes back up with a sufficient number of words. This is due to the fact that most explanations have intrinsic variability in

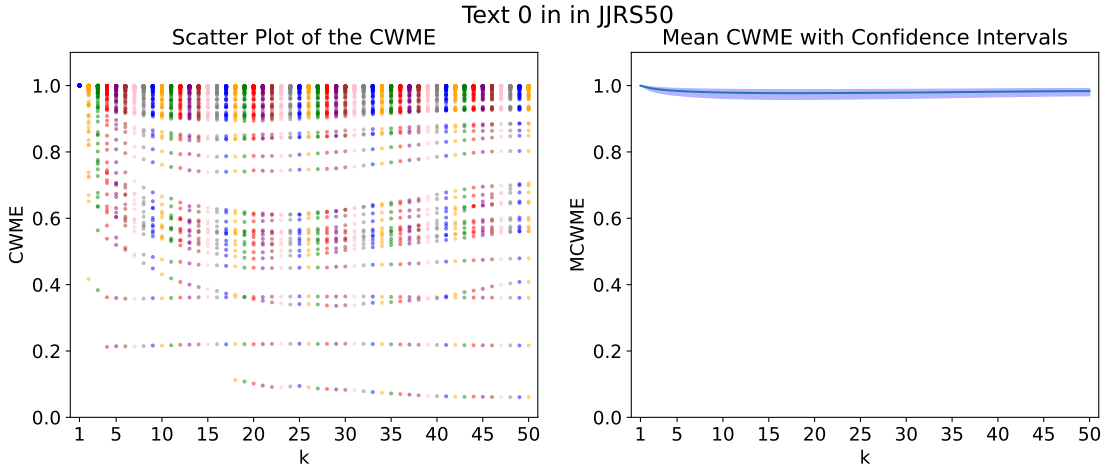

Figure 4.6: Scatter Plot of CWME for different values of k for text 0 in *JJRS50*

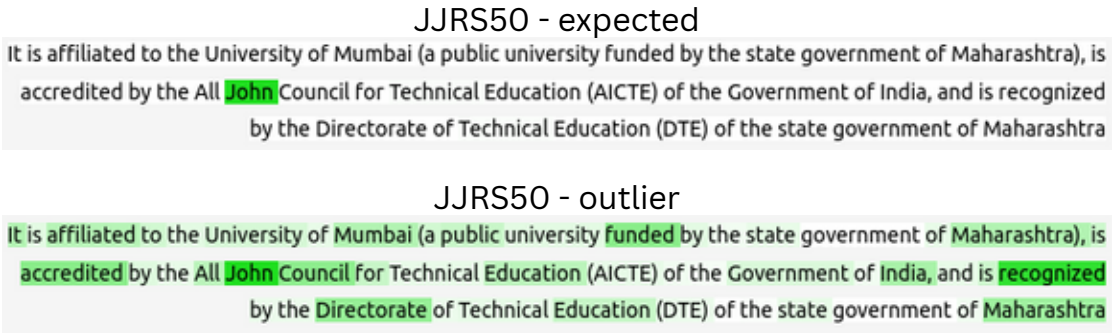

Figure 4.7: Attention Maps of Explanations for text 0 in *JJRS50*

the attribution scores they produce and they do not select all the same tokens in the same order of importance. Qualitatively, the outlying explanations are very different from expected explanations as shown in Figure 4.7.

The variability does not come from the top attribution. It must therefore come from the other words, which are all given random, close-to-zero, attribution scores. It means that increasing the length of a text will automatically degrade its performance. The top-k MCWME seemed like a pragmatic solution to limit that noise but, in practical cases, since models usually do not select the same tokens, it leads to worse performance than when taking all words into account. I will look at this problematic in Part 2 of this work.

This experiment shows that noisy attributions seem to play a big factor in the stability of explanations. Since longer texts contain more attribution scores, they therefore struggle more.

4.2.4 Importance of Context on Explanation Stability

This experiment tackles the question: "**Does context matter when classifying, even when there is no need for this context ?**". The task at hand is trivial and requires no context, just identifying the target word. Since *JJBoW10* contains the same input texts as *JJRS10* but shuffled, there is a 1-on-1 correspondence between texts which is respected in the following graphs.

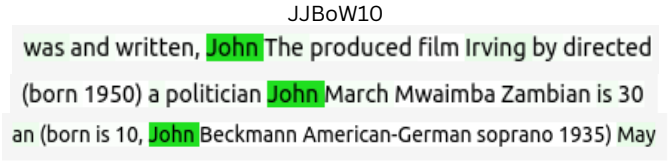

Figure 4.8: Attention Maps of Mean Explanation for texts in *JJBoW10*

Short Texts This first part looks at the results of short texts of 10 words. If we make the assumption that the LRP method is faithful, then the connections displayed in the attention maps shown in Figure 4.8 are the true reasoning for classification. These attention maps show that the highest scoring word is the discriminant keyword but other words seem to receive non-negligible attributions. I can then observe from Figure 4.9 that models produce less stable explanations for Bag-of-Words texts since the MCWME is lower and its confidence interval is wider. With coherent texts, the explanation is always targeted almost solely on the discriminant keyword with a noisy attribution to the other tokens as mentioned beforehand. I think that the attributions given to contextual words in the *JJBoW10* task are not entirely due to noise. My guess is that the pre-trained model learns meaningful information about how to structure a text and, when presented with something incoherent, struggles to make sense of what is happening even when it is not necessary for the task. It therefore tries to create meaningful connections between incoherent sequences words and hallucinates some.

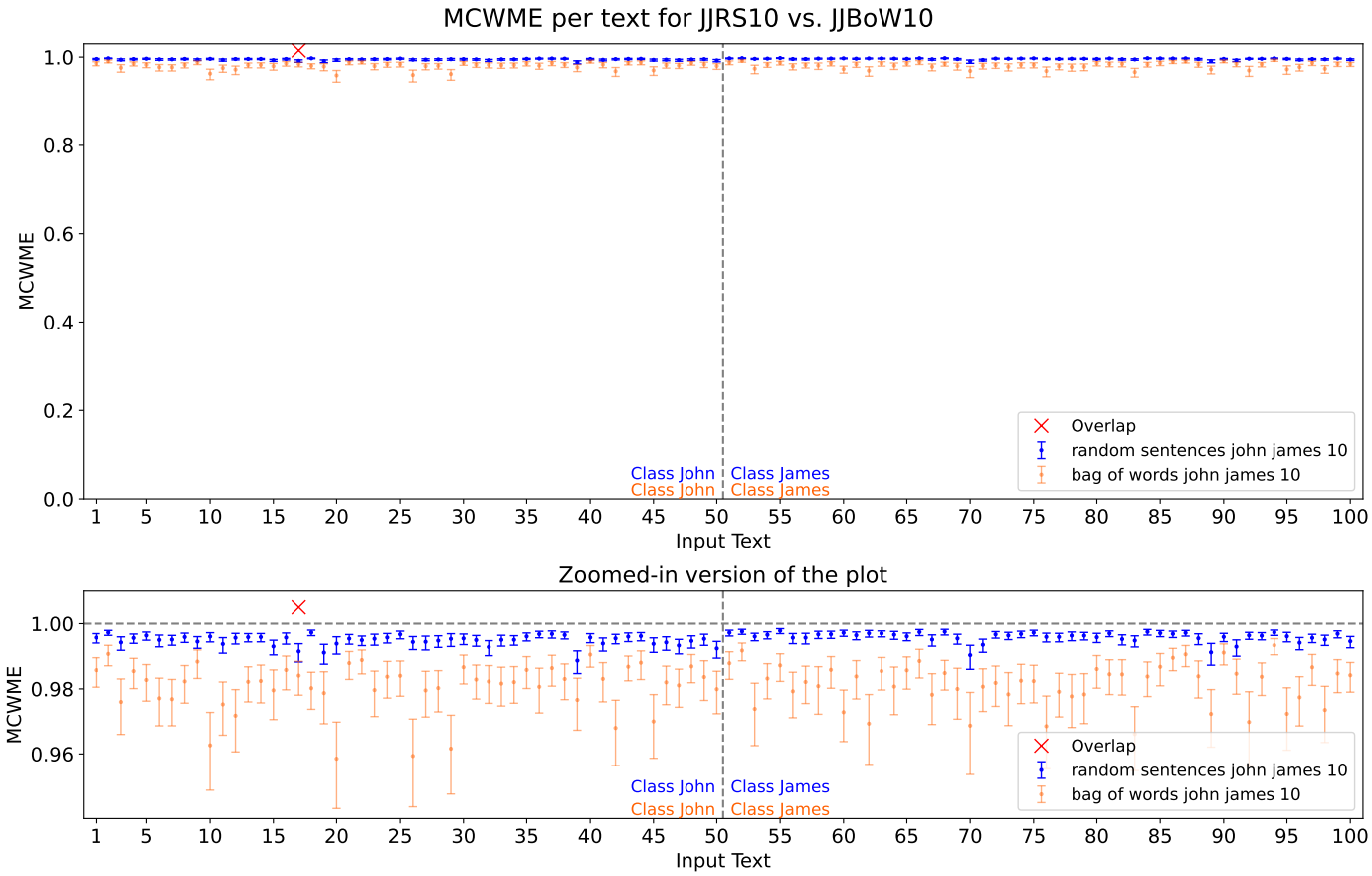

Figure 4.9: MCWME per text for *JJRS10* vs. *JJBoW10*

Based on Figure 4.9, I observe 3 things, *JJRS10* outperforms *JJBoW10* across all texts, has lower confidence intervals and both experiments almost never overlap. This allows me to conclude that there is a difference in stability between these two tasks. Therefore, context and the coherence of the text is important towards the classification. Otherwise, the model makes connections appear out of thin air.

Long Sentences Figure 4.10 shows the MCWME with its confidence interval for 100 texts for *JJRS50* and *JJBoW50*. Looking at the overlaps, we can quickly conclude that both distributions seem to be similar. The results presented here combine two different parameters, text length and presence of context.

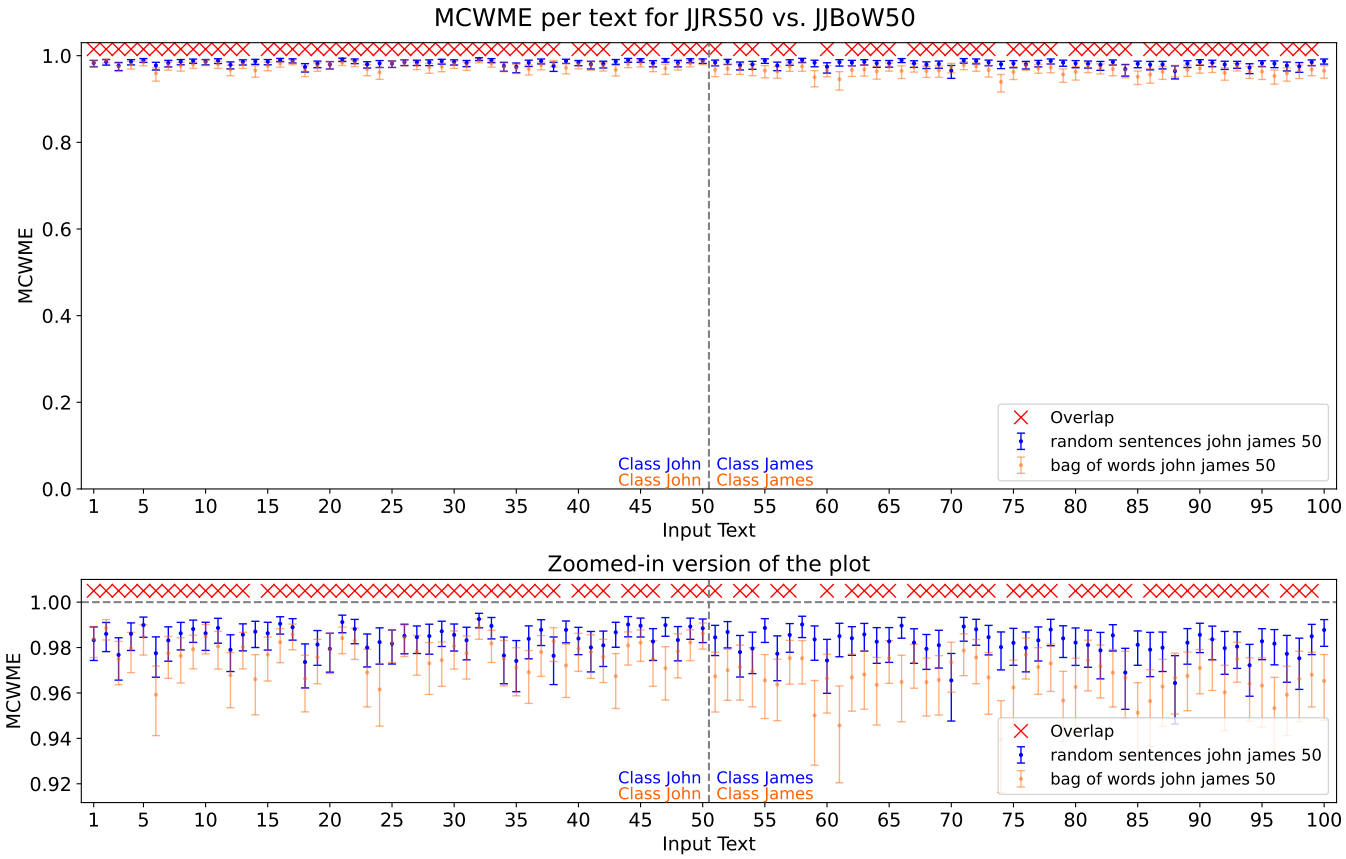

Figure 4.10: MCWME per Sentence for *JJRS50* vs. *JJBoW50*

Based on the comparison shown in Figure 4.11 between shuffled texts of different lengths, we can see that the difference is negligible implying that when the text lacks context, its length doesn't matter as much which I suggest is due to the attributions already being noisy.

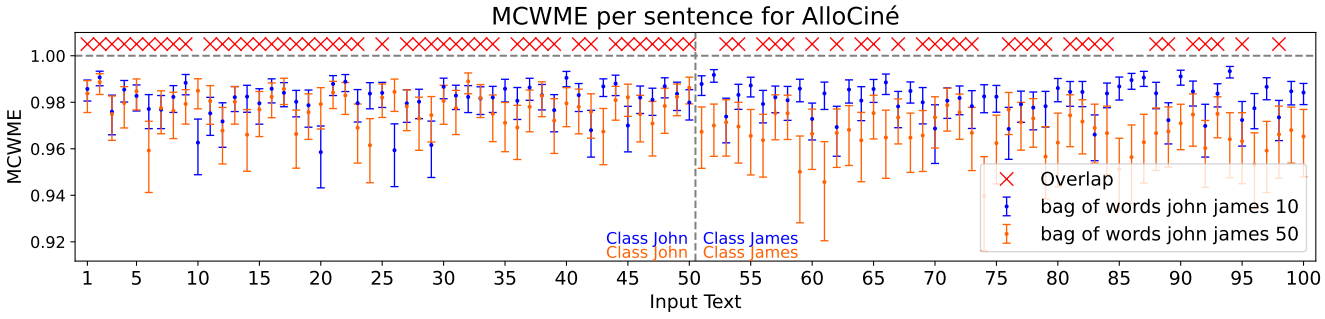

Figure 4.11: MCWME per Sentence for *JJBoW10* vs. *JJBoW50*

Context seems to play an integral role in the stability of explanations and a lack of it decreases the quality of explanations. This effect is less visible for longer texts as coherent long texts already suffer from noisy attributions. The need for coherence likely comes from the pre-training where models learn what the proper structure of a sentence is.

4.2.5 No Discriminant Words

Figure 4.12 shows the MCWME per text for the *JJRS10* and the *JNRS10v1* datasets. As expected, the explanations of the second class where the texts do not contain any keyword are pretty unstable and differ greatly from one to the next.

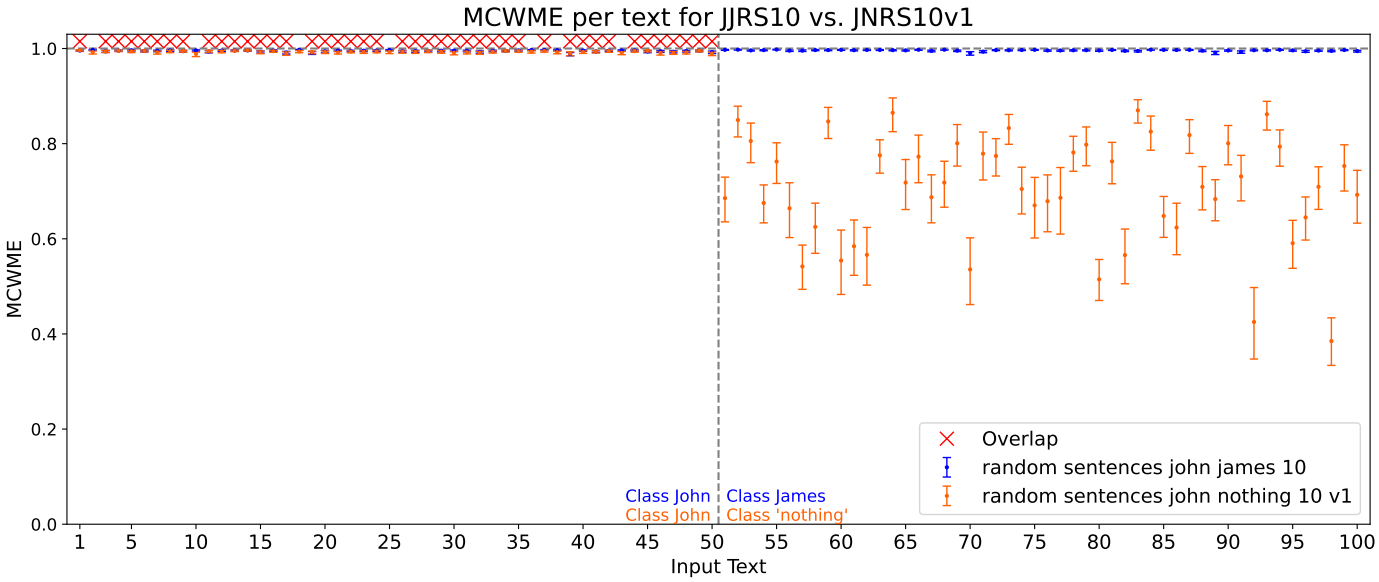

Figure 4.12: MCWME per text for *JJRS10* vs. *JNRS10v1*

As a reminder, the first version of the *JNRS10* dataset replaced the discriminant keyword by a random frequent English word. What is interesting is that, when looking at attention maps of the mean explanations shown in Figure 4.13, we can see in the `nothing` texts that the word that is given the most importance is the replaced word. This implies that the model, during its training, picked up on the fact that a word was replaced by another and tried to find it and, through the broken context, managed to do just that on unseen texts. This raises three interesting insights. Firstly, it means that models are extremely good at finding gaps in context and seem to grasp a fundamental understanding of language. Secondly, it means that the results provided in Figure 4.12 are biased as the distribution was

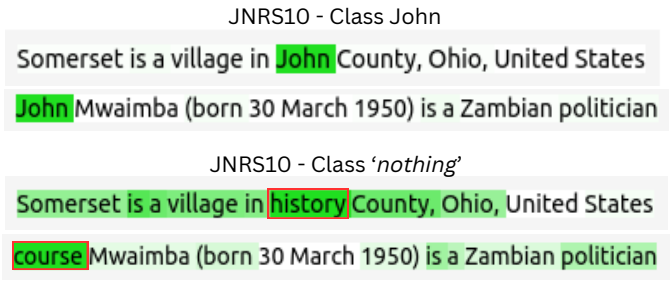

Figure 4.13: Attention Map of *JNRS10v1* texts

in fact provided a hint, the broken context, and does not represent entirely what is studied. Thirdly, it shines a light on the meaningful difficulty that is testing hypotheses in NLP. Testing hypotheses is really complicated as modifying datasets sometimes introduces unexpected bias.

The analyses related to the *JJRS50* vs. *JNRS50v1* task are presented in Annex A.3.2. I present here the key insights this analysis provides. Increasing the length of texts did not change the fact that the model located where the context was broken because the highest attributed-to word is still the replaced word. This means that the bias hasn't been diluted and is still present in my results.

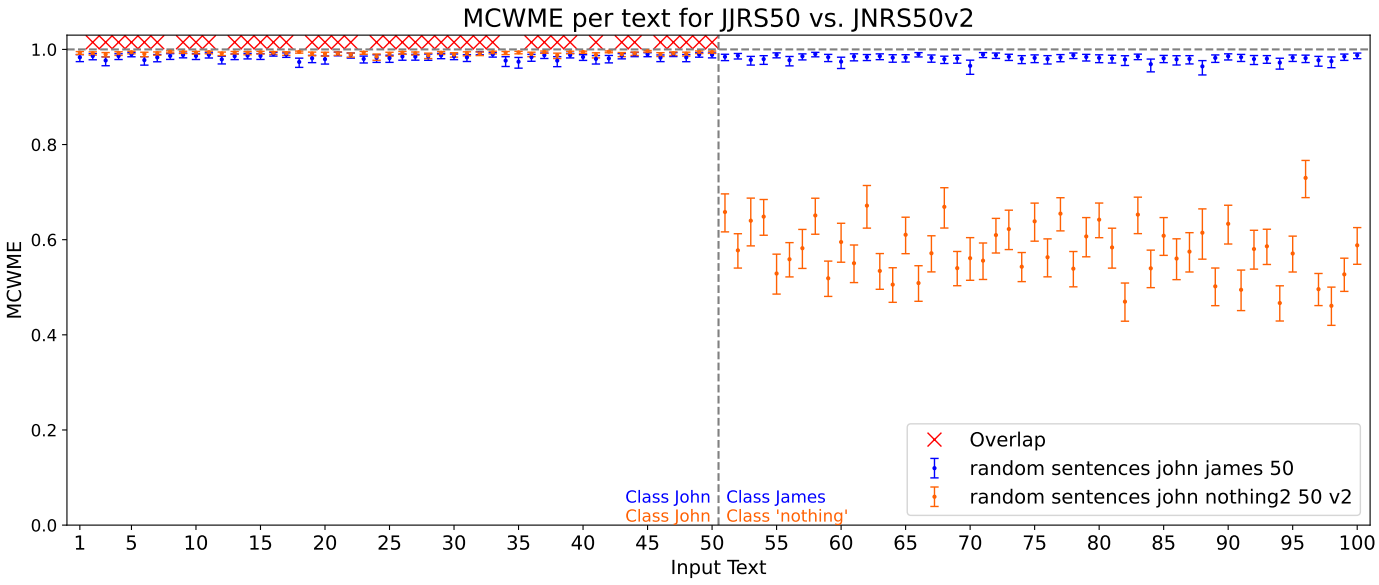

Figure 4.14: MCWME per Text for *JJRS50* vs. *JNRS50v2*

Figure 4.14 shows the results of the *JJRS50* and *JNRS50v2* datasets. Let's first discuss the potential residual bias and then consider what the distribution

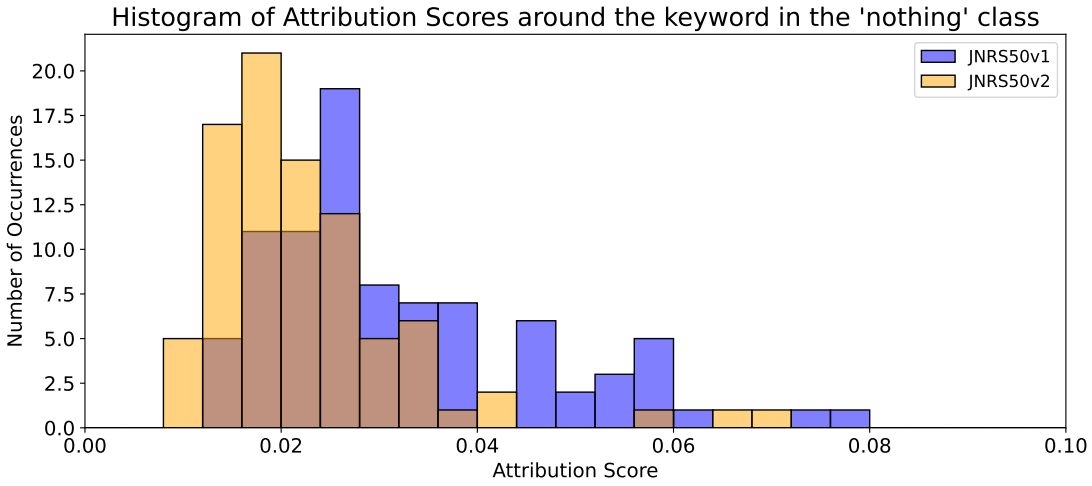

Figure 4.15: Distribution of attribution scores given to words surrounding where the keyword used to be; Surrounding window=1

looks like. Removing the discriminant word altogether introduces another bias linked to the length of the text. This bias is unquantifiable by the LRP method as it does not correspond to a specific keyword in the text but I consider it to be negligible (see Subsubsection 3.3.2.4). Moreover, removing the keyword does not remove the bias linked to the lack of context, the explanation method simply cannot highlight that word as it does not appear in its input. Figure 4.15 shows the attribution scores of the words surrounding the removed keyword in the mean explanations of *JNRS50v2* (in orange). I compare these attributions with the ones surrounding the random frequent word in *JNRS50v1* (in blue). The surrounding window is the number of words selected on both sides of the keyword. We can see that the distributions are similar and, if anything, the attributions given to *JNRS50v2* seem lower. This indicates that the model does not seem to either detect the context break or either cannot represent it in the attribution scores of neighbouring tokens. Qualitatively, Figure 4.16 shows attention maps of mean explanations. The attribution scores are not cluttered around where the context was broken which comforts my idea that the model cannot represent it. I therefore suppose that the bias introduced due to the context break is minimal.

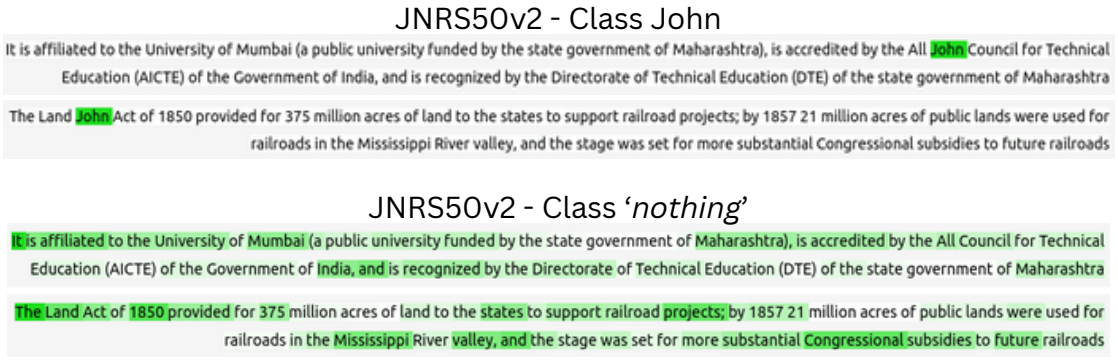

Figure 4.16: Attention Maps of Mean Explanations of *JNRS50v2* texts

Regarding the distribution, I conclude that the results are as expected. The explanations provided by LRP are very unstable. However, it is noteworthy that, while they vary a lot, they only reach values of 0.45 at worse and are far from the worse case scenario of 0.0. This means that there exists some sort of structure in those explanations that is shared between every text. Additionally, we can note that LRP is not suited for explaining texts that do not contain discriminant words. It is not that the method in and of itself is unfaithful, it is just that the kind of explanations provided lacks plausibility. Moreover, when the classification task is difficult, the inner mechanisms are more complex and may require more than just one attention score per word to be plausible. In the scenario of this experiment, the problem with LRP explanations is therefore not faithfulness but plausibility as explanations do not seem to make any sense at all and lack intuitive meaning. Even humans would probably struggle to find plausible explanations in those texts as there are no definite markers. The distribution observed is not completely random though as it averages 0.6 meaning that the models still agreed somewhat on their explanations.

I therefore conclude that, in this synthetic experiment, when a text lacks a discriminant keyword, it provides unstable explanations, regardless of the length of the text.

4.3 Task Difficulty & Explanation Stability

The following results are heavily dataset-dependent and are not general in any way. They correspond to the experiments explained and designed in Section 3.5.

4.3.1 InfOpinions : Information vs. Opinion

Models trained for this experiment average 95% test accuracy. I keep 100 functionally equivalent models on 60 texts (30 from both classes). Figure 4.17 shows the comparison of the MCWME for randomized pairs of texts between the two classes in *InfOpinions*. The `information` class is in blue and the `opinion` class is in orange. Contrary to my hypothesis, the `information` texts are more stable than `opinions`. They are consistently above the opinions even if there are some overlaps. This contradicts the belief that strong markers lead to more stable LRP explanations. I expected `opinions` to be more stable as texts tend to contain more expressive vocabulary or more extravagant punctuation, letting the model easily highlight them.

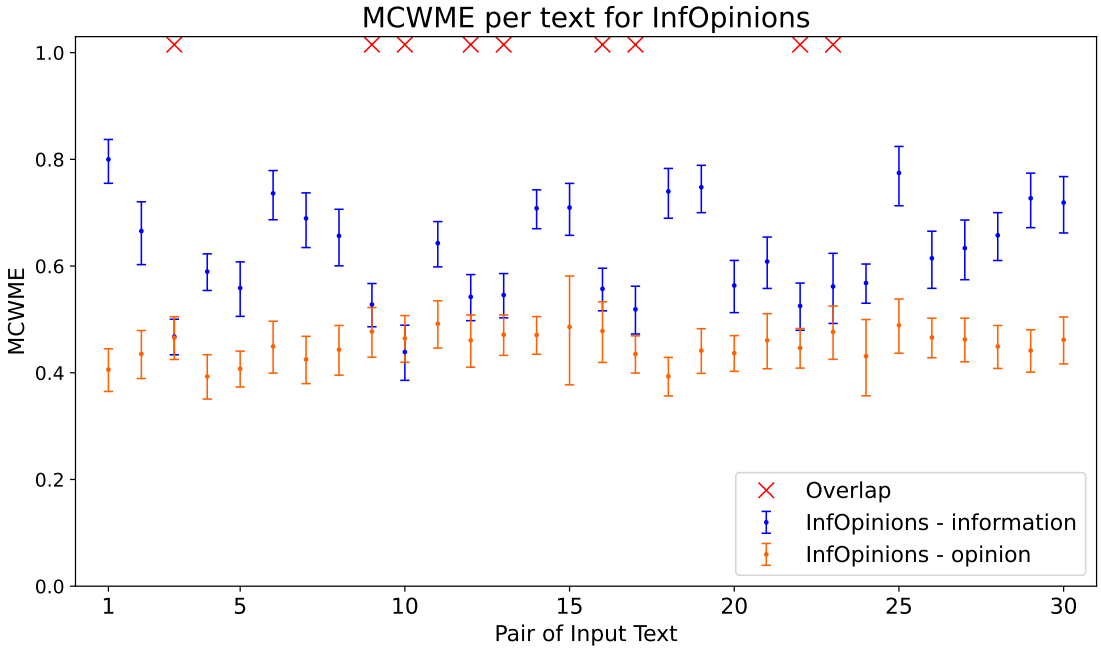

Figure 4.17: MCWME per pair of text for *InfOpinions*

I hypothesize that the `information` class is more stable for two reasons, illustrated in Figure 4.18. Firstly, `information` texts could be following a more clear and standardized structure which repeats itself from one text to the next. Secondly, `opinions` are more expressive but differ greatly from one text to the next maybe not allowing the model to find trends easily. The problem would therefore be that the dataset is not big enough, not allowing the model to generalize sufficiently well. Moreover, `opinions` usually contain multiple polarizing words per text which may increase the variability of explanations for one simple reason, there may be

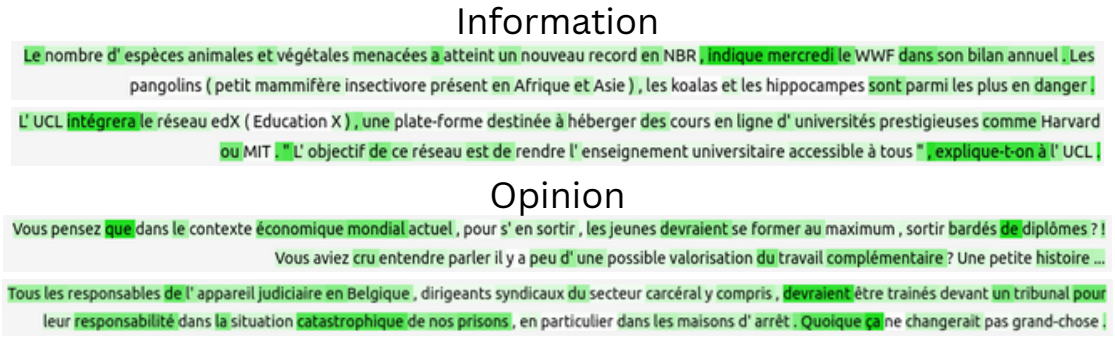

Figure 4.18: Attention Maps of Mean explanations of *InfOpinions* texts

multiple ways to explain a decision that are all faithful and plausible. This raises the question of whether there exist some clusters and of how to find them if they exist.

4.3.2 AlloCiné : Positive vs. Negative

Figure 4.19 shows the MCWME per text for the *AlloCiné* dataset. Since I compare both classes of the dataset, I constituted random pairs of texts. The `negative` class is in blue and the `positive` class is in orange. Models trained on this task obtained around 85% accuracy. Results use 100 functionally equivalent models. As we can see, there seems to be no clear difference in stability between the two classes. Indeed, they oscillate around the same MCWME value, have confidence intervals with similar widths and almost all pairs of texts overlap. From the perspective of markers in text, this result makes sense as both classes contain the same kind of polarized vocabulary.

However, in the greater scheme of things, a MCWME that varies around 0.5 does not characterize extremely stable explanations. As a point of comparison, it is around the same stability as the class that contained no discriminant word in Subsection 4.2.5. There is some structure to them but they vary non-negligibly from one to the next. This instability may stem from multiple sources. My first hypothesis is that since there are a lot of different markers in the text, multiple explanations exist that may all be plausible. This raises the question of clusters of explanations again.

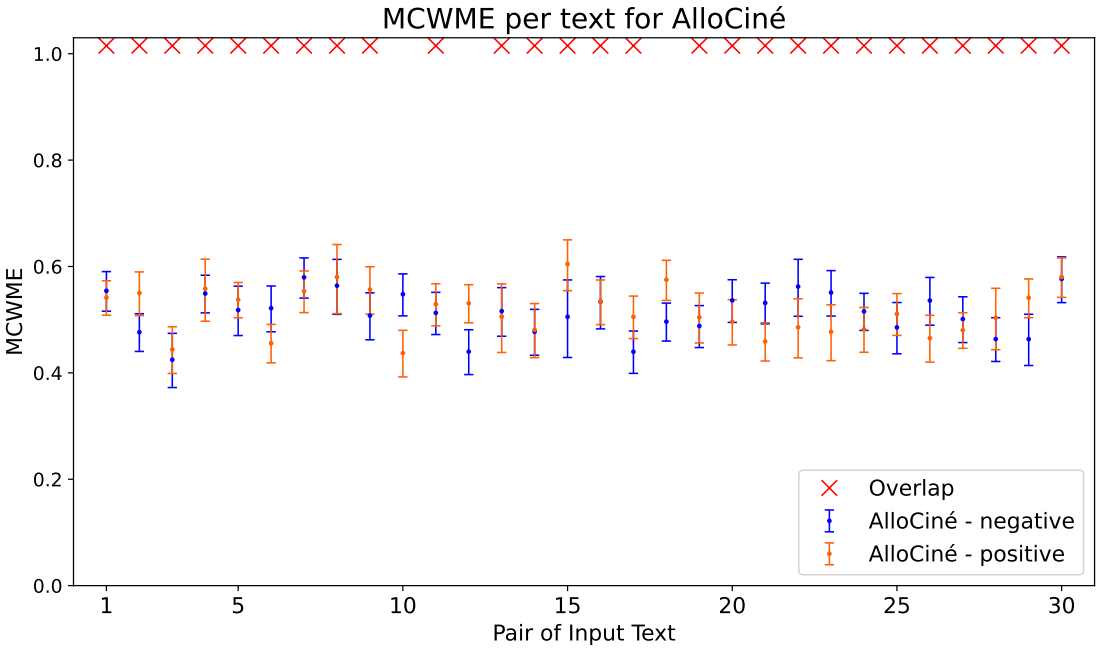

Figure 4.19: MCWME per Text for *AlloCiné*

4.3.3 InfOpinions vs. AlloCiné

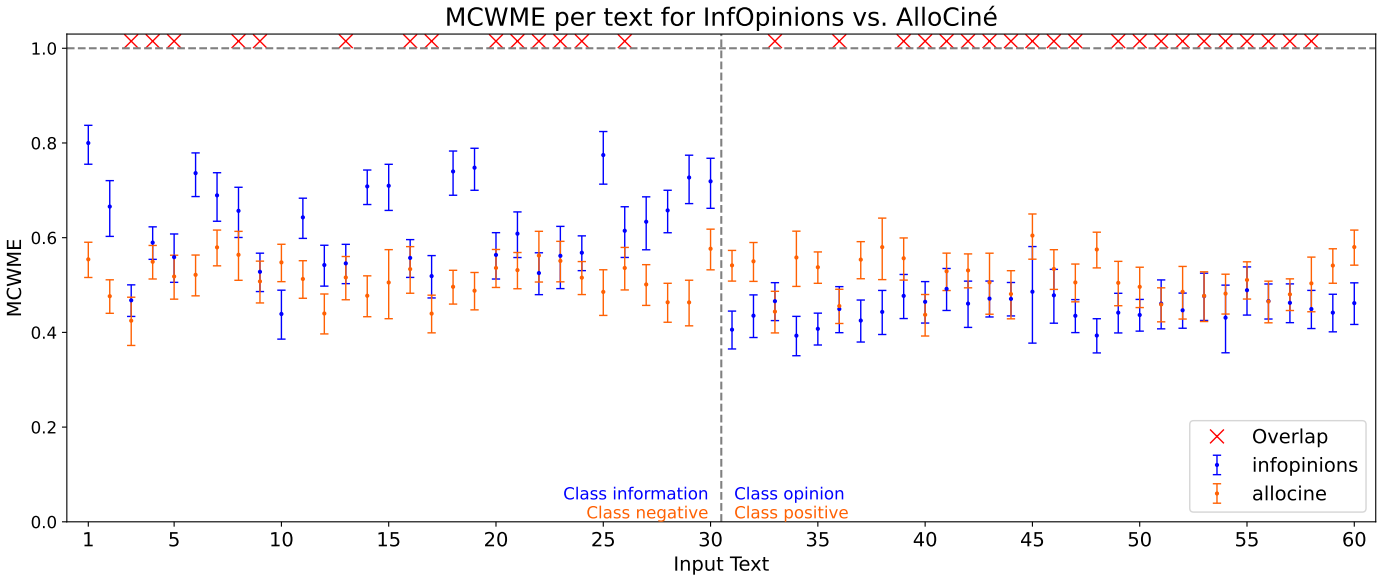

Figure 4.20: MCWME per Text for *InfOpinions* vs. *AlloCiné*

Figure 4.20 shows the MCWME for texts in *AlloCiné* (in orange) and *InfOpinions* (in blue). The classes compared are `information` vs. `negative` and `opinion` vs. `positive`. The pairs of classes have been determined randomly. Moreover, since *AlloCiné* texts share the same stability across both classes, it does not have any impact on my analyses. Based on their performance on the test set, the *AlloCiné* task is harder than the *InfOpinions* task as the accuracy obtained is significantly lower as shown in Table 4.1. Overall, both tasks seem to exhibit the same level of stability as a lot of pairs of confidence intervals seem to overlap. This could suggest that the difficulty of the task is not correlated with the stability of explanations in all cases, as was previously shown for synthetic datasets.

Dataset	**Mean Accuracy**	**Confidence Interval**
InfOpinions	0.909	[0.907, 0.911]
AlloCiné	0.875	[0.873, 0.878]

Table 4.1: Testing Accuracy

Additionally, it looks like the `information` class is a little bit above the three others (`positive`, `negative`, `opinion`). The common denominator of these three classes is the fact that they contain strong markers and that they are very diverse. My hypothesis as to why the explanations are less stable is that the training set is not big enough and the model has not seen enough texts to generalize well. I could imagine that the space in which the `information` sentences lie is smaller than the space of the other classes, giving it a more general structure that the models can easily learn, increasing its stability. Based on this hypothesis, more sentences are necessary to explore that space. An interesting research direction could be to increase the size of the training set confirm this hypothesis or not. Apart from that, as shown in Figure 4.18, it seems like the information class also has some strong markers that define it and that the models seems to pinpoint efficiently. The stability is maybe a parameter of the number of markers. Studying its effect looks like an interesting research direction.

Part 5

Discussion & Conclusion

In this work, I first verified that the Layer-wise Relevance Propagation (LRP) method wasn't trivially unfaithful, obtaining essentially perfect stability on a synthetic task. I showed that even for that trivial task, with a very simple objective, containing short sentences and no punctuation, the model did not achieve perfect stability, raising questions about the usefulness of this method in more complicated contexts. Next, I investigated the effect of different factors on the variability of explanations using synthetic datasets. The first parameter I looked at was the length of the sentence. I showed that it has a significant impact on the stability due to every word being attributed a random close-to-zero noisy attribution score, reducing stability. The second parameter I looked at is the coherence of the sentence. Context as a whole seems to be really important for stability with the model relying on it for its prediction and even finding where the context was broken when it was. A lack of context increases instability but a class not containing any markers increases it even more. When a sentence does not contain any keyword, the explanations provided are very diverse but still share a non-negligible structure. Based on the fact that two similarly-performing models have very different MCWME, I also showed that the variability of explanations is not correlated to the difficulty of the task in the case of synthetic datasets.

Using two real-world datasets, I then looked into the variability of explanations with respect to the difficulty of the task as well as with the presence of strong markers. Based on my results, which are very dataset-dependent, I conclude that even if two tasks perform with similar performance, the stability of their explanations can differ, confirming this result from the synthetic part. Moreover, within *InfOpinions*, there seems to be a significant difference in stability between the classes. I hypothesize that the instability of the `opinions` comes from the fact that there are too many strong markers.

As suggested during the work, further research could tackle the topics of looking

at how explanations are distributed when multiple discriminant keywords are present in a sentence, to try to mimic real-world datasets. It could also go more in-depth as to why the `information` class is more stable than the `opinion` class and link that to the presence of strong markers and of structure. The question of whether a gender bias present in the pre-training influences the stability of explanation is interesting and deserves a careful experimental setup. Unrelated to that, adapting the NLP experiments conducted in this work to images could be interesting.

Overall, my results are not very optimistic. How could the explanation method justify effectively a judicial decision of hundreds of pages when it struggles to explain stably a 50 word text with no punctuation ? This all loops back to the question of whether it will ever be possible to explain the decisions taken by large models using so simple representations. And is it even a simple representation ? A distribution of attribution scores is plausible when the sentence contains 10 words but I wouldn't bet on it making much sense when there are tens of thousands, even if the behaviour shown is perfectly faithful. I think that the most promising direction in explainable AI is to ask the model to provide a plausible explanation of its decision and to observe within its weights if it is faithful or not in the style of mechanistic interpretability[1]. This is very difficult, requires cutting-edge models and a lot of compute and data which begs the question of the responsibility of private companies in this problem. It also raises the question of whether it would be ethical to use such systems even if the explanations provided are unstable. It is a desiderata that comforts us in the idea that an explanation is faithful but it may not even be a necessary prerequisite for faithfulness. Until a stable method for explaining complex decisions is discovered, I personally believe that large language models can still be used, even in sensitive domains, given their significant potential benefits. However, their outputs must always be approached with great caution. They should be regarded as tools to support decision-making, not as definitive solutions.

[1] Bereska et al. 2024

Part 6

Use of AI Tools

I used GitHub Copilot while coding for auto-completes and for help with the syntax when using libraries I hadn't used before and ChatGPT for some light reformulation suggestions during the writing process.

Bibliography

3Blue1Brown (2024). *Attention in transformers, step-by-step | DL6*. YouTube. URL: https://www.youtube.com/watch?v=eMlx5fFNoYc&t=269s.

Agarwal, Chirag et al. (2024). *Faithfulness vs. Plausibility: On the (Un)Reliability of Explanations from Large Language Models*. arXiv: 2402.04614 [cs.CL]. URL: https://arxiv.org/abs/2402.04614.

Bach, Sebastian et al. (July 2015). "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLOS ONE* 10.7, pp. 1–46. DOI: 10.1371/journal.pone.0130140. URL: https://doi.org/10.1371/journal.pone.0130140.

Bereska, Leonard et al. (2024). "Mechanistic Interpretability for AI Safety - A Review". In: *CoRR* abs/2404.14082. DOI: 10.48550/ARXIV.2404.14082. arXiv: 2404.14082. URL: https://doi.org/10.48550/arXiv.2404.14082.

Bhardwaj, Rishabh et al. (2021). "Investigating Gender Bias in BERT". In: *Cogn. Comput.* 13.4, pp. 1008–1018. DOI: 10.1007/S12559-021-09881-2. URL: https://doi.org/10.1007/s12559-021-09881-2.

Binder, Alexander et al. (2016). "Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers". In: *Artificial Neural Networks and Machine Learning - ICANN 2016 - 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II*. Ed. by Alessandro E. P. Villa et al. Vol. 9887. Lecture Notes in Computer Science. Springer, pp. 63–71. DOI: 10.1007/978-3-319-44781-0_8. URL: https://doi.org/10.1007/978-3-319-44781-0%5C_8.

Blard, Théophile (2020). *French sentiment analysis with BERT*. URL: https://github.com/TheophileBlard/french-sentiment-analysis-with-bert.

Bogaert, Jérémie et al. (2023). *TIPECS : A corpus cleaning method using machine learning and qualitative analysis*. URL: https://dial.uclouvain.be/pr/boreal/object/boreal:276581.

Bogaert, Jérémie et al. (2024). "Explanation sensitivity to the randomness of large language models: the case of journalistic text classification". In: *CoRR* abs/2410.05085. DOI: 10.48550/ARXIV.2410.05085. arXiv: 2410.05085. URL: https://doi.org/10.48550/arXiv.2410.05085.

Bogaert, Jérémie et al. (July 2025). "Consolidating Explanation Stability Metrics". In: *XAI 2025*. Springer.

Brown, Tom B. et al. (2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

captum, Facebook Open Source - (2025). *Captum - Model Interpretability for PyTorch - captum.ai*. https://captum.ai/. [Accessed 24-05-2025].

Chefer, Hila et al. (2021). "Transformer Interpretability Beyond Attention Visualization". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, pp. 782–791. DOI: 10.1109/CVPR46437.2021.00084. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Chefer%5C_Transformer%5C_Interpretability%5C_Beyond%5C_Attention%5C_Visualization%5CCVPR%5C_2021%5C_paper.html.

Clark, Kevin et al. (Aug. 2019). "What Does BERT Look at? An Analysis of BERT's Attention". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Ed. by Tal Linzen et al. Florence, Italy: Association for Computational Linguistics, pp. 276–286. DOI: 10.18653/v1/W19-4828. URL: https://aclanthology.org/W19-4828/.

Cristian, Leo (2024). *The Math Behind Stochastic Gradient Descent*. en. Accessed: 2025-5-20. URL: %5Curl%7Bhttps://towardsdatascience.com/stochastic-gradient-descent-math-and-python-code-35b5e66d6f79/%7D.

Devlin, Jacob et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

Doshi-Velez, Finale et al. (2017). *Towards A Rigorous Science of Interpretable Machine Learning*. arXiv: 1702.08608 [stat.ML]. URL: https://arxiv.org/abs/1702.08608.

European Parliament et al. (May 4, 2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council*. of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). OJ L 119, 4.5.2016, p. 1–88. URL: https://data.europa.eu/eli/reg/2016/679/oj (visited on 04/13/2023).

Gallegos, Isabel O. et al. (2024). "Bias and Fairness in Large Language Models: A Survey". In: *Comput. Linguistics* 50.3, pp. 1097–1179. DOI: 10.1162/COLI_A_00524. URL: https://doi.org/10.1162/coli%5C_a%5C_00524.

Gardner, Matt et al. (2021). "Competency Problems: On Finding and Removing Artifacts in Language Data". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Ed. by Marie-Francine

Moens et al. Association for Computational Linguistics, pp. 1801–1813. DOI: 10.18653/V1/2021.EMNLP-MAIN.135. URL: https://doi.org/10.18653/v1/2021.emnlp-main.135.

Geva, Mor et al. (Nov. 2019). "Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, pp. 1161–1166. DOI: 10.18653/v1/D19-1107. URL: https://aclanthology.org/D19-1107/.

Gururangan, Suchin et al. (June 2018). "Annotation Artifacts in Natural Language Inference Data". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Ed. by Marilyn Walker et al. New Orleans, Louisiana: Association for Computational Linguistics, pp. 107–112. DOI: 10.18653/v1/N18-2017. URL: https://aclanthology.org/N18-2017/.

Huda, Mahmood (2024). *6 Types of Useful Transformer Models and their Use Cases — datasciencedojo.com*. https://datasciencedojo.com/blog/transformer-models-types-their-uses/. [Accessed 23-04-2025].

Jacovi, Alon et al. (July 2020). "Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness?" In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, pp. 4198–4205. DOI: 10.18653/v1/2020.acl-main.386. URL: https://aclanthology.org/2020.acl-main.386/.

Joshi, Aditya et al. (Sept. 2017). "Automatic Sarcasm Detection: A Survey". In: *ACM Comput. Surv.* 50.5. ISSN: 0360-0300. DOI: 10.1145/3124420. URL: https://doi.org/10.1145/3124420.

Kästner, Christian (2021). *Interpretability and Explainability*. Carnegie Mellon University. URL: https://ckaestne.medium.com/interpretability-and-explainability-a80131467856.

Kotek, Hadas et al. (2023). "Gender bias and stereotypes in Large Language Models". In: *Proceedings of The ACM Collective Intelligence Conference, CI 2023, Delft, Netherlands, November 6-9, 2023*. Ed. by Michael S. Bernstein et al. ACM, pp. 12–24. DOI: 10.1145/3582269.3615599. URL: https://doi.org/10.1145/3582269.3615599.

Lehmann, E. L. et al. (2008). *Testing Statistical Hypotheses*. 3rd. Springer texts in statistics. Springer.

Liu, Yinhan et al. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692.

Lyu, Qing et al. (2024). *Towards Faithful Model Explanation in NLP: A Survey*. arXiv: 2209.11326 [cs.CL]. URL: https://arxiv.org/abs/2209.11326.

Martin, Louis et al. (2020). "CamemBERT: a Tasty French Language Model". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. DOI: 10.18653/v1/2020.acl-main.645. URL: http://dx.doi.org/10.18653/v1/2020.acl-main.645.

McCoy, R. Thomas et al. (July 2019). "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen et al. Florence, Italy: Association for Computational Linguistics, pp. 3428–3448. DOI: 10.18653/v1/P19-1334. URL: https://aclanthology.org/P19-1334/.

Mienye, Ibomoiye Domor et al. (2024). "A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications". In: *Information* 15.12. ISSN: 2078-2489. DOI: 10.3390/info15120755. URL: https://www.mdpi.com/2078-2489/15/12/755.

Norvelle, Erik (2013). *NameDatabases*. URL: https://github.com/smashew/NameDatabases/tree/master.

Ortiz Suárez, Pedro Javier et al. (July 2019). "Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures". In: *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Ed. by Piotr Bański et al. Cardiff, United Kingdom: Leibniz-Institut für Deutsche Sprache. DOI: 10.14618/IDS-PUB-9021. URL: https://inria.hal.science/hal-02148693.

Ortman, Mike (2018). *Wikipedia Sentences*. URL: https://www.kaggle.com/datasets/mikeortman/wikipedia-sentences/data.

Parthasarathy, Venkatesh Balavadhani et al. (2024). "The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities". In: *CoRR* abs/2408.13296. DOI: 10.48550/ARXIV.2408.13296. arXiv: 2408.13296. URL: https://doi.org/10.48550/arXiv.2408.13296.

Ribeiro, Filipe N. et al. (2016a). "SentiBench - a benchmark comparison of state-of-the-practice sentiment analysis methods". In: *EPJ Data Sci.* 5.1, p. 23. DOI: 10.1140/EPJDS/S13688-016-0085-1. URL: https://doi.org/10.1140/epjds/s13688-016-0085-1.

Ribeiro, Marco Túlio et al. (2016b). “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Balaji Krishnapuram et al. ACM, pp. 1135–1144. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

Sarikaya, Ferhat (Nov. 2024). *Batch size selection in deep learning: A comprehensive analysis of training dynamics and performance optimization*. en. Accessed: 2025-5-20. URL: %5Curl%7Bhttps://medium.com/@ferhatsarikaya/batch-size-selection-in-deep-learning-a-comprehensive-analysis-of-training-dynamics-and-5b50f3dfbaf9%7D.

Selvaraju, Ramprasaath R. et al. (2017). “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, pp. 618–626. DOI: 10.1109/ICCV.2017.74. URL: https://doi.org/10.1109/ICCV.2017.74.

Silver, N Clayton et al. (Feb. 1987). “Averaging correlation coefficients: Should Fisher’s z transformation be used?” en. In: *J. Appl. Psychol.* 72.1, pp. 146–148.

Smilkov, Daniel et al. (2017). “SmoothGrad: removing noise by adding noise”. In: *CoRR* abs/1706.03825. arXiv: 1706.03825. URL: http://arxiv.org/abs/1706.03825.

Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *J. Mach. Learn. Res.* 15.1, pp. 1929–1958. DOI: 10.5555/2627435.2670313. URL: https://dl.acm.org/doi/10.5555/2627435.2670313.

Sun, Tony et al. (2019). “Mitigating Gender Bias in Natural Language Processing: Literature Review”. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen et al. Association for Computational Linguistics, pp. 1630–1640. DOI: 10.18653/V1/P19-1159. URL: https://doi.org/10.18653/v1/p19-1159.

Szegedy, Christian et al. (2014). *Intriguing properties of neural networks*. arXiv: 1312.6199 [cs.CV]. URL: https://arxiv.org/abs/1312.6199.

Tatman, Rachael (2017). *English Word Frequency*. URL: https://www.kaggle.com/datasets/rtatman/english-word-frequency.

Vaswani, Ashish et al. (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al., pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Voita, Elena et al. (2019). "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Ed. by Anna Korhonen et al. Association for Computational Linguistics, pp. 5797–5808. DOI: 10.18653/V1/P19-1580. URL: https://doi.org/10.18653/v1/p19-1580.

Weidinger, Laura et al. (2021). "Ethical and social risks of harm from Language Models". In: *CoRR* abs/2112.04359. arXiv: 2112.04359. URL: https://arxiv.org/abs/2112.04359.

Welch, B. L. (Jan. 1947). "THE GENERALIZATION OF 'STUDENT'S' PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARLANCES ARE INVOLVED". In: *Biometrika* 34.1-2, pp. 28–35. ISSN: 0006-3444. DOI: 10.1093/biomet/34.1-2.28. eprint: https://academic.oup.com/biomet/article-pdf/34/1-2/28/553093/34-1-2-28.pdf. URL: https://doi.org/10.1093/biomet/34.1-2.28.

Wiegreffe, Sarah et al. (2019). "Attention is not not Explanation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, pp. 11–20. DOI: 10.18653/V1/D19-1002. URL: https://doi.org/10.18653/v1/D19-1002.

Wu, Zhengxuan et al. (2021). "On Explaining Your Explanations of BERT: An Empirical Study with Sequence Classification". In: *CoRR* abs/2101.00196. arXiv: 2101.00196. URL: https://arxiv.org/abs/2101.00196.

Xinying, Song et al. (2021). "Fast WordPiece Tokenization". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Ed. by Moens Marie-Francine et al. Association for Computational Linguistics, pp. 2089–2103. DOI: 10.18653/V1/2021.EMNLP-MAIN.160. URL: https://doi.org/10.18653/v1/2021.emnlp-main.160.

Zhao, Haiyan et al. (2023). "Explainability for Large Language Models: A Survey". In: *CoRR* abs/2309.01029. DOI: 10.48550/ARXIV.2309.01029. arXiv: 2309.01029. URL: https://doi.org/10.48550/arXiv.2309.01029.

Appendix A

Other Experiments I conducted

A.1 Datasets

A.1.0.1 John/Jane Random Sentences Word Giveaway

This dataset contains two classes. Each class contains one discriminant word which determines the class in its entirety. This means that belonging to a class only depends on the presence of that word. The discriminant words I have chosen are the first names `John` and `Jane`. Each class contains 5.000 sentences with one occurrence of the name per sentence. Both classes contain the same sentences with the names interchanged. This dataset is constructed from the Anonymized dataset. I sample sentences which only contain one `<name>` token. The sentences in this dataset have a length of 10 words to be able to efficiently compare with the JJRS10 dataset. I name this dataset **John/Jane Random Sentences 10 (*John/Jane RS10*)**. An example of what the structure of sentences in this dataset looks like is shown in Table A.1. All sentences have been stripped of their finishing period for reasons that are explained in 3.4.1.

Anonymized	*<name>* is going to the beach tomorrow
Class 1 (John)	*John* is going to the beach tomorrow
Class 2 (Jane)	*Jane* is going to the beach tomorrow

Table A.1: *John/Jane Random Sentences*' structure example

A.2 Methodology

A.2.1 Gender Bias in Pre-training

The overall topic of gender bias is very complex and needs way more than one little subsection in this Master's thesis to be fully explored. I just ask myself a small side question: *Does explanation stability depend on the gender of the name ?*. The goal is just to observe whether there is a statistical difference in the stability of explanations and, if so, to hypothesize why it might or might not be the case. It has been shown in multiple studies that LLMs are prone to reproducing all kinds of biases present in their training data (Gallegos et al. 2024). One of those biases is gender bias which manifests itself when a model performs worse in sentences where the subject is a woman than when it is a man (Sun et al. 2019, Bhardwaj et al. 2021). Gender bias in LLMs can cause damage. For example, "*adults who are exposed to stereotypes may adopt them or have ones they already believe reinforced, causing them to engage in (conscious or unconscious) discrimination*" says Kotek et al. 2023. Gender bias is especially problematic in sensitive domains like automatic insurance or job CV selection.

To answer this question, I am using two datasets:

- JJRS10: The task to solve when using this dataset is trivial: *predict whether the sentence contains the keyword* `John` *or* `James`. This dataset contains sentences of 10 words. This is the baseline where both discriminant keywords are masculine names.
- John/Jane RS10: The task to solve when using this dataset is trivial: *predict whether the sentence contains the keyword* `John` *or* `Jane`. This dataset contains sentences of 10 words. This dataset contains one masculine and one feminine name.

The analyses in this experiment are two-fold. Firstly, I will look at whether the explanation stability from the two experiments differ significantly. Secondly, I will concentrate on the *John/Jane RS10* task and see whether the explanations differ from one class to the other.

The results of this experiment are presented in Subsection A.3.1.

A.3 Results

A.3.1 Gender Bias in Pre-training

Figure A.1 shows the MCWME for 100 sentences from the JJRS10 and the John/-Jane RS10 datasets. Each experiment explains 50 sentences from one class and

50 from the other class. They are separated by the middle gray dashed line. The mean explanation for the first sentence from each dataset is shown in Figure A.2.

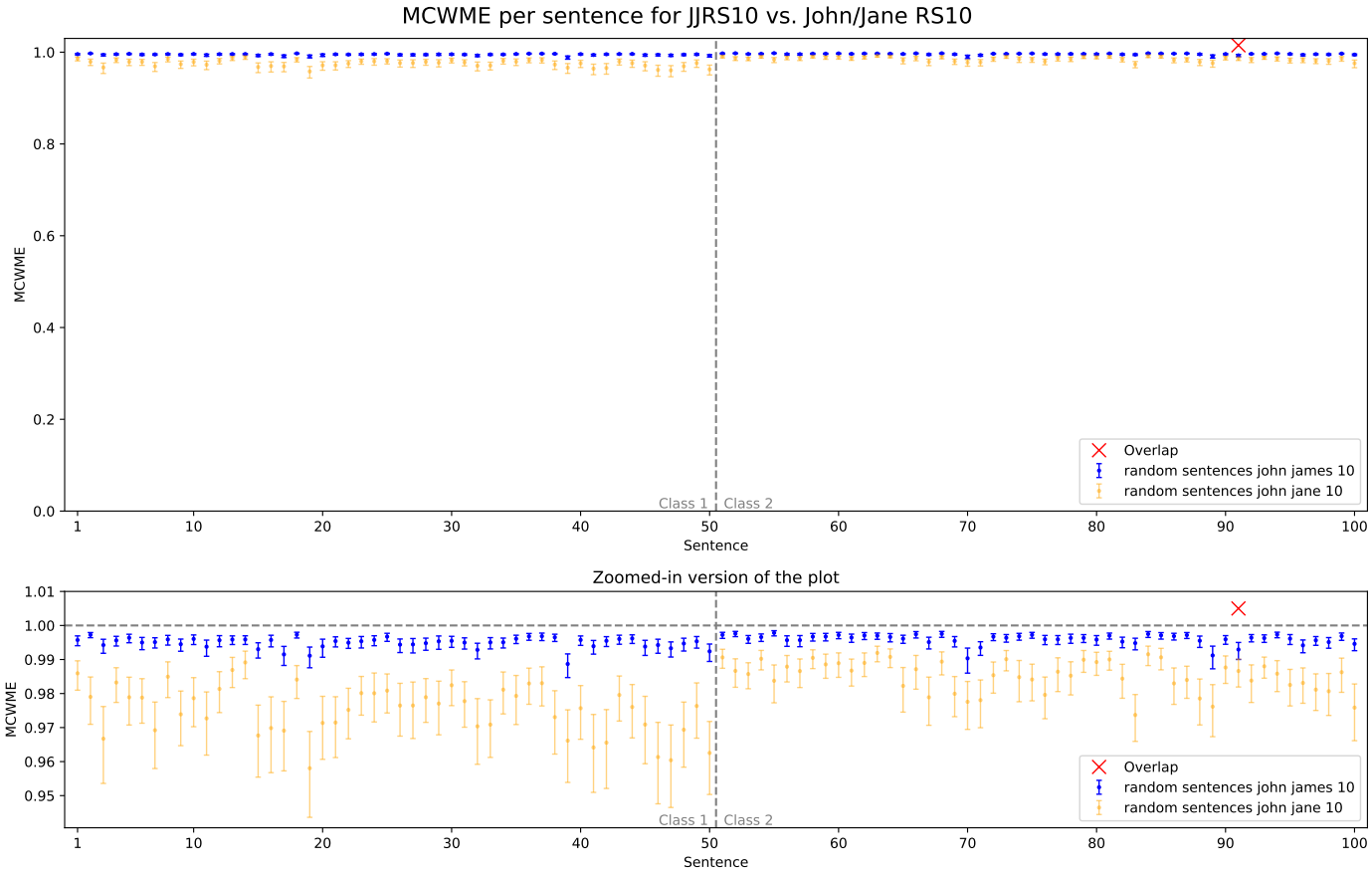

Figure A.1: MCWME per Sentence for *JJRS10* vs. *John/Jane RS10*

Let's first compare the two experiments together. We can see that the *John/Jane* experiment (in orange) performs significantly worse than the other (in blue) on both classes. As a reminder, the only thing that changed between these two datasets is that the `James` keyword was replaced by the `Jane` keyword, replacing a masculine name by a feminine one. This is extremely curious as the class that seems to suffer the most from this is actually the `John` class. I expected the `John` class to stay as stable as before like a baseline and the `Jane` class to struggle due to sentences linking themselves more with a feminine name than with a masculine name. Regardless of why this appears for the `John` class, the `Jane` class performs significantly worse than the `James` class from *JJRS10*. It seems like, even though the sentences have been constructed off neutral Wikipedia sentences, the gender of

Figure A.2: Attention Maps of Mean Explanation of First Sentence in; 1st: *JJRS10* John; 2nd: *JJRS10* James; 3rd: *John/Jane RS10* John; 4th: *John/Jane RS10* Jane

the name seems to play a role. This variability doesn't come from the selection of the first word as the top-1 explanation is 1.0 for every sentence on every task. It must therefore come from the attribution given to the other words. The model likely hallucinates some connections.

Within the *John/Jane RS10* experiment, we can clearly see the trend that the `Jane` class is more stable than the `John` class. This is extremely bizarre and I cannot explain it. I do not have any meaningful insights to bring forth. Those results are puzzling and I think they are interesting as is. Looking at this issue more in-depth could be an interesting research direction.

I acknowledge the big limitations surrounding this experiment as this topic hasn't been covered in a systematic enough way. I should have first of all verified that the quality of the John/James explanations was due to their gender and not just because those names are very frequent using another 2-masculine names baseline (John/Jack). Secondly, launching only one experiment with a feminine name can be problematic as the results could just be linked to that particular name and not all feminine names. However, even through those limitations still lie some strikingly weird results.

A.3.2 No Discriminant Words - Long Sentences

A way to deal with this introduced bias could be to "drown" it. Increasing the size of the length will likely make it more difficult for the model to find the broken context and dilute the impact on the bias. Figure A.3 shows the MCWME and its confidence interval for the *JJRS50* and *JNRS50v1* datasets. The results are as expected just like the ones presented before. Indeed, the `John` class behaves perfectly in both tasks. The class which contains no discriminant words still has a lot of variability. When looking at attention maps, shown in Figure A.4, we can see that the models still manage to target the added word and to find the broken context which indicates that the results in the Error bar are an overestimation of the stability because there is a bias in the dataset.

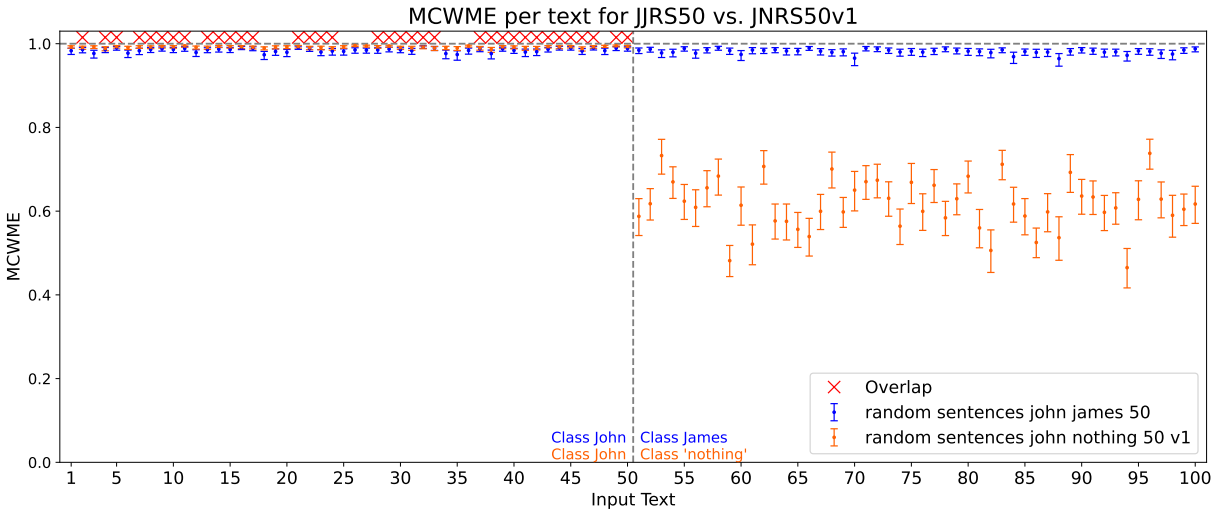

Figure A.3: MCWME per text for *JJRS50* vs. *JNRS50v1*

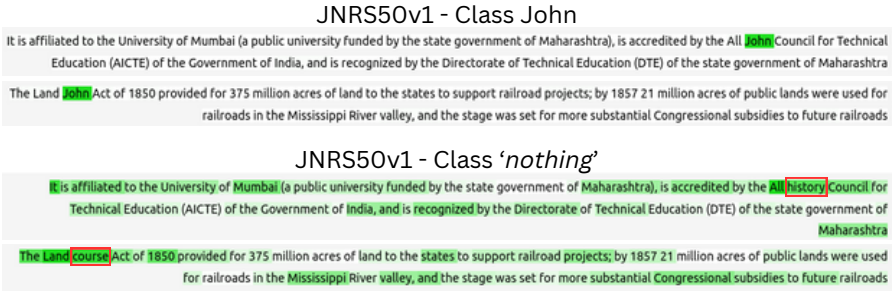

Figure A.4: Mean Explanations of *JNRS50v1* sentences

Appendix B

Fine-Tuning

B.1 Training Loop

Algorithm 1 Training Loop

Data:
`trainSet` divided in `nbTrainBatches` batches
`valSet` divided in `nbValBatches` batches
`testSet` divided in `nbTestBatches` batches
Algorithm:
Send `model` to the GPU
Initialize the AdamW `optimizer`
Initialize a ReduceLROnPlateau `scheduler`
for Epoch in range(`nbEpochs`) **do**
    Put `model` in training mode
    for `Batch` in `trainSet` **do**
        Set `model` gradients to 0
        Load `Batch` to the GPU
        Perform a forward pass and monitor the `loss` of `Batch`
        Compute the `gradients` of the `loss` wrt. every parameter
        Modify `model` parameters using the previously-computed `gradients`
    end for
    Put `model` in evaluation mode
    for `Batch` in `valSet` **do**
        Load `Batch` to the GPU
        Compute the `loss` and the `logits` of the `Batch`
        Compute the `accuracy` by comparing the `logits` with the `trueLabels`
    end for
    Step the `scheduler`
end for
return `model`, `accuracies`

B.2 Testing Loop

Algorithm 2 Testing Loop

Data:
`testSet` divided in `nbTestBatches` batches
Algorithm:
Send `model` to the CPU
Put `model` in evaluation mode
for `Batch` in `testSet` **do**
    Load `Batch` to the CPU
    Use the `model` to get `logits`
    Compute the `accuracy` by comparing the `logits` with the `trueLabels`
end for
return `accuracies`

Appendix C

Mean Correlation with Mean Explanation

Algorithm 3 Compute MCWME for 1 text (with LOO strategy)

Data:
Models [`nbModels`]
Sentence [`nbWordsPerText`]
Explanations [`nbModels` × `nbWordsPerText`]
Algorithm:
`CWME` ← empty list
for `model` ∈ Models **do**
    `currExpl` ← explanation from `model`
    `meanExpl` ← mean of all explanations except for `model`
    Append `Corr(currExpl, meanExpl)` to `CWME`
end for
`fisherCWME` ← `fisherZTransform(CWME)`
`fisherMeanCWME` ← `Mean(fisherCWME)`
`fisherCI` ← `ConfInterval(fisherCWME)`
`MeanCWME` ← `inverseZTransform(fisherMeanCWME)`
`MeanCI` ← `inverseZTransform(fisherMeanCI)`
return `MeanCWME`, `MeanCI`

Appendix D

InfOpinions Truncation

The text :

> Les nouveaux moyens de communication ont poussé les autorités académiques à communiquer pour calmer les esprits . Cela s' est fait via un e-mail envoyé à l' ensemble des étudiants . Il fallait faire taire certaines fausses rumeurs , explique Marcel Raymond , vice-recteur aux affaires étudiantes : " Il y avait déjà des rumeurs qui circulaient en disant que c' est un étudiant de telle année , il y a ceci , il y a ça . Moi j' ai décidé d' envoyer un mail à tous les étudiants . C' est vrai qu' avec facebook les rumeurs circulent très vite et on préfère donner l' information correcte " . Les facultés de Namur tenaient aussi à rappeler l' existence de ce service médico psychologique . En temps normal il est utile pour des questions d' orientation des étudiants , en période d' examen il peut aider à trouver une réponse au stress . Durant les périodes de blocus , les psychologues ont beaucoup plus de travail . RTBF

has been transformed to:

> Les nouveaux moyens de communication ont poussé les autorités académiques à communiquer pour calmer les esprits . Cela s' est fait via un e-mail envoyé à l' ensemble des étudiants . Il fallait faire taire certaines fausses rumeurs , explique Marcel Raymond , vice-recteur aux affaires étudiantes : " Il y avait déjà des rumeurs qui circulaient en disant que c' est un étudiant de telle année , il y a ceci , il y a ça ."

Appendix E

Attention Maps

E.1 JJBoW50

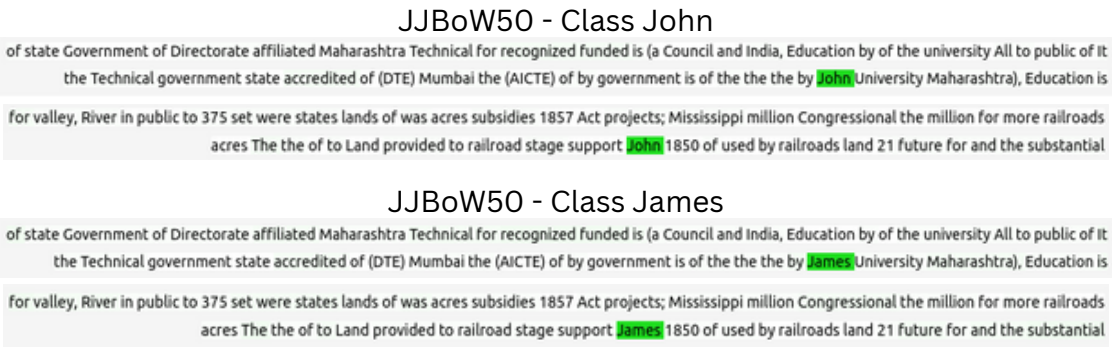

Figure E.1: Attention Maps of the Mean Explanation for texts in *JJBoW50*

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**