LLM Explainability via Attributive Masking Learning

Oren Barkan[*1] **Yonatan Toib**[*1] **Yehonatan Elisha**[*2]
Jonathan Weill[2] **Noam Koenigstein**[2]

[1]The Open University, Israel
[2]Tel Aviv University, Israel

Abstract

In this paper, we introduce Attributive Masking Learning (AML), a method designed for explaining language model predictions by learning input masks. AML trains an attribution model to identify influential tokens in the input for a given language model's prediction. The central concept of AML is to train an auxiliary attribution model to simultaneously 1) mask as much input data as possible while ensuring that the language model's prediction closely aligns with its prediction on the original input, and 2) ensure a significant change in the model's prediction when applying the inverse (complement) of the same mask to the input. This dual-masking approach further enables the optimization of the explanation w.r.t. the metric of interest. We demonstrate the effectiveness of AML on both encoder-based and decoder-based language models, showcasing its superiority over a variety of state-of-the-art explanation methods on multiple benchmarks. Our code is available at: https://github.com/amlconf/aml

1 Introduction

AI is transforming every facet of modern life, sparking innovation and driving progress across diverse fields. From groundbreaking advancements in computer vision (He et al., 2016; Dosovitskiy et al., 2020; Liu et al., 2022; Carion et al., 2020; Assran et al., 2023; Barkan et al., 2023e) and natural language processing (NLP) (Mikolov et al., 2013; Vaswani et al., 2017; Devlin et al., 2018; Brown et al., 2020; Barkan, 2017; Barkan et al., 2020f,e, 2021b; Ginzburg et al., 2021; Malkiel et al., 2020, 2022b) to audio synthesis (Engel et al., 2020; Kong et al., 2020; Barkan and Tsiris, 2019; Barkan et al., 2019b, 2023g) and recommender systems (Barkan and Koenigstein, 2016; He et al., 2017; Ben-Elazar et al., 2017; Wang et al., 2019; He et al., 2020; Barkan et al., 2019a, 2020a,b,d, 2021d, 2023f; Katz et al., 2022).

In the field of NLP, the Transformer architecture (Vaswani et al., 2017), has given rise to increasingly sophisticated language models (LMs) (Devlin et al., 2018; Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023; Albert Q. Jiang, 2023). These models, characterized by their growing size and complexity, have become pivotal components in numerous applications. Consequently, the demand for a comprehensive understanding of the decision-making processes employed by LMs has surged. However, the inherent complexity of these models often obscures the transparency of their predictions, necessitating the development of explainability methods to reveal the underlying factors in the input influencing their outputs. As a result, a large variety of explanation methods were developed (Ribeiro et al., 2016; Sundararajan et al., 2017; Lundberg and Lee, 2017; Abnar and Zuidema, 2020; Ferrando et al., 2022; Barkan et al., 2021c; Malkiel et al., 2022a). In tandem with the evolution of explanation methods, various explanation metrics and benchmarks were proposed to quantitatively assess the efficacy of explainability methods (Samek et al., 2017; DeYoung et al., 2020). However, consensus within the literature regarding the ultimate explanation metric remains elusive, with each metric providing unique insights into different facets of the explanation quality.

In this work, we introduce the Attributive Masking Learning (AML) framework that enables explanation via the generation of an attribution map tailored to the specific metric at hand. AML involves training an auxiliary *attribution* model to identify influential tokens in the input for a given LM's prediction. This attribution model is specifically trained to achieve two goals simultaneously: 1) masking as much input data as possible while ensuring the LM's predictions on the masked and

[*] Equal contribution.

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 9522–9537

original inputs remain closely aligned, and 2) inducing a significant change in the model's prediction when the inverse of the same mask is applied to the input. These two objectives bear resemblance to faithfulness metrics (DeYoung et al., 2020). The AML optimization is self-supervised and consisting of a two-phase process: a **mandatory** offline AML pretraining phase (pAML), followed by an **optional** online AML finetuning phase (fAML). In the pAML phase, the attribution model is pretrained on a training set of examples and their respective predictions (generated by the LM to be explained). The pretraining process aims to enable the model to generate meaningful attribution maps by optimizing the AML dual-mask loss while monitoring the metric of interest. The pAML phase occurs once and offline. At the end of the pAML phase, an attribution map can be generated for any input through a **single** forward pass of the input and its prediction via the pretrained attribution model. For further refinement of the attribution map, an optional subsequent AML finetuning (fAML) step can be employed in an online manner by continuously finetuning the pretrained attribution model on a specific instance w.r.t. the metric of interest, hence producing metric-adapted explanations. We present a comprehensive evaluation encompassing 13 explanation methods, 5 LMs (including the latest Llama2 and Mistral models (Touvron et al., 2023; Albert Q. Jiang, 2023)), 4 datasets, and 5 explanation metrics. Our results demonstrate the effectiveness of AML. Specifically, pAML exhibits competitive performance with a notably fast runtime, while fAML surpasses all state-of-the-art approaches, achieving significant improvements in accuracy across multiple benchmarks. In summary, our contributions include: 1) We introduce AML method, facilitating a novel dual-mask optimization of the attribution map w.r.t. the specific metric of interest. 2) We present a comprehensive investigation encompassing a wide range of datasets, explanation methods, and evaluation metrics, applied to various LLM architectures. Our findings show that AML set a new state-of-the-art in LLM explainability.

2 Related Work

Explainability research has made significant strides over the past decade, introducing a wide range of methods across various application domains (Fong et al., 2019; Simonyan et al., 2013; Fong and Vedaldi, 2017; Selvaraju et al., 2017; Zhou et al., 2018; Barkan et al., 2020c, 2021c,a, 2023d,c,a,b; Gaiger et al., 2023; Barkan et al., 2024; Malkiel et al., 2022a; Chefer et al., 2021a,b). Gradient-based methods rely on the gradients of the model's prediction score concerning the input embeddings. Basic gradient methods, such as Vanilla Gradients (Simonyan et al., 2013) and GradientXInput (Shrikumar et al., 2016), operate on the principle of gradient computation. Integrated Gradients (IG) (Sundararajan et al., 2017) is a path-integration method, computing gradients on interpolated points along a straight line between the data and an uninformative baseline. Approaches like DeepLift (Shrikumar et al., 2017) and GradientShap (Lundberg and Lee, 2017) can be viewed as approximations of IG. A recent addition, Sequential Integrated Gradients (SIG) (Enguehard, 2023), offers a path-integration method that addresses concerns about altered sentence meaning by computing the importance of each word while keeping other words fixed and creating interpolations between the baseline and the word of interest.

Relevance-Decomposition methods break down the network's representation into vectors, each influencing the model's prediction differently. GlobEnc (Modarressi et al., 2022) and ALTI (Ferrando et al., 2022) incorporate local-level decomposition, aggregating resulting vector norms using rollout (Abnar and Zuidema, 2020) to construct global-level explanations. The recent introduction of DecompX (Modarressi et al., 2023) showcases state-of-the-art results by constructing decomposed token representations and sequentially propagating them through the model without interlayer mixing.

Perturbation-based methods, including well-known techniques like LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017), introduce perturbations to individual inputs or neurons and observe their impact on subsequent neurons in the network. Solvex (Zhou and Shah, 2023), on the other hand, is a search method that applies beam-search as an approximation to an exhaustive search on the ranking of tokens to find the best explanation per metric. However, a major limitation of Solvex is its inefficiency due to the application of naive beam-search, resulting in significantly slower runtime compared to methods that require a single or a small number of passes through the model, such

as vanilla gradients and IG.

In contrast to the aforementioned works, AML utilizes an auxiliary attribution model that learns to preserve and alter the prediction of the model to be explained by masking out less and more influential elements in the input, respectively and simultaneously. This unique process facilitates the creation of metric-adapted explanations through a single pass (pAML) or, optionally, a small number of additional passes (fAML) through the model during inference.

3 Attributive Masking Learning

Let $\mathcal{V}$ represent the vocabulary, and consider $v = (v_i)_{i=1}^k$ as a sequence of k tokens, representing an input text, where each $v_i \in \mathcal{V}$. Subsequently, we define $\mathbf{x}_v = (\mathbf{x}_{v_i})_{i=1}^k$ as a sequence of embeddings, where $\mathbf{x}_{v_i} \in \mathbb{R}^d$ represents the token v_i.

In this work, our focus is on classification tasks. Therefore, we assume a model F that takes an input $\mathbf{x}_v$ and produces an output $F(\mathbf{x}_v) \in [0,1]^c$, where $F_i(\mathbf{x}_v)$ is the probability assigned to class i, and c is the number of available classes. F can take the form of a LM utilizing a classification head, which outputs probabilities for each class in the specific task. This involves either finetuning the LM or using it as a backbone for transfer learning of specific task (Liu et al., 2019).

The majority of recent LM architectures, also known as large language models (LLMs) are decoder-based, performing a completion task by generating a sequence of tokens in the output (Brown et al., 2020; Touvron et al., 2023). While these LLMs can also be finetuned by employing classification heads, a more common approach is classification via completion: the LLM is prompted with the specific task either in a few-shot or zero-shot and instructed to output a token representing the correct class. For example, in sentiment analysis, if the relevant classes are defined to the LLM (via prompt) as 'pos' and 'neg', one can compute the logit scores for the tokens 'pos' and 'neg' and apply softmax to obtain the probabilities of the token associated with each individual class. We note that this approach can be implemented either before or after finetuning the LLM on the relevant dataset in a completion manner. In this work, we conduct experiments in various settings, including finetuning with classification heads on top of the LLM as well as classification via completion (decoder-based models).

AML enables the attribution of the prediction $F(\mathbf{x}_v)$ to individual tokens in $\mathbf{x}_v$ by training an *attribution* model G_θ (parameterized by θ). This model learns to mask as much data as possible from $\mathbf{x}_v$ while preserving important information, ensuring that the predictions on the *masked* input and the *inversely masked* input[1] match and diverge from the prediction on the original input $F(\mathbf{x}_v)$, respectively.

The attribution model G_θ takes the input $\mathbf{x}_v$ and information[2] regarding $F(\mathbf{x}_v)$ denoted by $\mathbf{y}_v$, and produces an attribution map

$$\mathbf{a}_v := G_\theta(\mathbf{x}_v, \mathbf{y}_v). \tag{1}$$

$\mathbf{a}_v \in [0,1]^k$ is a k-dimensional vector, with the i-th entry $\mathbf{a}_v[i]$ representing the attribution score assigned to the token $\mathbf{x}_{v_i}$ in the input $\mathbf{x}_v$. Subsequently, $\mathbf{a}_v$ is employed to derive the masked and inversely masked versions of the input,

$$\mathbf{x}'_v := M(\mathbf{a}_v, \mathbf{x}_v) \quad \text{and} \quad \mathbf{x}''_v := M(1 - \mathbf{a}_v, \mathbf{x}_v), \tag{2}$$

respectively, where M is a function that masks the information in $\mathbf{x}_v$ based on the provided attribution map ($\mathbf{a}_v$ or $1 - \mathbf{a}_v$ as stated in Eq. 2). Given that $\mathbf{x}_v$ is a sequence of tokens, the function M masks each individual token $\mathbf{x}_{v_i}$ according to its attribution score $\mathbf{a}_v[i]$ such that a lower $\mathbf{a}_i$ value indicates a stronger masking effect on $\mathbf{x}_{v_i}$.

The masking operation is carried out in a token-wise manner and can be implemented as either the replacement or omission of the token to be masked in various forms, including, but not limited to: 1) Replacement by designated tokens, e.g., the <MASK> or the <UNK> token in encoder-based models and decoder-based models, respectively. 2) Replacement by a randomly drawn token. 3) Replacement by another token predicted based on the surrounding context of the token to be masked. 4) Replacement by a new token embedding representing the mask token, which is learned as part of the optimization process. 5) Omission of the token to be masked from the original sequence.

In this work, we have opted for option 1, while

[1] The inversely masked input is obtained by masking the input according to the inverse (complement) of the mask

[2] $\mathbf{y}_v$ can be defined in two ways: either as the original prediction $F(\mathbf{x}_v)$ or as a one-hot vector representing a specific class of interest. Typically, this class corresponds to the one with the highest score in $F(\mathbf{x}_v)$.

leaving the exploration of other options for future research[3]. Therefore, we assume the existence of a unique embedding $\mathbf{m} \in \mathbb{R}^d$ used for masking information, such that every token representation to be masked is either replaced by $\mathbf{m}$ (hard masking) with some masking probability or combined with $\mathbf{m}$ (soft masking) according to the masking probability. In this study, we opt for soft masking, where each token representation in the masked input $\mathbf{x}'_v$ is obtained by a convex combination with the mask token $\mathbf{m}$ according to the attribution map $\mathbf{a}_v$, and therefore, M is implemented as follows:

$$M(\mathbf{a}_v, \mathbf{x}_v) := (\mathbf{a}_v[i]\mathbf{x}_{v_i} + (1 - \mathbf{a}_v[i])\,\mathbf{m})_{i=1}^{k}\,. \tag{3}$$

Accordingly, the inversely masked input $\mathbf{x}''_v$ from Eq. 2 is obtained by employing Eq. 3 using the inverse of the attribution map, $1 - \mathbf{a}_v$.

3.1 AML optimization and inference

Our goal is to generate an attribution map highlighting the tokens in the input $\mathbf{x}_v$ that most significantly contribute to and rationalize the prediction $(\mathbf{x}_v)$. To this end, we train G_θ to produce an attribution map $\mathbf{a}_v$ by minimizing the following loss function:

$$\begin{aligned}\mathcal{L}(\mathbf{x}_v, \mathbf{y}_v) =& \lambda_p \mathcal{L}_p(F(\mathbf{x}'_v), \mathbf{y}_v) + \lambda_a \mathcal{L}_a(\mathbf{a}_v) \\ &+ \lambda_{inv} \mathcal{L}_{inv}(F(\mathbf{x}''_v), \mathbf{y}_v),\end{aligned} \tag{4}$$

where λ_p, λ_{inv} and λ_a are hyperparameters controlling the inherent trade-off between the three loss terms. $\mathcal{L}_p$ is the cross-entropy loss between the prediction on the masked input $F(\mathbf{x}'_v)$ and $\mathbf{y}_v$ (derived from the prediction on the original input $F(\mathbf{x}_v)$ as explained earlier):

$$\mathcal{L}_p(F(\mathbf{x}'_v), \mathbf{y}_v) = -\sum_{j=1}^{c} \mathbf{y}_v[j] \log(F(\mathbf{x}'_v)[j]). \tag{5}$$

$\mathcal{L}_p$ encourages the prediction on the masked input to align with the prediction on the original input.

The $\mathcal{L}_{inv}$ loss is defined as follows:

$$\mathcal{L}_{inv}(F(\mathbf{x}''_v), \mathbf{y}_v) = -\log(1 - F(\mathbf{x}''_v)[y]), \tag{6}$$

where y is the entry in $\mathbf{y}_v$ receiving the highest probability. $\mathcal{L}_{inv}$ ensures that if the tokens with the highest attribution scores are masked out (using the inverse map $1 - \mathbf{a}_v$), the resulting prediction significantly differ from the prediction on the original input. This is since the probability of the class y (the same class that recieved the highest score in the prediction $F(\mathbf{x}_v)$) is encouraged to be minimized when the prediction is made on the inversely masked input $F(\mathbf{x}''_v)$. In our initial experimentation, we investigated other alternatives by setting $\mathcal{L}_{inv}$ to $F(\mathbf{x}''_v)[y]$ or to the negative entropy of $F(\mathbf{x}''_v)$. Yet, these alternatives yielded slightly worse results.

Lastly, $\mathcal{L}_a$ is the binary cross entropy (BCE) between $\mathbf{a}_v$ and the zero prior:

$$\mathcal{L}_a(\mathbf{a}_v) = -\frac{1}{k}\sum_{j=1}^{k} \log(1 - \mathbf{a}_v[j]). \tag{7}$$

$\mathcal{L}_a$ serves as a regularization term, encouraging sparsity in attribution maps. It is worth noting that in our initial experiments, we observed that L1 regularization performs similarly to BCE. The optimization of $\mathcal{L}$ (Eq. 4) proceeds with gradient descent on the parameters θ of G_θ.

In practical terms, we employ a two-phase optimization approach: a mandatory *pretraining* phase (pAML), followed by an optional *finetuning* phase (fAML) that can be employed for further refinement of the attribution map in inference time. During the pretraining phase, we minimize $\frac{1}{|T|}\sum_{v \in T} \mathcal{L}(\mathbf{x}_v, \mathbf{y}_v)$ on a training set T via mini-batch training, resulting in a pretrained attribution model G_{θ_T}. Following pretraining, we can simply use G_{θ_T} during inference to generate an attribution map for any input $\mathbf{x}_v$ with a single forward pass through G_{θ_T}, i.e., $\mathbf{a}_v = G_{\theta_T}(\mathbf{x}_v, \mathbf{y}_v)$.

However, for further refinement, we propose instance-specific finetuning of the attribution model during inference. In this phase, we finetune the pretrained parameters θ_T for a specific instance $\mathbf{x}_v$ by minimizing $\mathcal{L}(\mathbf{x}_v, \mathbf{y}_v)$ (Eq. 4). This results in a finetuned attribution model G_{θ_v} and subsequently, a more refined attribution map $\mathbf{a}_v = G_{\theta_v}(\mathbf{x}_v, \mathbf{y}_v)$.

A notable advantage of AML lies in its capability to monitor a specific metric of interest throughout the optimization process. Specifically, it allows for the selection of the attribution model and attribution map that perform the best on the metric at hand during the pretraining and finetuning phases, respectively. Finally, it is important to clarify that although the explained model F participates in the

[3]Our empirical findings indicate that option 1 already achieves state-of-the-art results. However, it is noteworthy that a study by (Zhao and Shan, 2024) suggests that option 3 may yield even better results.

optimization process, its parameters remain fixed throughout the entire optimization procedure.

3.2 AML complexity

The computational complexity of the pretraining phase depends on the size of the training set T and the number of training epochs. In our experiments, utilizing a training set size of about 1,000 examples proved sufficient. At inference time, once the pretrained attribution model G_{θ_T} is established, the computational complexity of pAML involves a single forward pass, hence comparable to the complexity of vanilla gradient method. When finetuning is applied (fAML), the best-performing attribution map typically emerges after a very small number of finetuning steps. If we denote the complexity of a single forward pass through G as R, the inference complexity when applying n finetuning steps is approximately $(4n + 1)R$ (assuming forward and backward passes take approximately the same time, and that the propagation time via F and G is consistent).

From a practical standpoint runtime comparison tests are reported in Sec. A.3 in the Appendix. These runtimes are feasible even for real-time applications. Therefore, for scenarios where speed is paramount, pAML may be the preferred option. Conversely, when precision is of utmost importance and a slight increase in runtime is acceptable, fAML emerges as the optimal choice.

Finally, it is worth noting that the attribution model G is not limited to the same architecture as the explained model F. This flexibility allows for employing a much lighter model G, resulting in even faster generation of the attribution map during inference. For instance, in our experiments, we explained Llama2 and Mistral using a much lighter RoBERTa architecture for G.

3.3 AML implementation

Our experiments encompass both encoder-based and decoder-based models.

Encoder-based Models For BERT, RoBERTa, and DistilBERT (Sanh et al., 2019), we set F to the finetuned version of each respective model[4]. Since BERT, RoBERTa, and DistilBERT support the <MASK> token, we designate $\mathbf{m}$ as the embedding associated with this token[5].

The attribution model G_θ employs the pretrained version of the corresponding model as a backbone, augmented with an additional shared MLP head. This MLP head is placed on top of each token representation generated by the last encoder block of the backbone. The MLP head is implemented as a $d \to d \to 1$ network (where d is the token embedding dimensionality), incorporating hyperbolic tangent activation on the hidden layer and sigmoid activation on the output layer, yielding the attribution score for each token. It is essential to highlight that the same MLP head is applied to each token representation produced by the backbone. Subsequently, all attribution scores are consolidated into a vector representing the attribution map. Throughout the AML pretraining and finetuning phases (Sec. 3.1), both the backbone and the MLP head are optimized w.r.t. $\mathcal{L}$ (Eq. 4).

Decoder-based models In this work, we experimented with the Llama2 7B (Touvron et al., 2023) and Mistral 7B (Albert Q. Jiang, 2023) models, setting F as the pretrained model. These models support the <UNK> token, hence we set $\mathbf{m}$ to the embedding associated with this token.

Llama2 and Mistral are general-purpose models for language generation and understanding. Although few-shot prompting is typically more aligned with the common usage of LLMs, rather than finetuning for specific datasets, we observed limited performance for EMR task. Therefore, for this task, we finetuned the models on the dataset using LoRA. For the remaining classification tasks, we employed a few-shot prompting approach, where task-relevant prompts and multiple examples were provided to guide the model's responses. Specifically, we prompt the LLM with the definition of the classification task, including the tokens representing each class and several shots. This prompt, denoted as u, instructs the LLM to predict the specific token corresponding to the correct class as the initial token generated in the output. The input example v was then concatenated with the prompt u, forming $[u, v]$, which was fed to the LLM. Finally, we utilize the logit scores associated with the relevant class tokens from the LLM's initial token prediction as the output vector, followed

[4] All encoder-based models utilize classification heads and undergo finetuning for specific tasks.

[5] We observed that using <MASK> performs better than other alternatives such as the <PAD> token, omitting the token, or replacing the token with a random token.

by a softmax operation. The exact prompting for each task is outlined in our GitHub repository.

The attribution model G_θ adopts the aforementioned encoder-based architecture, utilizing a pretrained RoBERTa backbone with an MLP head. Since the objective of G_θ is to produce attribution scores for the tokens in the input v, we input G_θ with $\mathbf{x}_v$ (and not $[\mathbf{x}_u, \mathbf{x}_v]$), thereby producing attribution scores exclusively for the tokens in v. Consequently, the masking operation, when applied, pertains only to the tokens in v. In future work, we plan to explore attribution for the u part, examining the attribution of tokens in the shots as well. Furthermore, due to subtle differences between the RoBERTa (source) and Llama2 (target) tokenizers, we employed one-to-many (source token is split into multiple target tokens) and many-to-one (source tokens are merged into a single target token) mappings. Specifically, in the case of source token split (one-to-many), the attribution score was replicated for each target token. In case of token merge, the attribution score for the target token is set to the maximum[6] among the source tokens.

$\mathbf{y}_v$ encoding As mentioned earlier, $\mathbf{y}_v$ can be defined in two distinct ways: either as the original predicted probability distribution $F(\mathbf{x}_v)$ or as a delta distribution corresponding to the class associated with the highest score in $F(\mathbf{x}_v)$. In this work, we opt for the former option. To encode $\mathbf{y}_v$, we introduce a dedicated set of c embeddings denoted as $Z = \{\mathbf{z}_j\}_{j=1}^{c}$, where $\mathbf{z}_j \in \mathbb{R}^d$ represents the embedding associated with class j. Additionally, a special embedding $\mathbf{z}_0 \in \mathbb{R}^d$ is introduced, symbolizing the prediction information to the attribution model (akin to the segment embedding in BERT). In our implementation, G_θ processes $\mathbf{x}_v$ and $\mathbf{y}_v$ as follows: Firstly, it encodes $\mathbf{y}_v$ by computing a convex combination of the class embeddings based on their respective probabilities in $F(\mathbf{x}_v)$ and adds them to $\mathbf{z}_0$ to produce $\mathbf{z}_v = \mathbf{z}_0 + \sum_{j=1}^{c} F(\mathbf{x}_v)[j]\mathbf{z}_j$. Subsequently, it appends $\mathbf{z}_v$ to $\mathbf{x}_v$ and forwards it as the input to the specific backbone used by G_θ. It is crucial to clarify that the class embeddings in Z and $\mathbf{z}_0$ are not part of the original vocabulary $\mathcal{V}$ supported by the backbone. Consequently, learning these embeddings is a necessary component of the optimization process during the AML pretraining phase. However, during the AML finetuning phase, both Z and $\mathbf{z}_0$ remain frozen.

[6] Our experimentation indicates that mean aggregation performs on par.

AML hyperparameters The hyperparameters λ_p, λ_a, and λ_{inv} (Eq. 4) are adjusted automatically (per metric) during the pretraining phase on a separate validation set and maintain their fixed values throughout the finetuning process (if applied). While it is feasible to continuously adjust hyperparameters during the finetuning process for specific examples, in this work, we abstained from doing so to prioritize inference time efficiency of fAML. In Sec. A.2 in the Appendix, we further ablate on the necessity of each loss term in Eq. 4 associated with its corresponding hyperparameter. For the exact optimization details, the reader is referred to Sec. 4 and our GitHub repository.

4 Experimental Setup and Results

Datasets We evaluate the explanation methods on 4 different datasets, offering a comprehensive evaluation across a wide spectrum of classification tasks and text lengths: **SST2** (Socher et al., 2013), Rotten Tomatoes (**RTN**) (Pang and Lee, 2005), and **IMDB** (Maas et al., 2011) are binary sentiment classification in short, medium-sized, and long texts, respectively. Emotion Recognition (**EMR**) (Saravia et al., 2018): Emotion classification (6 classes: Sadness, Joy, Love, Anger, Fear, and Surprise) predominantly in short texts. To ensure transparency and reproducibility, we processed datasets using the HuggingFace library (Wolf et al., 2019), employing a procedure similar to that outlined in (Enguehard, 2023). Furthermore, our evaluation includes the annotated Movie Reviews datasets sourced from the ERASER benchmark (DeYoung et al., 2020), where ground-truth annotations are provided by human annotators. These datasets are used for assessing the concordance between the generated explanations and human-annotated explanations. The code for data processing is available in our GitHub repository. The results of this evaluation are presented in the Appendix.

Models Our evaluation includes the following models: BERT-Base (Devlin et al., 2018), RoBERTa-Base (Liu et al., 2019), DistilBERT (Sanh et al., 2019), Llama2 7B (Touvron et al., 2023), and Mistral 7B (Albert Q. Jiang, 2023). For the first three models, we employed their finetuned versions specific to each dataset. In

contrast, Llama2 and Mistral were evaluated in few-shot prompting mode[7] for the SST2, RTN, and IMDB tasks, which aligns with the typical usage of LLMs, as discussed in Sec. 3. Due to their underperformance on the EMR task, we finetuned both Llama2 and Mistral using LoRA and report the results accordingly. All model links are available in our GitHub repository.

Evaluated explanation methods Our evaluation covers a diverse set of explanation methods: **SIG** (Enguehard, 2023), GlobEnc (**GLOB**) (Modarressi et al., 2022), **ALTI** (Ferrando et al., 2022), DecompX (**DCMP**) (Modarressi et al., 2023), GradientXInput (**GXI**) (Shrikumar et al., 2016), Integrated Gradients (**IG**) (Sundararajan et al., 2017) DeepLift (**LIFT**) (Shrikumar et al., 2017), Solvex (**SLVX**) (Zhou and Shah, 2023), GradientShap (**SHAP**) (Lundberg and Lee, 2017), and **LIME** (Ribeiro et al., 2016). For each of these methods, we used the codebase and hyperparameter search procedure recommended by the original works. Additionally, we introduce an explanation method based on LLMs (**LLM**). In this approach, we prompt the finetuned, open-source versions of Llama-Instruct and Mistral-Instruct with the task and input, instructing the models to explain their predictions by ranking the tokens in the input example according to their importance.

Lastly, our AML method was tested in its two modes: using the pretrained attribution model G_{θ_T} (**pAML**), and the instance-specific finetuned attribution model G_{θ_v} (**fAML**). The hyperparameters λ_p, λ_a and λ_{inv} were optimized during the pretraining phase using a dedicated validation set for each dataset-metric combination. The hyperparameters optimization was conducted using the default TPE sampler from the Optuna package (Akiba et al., 2019). All hyperparameters remained fixed during the finetuning phase. AML loss optimization utilized the AdamW optimizer. Throughout this process, we monitored the metric of interest, selecting the best-performing attribution model and attribution map in the pretraining and finetuning phase, respectively. All experiments were executed on an NVIDIA DGX 8xA100 server. The exact optimization configuration as well as dataset splits are available in our GitHub repository

[7]Prompts for decoder-based models are provided in our GitHub repository.

Evaluation measures For quantitative assessment of the explanation methods, we followed the evaluation protocol from recent works (Enguehard, 2023; Ferrando et al., 2022) and report results for the following set of metrics: Log-Odds (**LO**) (Shrikumar et al., 2017), Sufficiency (**Suff**), Comprehensiveness (**Comp**) and Area Over the Perturbation Curve (AOPC) for Sufficiency (**A-S**) and Comprehensiveness (**A-C**) (DeYoung et al., 2020). For Suff, LO and A-S the lower the better, while for Comp and A-C the higher the better. Unless specified otherwise, we use the same metric settings as detailed in the referenced papers. A detailed description of the metrics appears in the Appendix (Sec. A.5). While this set of metrics provides a comprehensive assessment of the faithfulness of the produced explanations, we further present results for the agreement with ground-truth explanations extracted by human annotators (DeYoung et al., 2020). These agreement tests evaluate the utility of the generated explanations from a human perspective.

4.1 Results

Table 1 presents results for all combinations of encoder-based model, explanation method, dataset, and metric. We identify the following trends: fAML consistently emerges as the top-performing method in most of the scenarios. The runners-up are typically DCMP, SLVX, and SIG, with pAML generally securing third place. We further observe that DCMP consistently underperforms with DistilBERT. These results demonstrate the superiority of our AML approach, establishing it as the new state-of-the-art. It is important to note that, due to its inefficiency with longer texts, SLVX was excluded from evaluation on the IMDB dataset. The substantial performance gap between fAML and pAML underscores the importance and effectiveness of the finetuning phase in AML. Nonetheless, pAML alone remains competitive, offering reduced runtime with strong performance.

Table 2 presents results for decoder-based models. To maintain focus, we report results only for a subset of the methods from Tab. 1. Some methods, such as DCMP and ALTI, were excluded from this comparison due to their requirement for substantial model architecture modifications, and lack of support for Llama2 and Mistral models in their released Git repositories. Similarly, SIG (for the IMDB dataset) and SLVX were excluded due to

		RoBERTa					DistilBERT					BERT				
		Suff↓	**LO↓**	**Comp↑**	**A-S↓**	**A-C↑**	**Suff↓**	**LO↓**	**Comp↑**	**A-S↓**	**A-C↑**	**Suff↓**	**LO↓**	**Comp↑**	**A-S↓**	**A-C↑**
SST2	fAML	**0.006**	**-2.033**	**0.576**	**0.012**	**0.273**	**0.001**	-2.156	**0.638**	**0.022**	0.316	**-0.010**	**-1.994**	0.467	**0.021**	**0.286**
	pAML	0.236	-0.614	0.235	0.097	0.100	0.131	-0.720	0.350	0.101	0.169	0.131	-0.663	0.240	0.095	0.140
	SIG	0.097	-1.340	0.381	0.066	0.205	0.064	-1.920	0.453	0.051	0.242	0.083	-1.133	0.378	0.063	0.208
	ALTI	0.116	-1.043	0.305	0.070	0.166	0.059	-1.407	0.368	0.047	0.205	0.093	-0.816	0.320	0.064	0.175
	DCMP	0.086	-1.366	0.393	0.057	0.216	0.359	-0.321	0.089	0.201	0.063	0.037	-1.450	0.467	0.030	0.253
	SLVX	0.128	-0.401	0.137	0.075	0.097	0.067	**-2.931**	0.595	0.059	**0.322**	0.125	-1.831	**0.517**	0.098	0.282
	LIFT	0.331	-0.300	0.098	0.189	0.060	0.234	-0.605	0.162	0.140	0.098	0.289	-0.314	0.118	0.168	0.071
	GLOB	0.227	-0.489	0.166	0.136	0.100	0.357	-0.262	0.073	0.199	0.060	0.233	-0.388	0.153	0.140	0.104
	SHAP	0.227	-0.788	0.226	0.135	0.131	0.207	-1.264	0.291	0.122	0.167	0.250	-0.534	0.191	0.145	0.111
	GXI	0.359	-0.245	0.077	0.200	0.049	0.305	-0.415	0.111	0.179	0.066	0.237	-0.401	0.153	0.140	0.094
	IG	0.116	-1.249	0.360	0.073	0.200	0.100	-1.823	0.415	0.065	0.229	0.110	-0.936	0.334	0.075	0.187
	LIME	0.050	-1.180	0.356	0.035	0.194	0.150	-1.281	0.298	0.104	0.162	0.134	-0.713	0.273	0.093	0.149
RTN	fAML	**-0.013**	**-1.536**	**0.516**	**0.023**	**0.292**	**-0.017**	**-0.653**	**0.477**	0.032	0.212	**-0.002**	**-2.767**	**0.656**	**0.017**	**0.295**
	pAML	0.126	-0.621	0.271	0.079	0.168	0.052	-0.501	0.296	0.071	0.124	0.181	-1.007	0.328	0.104	0.127
	SIG	0.114	-0.752	0.324	0.088	0.178	0.087	-0.520	0.316	0.065	0.169	0.157	-1.524	0.353	0.109	0.190
	ALTI	0.146	-0.489	0.228	0.103	0.131	0.119	-0.428	0.206	0.084	0.117	0.111	-1.266	0.334	0.079	0.193
	DCMP	0.072	-0.695	0.314	0.062	0.181	0.294	-0.291	0.068	0.169	0.045	0.045	-2.058	0.471	0.042	0.263
	SLVX	0.170	-0.233	0.114	0.112	0.078	-0.003	-0.620	0.466	**0.032**	**0.256**	0.153	-2.598	0.514	0.122	0.292
	LIFT	0.374	-0.142	0.059	0.209	0.040	0.232	-0.322	0.095	0.145	0.060	0.310	-0.523	0.155	0.183	0.092
	GLOB	0.229	-0.277	0.139	0.138	0.088	0.309	-0.273	0.046	0.174	0.035	0.269	-0.589	0.179	0.170	0.112
	SHAP	0.218	-0.418	0.188	0.139	0.111	0.171	-0.431	0.213	0.105	0.123	0.310	-0.744	0.181	0.185	0.111
	GXI	0.384	-0.096	0.051	0.215	0.034	0.290	-0.283	0.060	0.172	0.037	0.265	-0.652	0.173	0.163	0.102
	IG	0.121	-0.700	0.316	0.092	0.177	0.095	-0.523	0.322	0.068	0.173	0.183	-1.117	0.278	0.123	0.169
	LIME	0.117	-0.471	0.242	0.086	0.137	0.099	-0.469	0.269	0.072	0.145	0.186	-1.023	0.261	0.124	0.146
IMDB	fAML	**-0.017**	**-2.914**	**0.787**	**0.018**	**0.422**	**-0.029**	**-2.549**	**0.699**	**-0.004**	**0.443**	**-0.006**	**-4.823**	**0.808**	**0.009**	**0.418**
	pAML	0.057	-1.647	0.518	0.078	0.305	0.016	-1.659	0.526	0.051	0.344	0.029	-3.274	0.611	0.076	0.256
	SIG	-0.002	-1.303	0.383	0.060	0.238	-0.004	-2.041	0.570	0.039	0.356	0.058	-2.590	0.380	0.101	0.251
	ALTI	0.023	-0.604	0.246	0.068	0.156	0.029	-0.682	0.354	0.046	0.214	0.040	-1.252	0.373	0.051	0.234
	DCMP	-0.016	-1.699	0.544	0.032	0.322	0.296	-0.125	0.062	0.268	0.047	-0.005	-3.739	0.706	0.016	0.412
	LIFT	0.212	-0.116	0.054	0.223	0.038	0.100	-0.268	0.110	0.129	0.078	0.182	-0.330	0.085	0.191	0.060
	GLOB	0.071	-0.268	0.128	0.125	0.089	0.239	-0.093	0.040	0.248	0.038	0.160	-0.382	0.140	0.183	0.109
	SHAP	0.137	-0.565	0.189	0.160	0.120	0.132	-0.839	0.277	0.135	0.175	0.191	-0.887	0.167	0.198	0.111
	GXI	0.262	-0.094	0.037	0.252	0.028	0.158	-0.149	0.067	0.173	0.049	0.138	-0.388	0.095	0.171	0.068
	IG	0.037	-1.354	0.438	0.070	0.262	0.029	-1.649	0.550	0.049	0.330	0.083	-1.843	0.335	0.098	0.204
	LIME	0.121	-0.102	0.044	0.165	0.029	0.132	-0.153	0.076	0.168	0.053	0.149	-0.224	0.069	0.198	0.045
EMR	fAML	**0.111**	**-4.832**	**0.738**	**0.097**	**0.411**	**0.005**	**-2.557**	**0.707**	**0.022**	**0.385**	**0.000**	**-3.990**	**0.779**	**0.014**	**0.411**
	pAML	0.198	-3.872	0.595	0.141	0.324	0.069	-1.869	0.572	0.048	0.309	0.084	-2.644	0.626	0.045	0.330
	SIG	0.347	-2.006	0.438	0.214	0.239	0.147	-1.398	0.494	0.091	0.269	0.190	-2.063	0.512	0.116	0.279
	ALTI	0.172	-3.144	0.608	0.129	0.338	0.051	-1.635	0.562	0.036	0.307	0.044	-2.592	0.631	0.030	0.350
	DCMP	0.172	-3.326	0.622	0.129	0.337	0.537	-0.370	0.118	0.278	0.082	0.060	-2.763	0.625	0.044	0.346
	SLVX	0.322	-2.638	0.380	0.199	0.220	0.048	-2.392	0.664	0.037	0.361	0.101	-3.860	0.697	0.087	0.383
	LIFT	0.514	-2.020	0.332	0.282	0.183	0.362	-0.891	0.287	0.198	0.157	0.380	-1.422	0.311	0.207	0.174
	GLOB	0.211	-3.106	0.581	0.146	0.322	0.543	-0.369	0.120	0.278	0.084	0.094	-2.423	0.583	0.061	0.322
	SHAP	0.525	-1.293	0.254	0.288	0.151	0.307	-1.024	0.337	0.169	0.186	0.426	-1.157	0.278	0.228	0.167
	GXI	0.541	-1.510	0.258	0.294	0.147	0.415	-0.754	0.237	0.233	0.127	0.354	-1.472	0.330	0.197	0.187
	IG	0.328	-1.917	0.431	0.205	0.237	0.139	-1.396	0.498	0.086	0.269	0.190	-2.106	0.523	0.113	0.285
	LIME	0.198	-3.814	0.647	0.143	0.351	0.059	-1.911	0.593	0.041	0.320	0.104	-2.897	0.586	0.065	0.322

Table 1: Evaluation results for all combinations of encoder-based model, dataset, explanation method, and metric. See the notations in Sec. 4.

excessive runtime, rendering them impractical for this experiment. Similar trends to Tab. 1 emerge, with fAML consistently outperforming other methods by a significant margin in the majority of cases. pAML typically ranks as the runner-up. We further observe the LLM-based explanation method underperforms the leading explanation methods. We attribute it to the fact that in many cases, the models exhibits hallucinations, where the models generate output tokens that are not present in the original input, thus distorting the ranked list of explanations. Further qualitative results, ablation studies, runtime comparisons, and human evaluation metrics are provided in the Appendix.

		Llama2					Mistral				
		Suff↓	**LO↓**	**Comp↑**	**A-S↓**	**A-C↑**	**Suff↓**	**LO↓**	**Comp↑**	**A-S↓**	**A-C↑**
SST2	fAML	**0.019**	**-0.223**	**0.135**	**0.046**	**0.069**	**0.121**	**-0.325**	**0.228**	**0.086**	**0.120**
	pAML	0.093	-0.137	0.091	0.077	0.053	0.254	-0.155	0.122	0.127	0.091
	SIG	0.157	-0.098	0.065	0.090	0.041	0.281	-0.256	0.176	0.160	0.100
	LIFT	0.159	-0.056	0.033	0.093	0.023	0.319	-0.139	0.100	0.184	0.067
	SHAP	0.152	-0.072	0.048	0.088	0.032	0.308	-0.125	0.090	0.177	0.061
	GXI	0.159	-0.054	0.033	0.092	0.023	0.319	-0.142	0.102	0.184	0.067
	LLM	0.119	-0.101	0.069	0.069	0.042	0.257	-0.115	0.082	0.151	0.062
RTN	fAML	**0.033**	**-0.357**	**0.152**	**0.034**	**0.097**	**0.131**	**-0.432**	**0.275**	**0.120**	**0.143**
	pAML	0.156	-0.168	0.076	0.093	0.052	0.272	-0.225	0.131	0.188	0.087
	SIG	0.219	-0.146	0.088	0.126	0.054	0.312	-0.323	0.215	0.184	0.121
	LIFT	0.211	-0.115	0.068	0.124	0.040	0.375	-0.156	0.103	0.220	0.077
	SHAP	0.216	-0.103	0.059	0.124	0.043	0.365	-0.140	0.097	0.213	0.068
	GXI	0.212	-0.114	0.066	0.125	0.040	0.375	-0.161	0.105	0.220	0.077
	LLM	0.172	-0.152	0.091	0.104	0.056	0.323	-0.129	0.083	0.191	0.065
IMDB	fAML	**0.001**	**-0.518**	**0.203**	**0.078**	**0.125**	**0.026**	**-0.402**	**0.284**	**0.078**	**0.136**
	pAML	0.108	-0.343	0.134	0.170	0.092	0.101	-0.252	0.185	0.137	0.099
	LIFT	0.238	-0.288	0.138	0.202	0.088	0.284	-0.185	0.075	0.233	0.056
	SHAP	0.217	-0.289	0.116	0.191	0.073	0.263	-0.153	0.053	0.221	0.041
	GXI	0.237	-0.288	0.137	0.202	0.088	0.283	-0.186	0.076	0.233	0.056
	LLM	0.179	-0.306	0.081	0.168	0.045	0.250	-0.133	0.048	0.210	0.039
EMR	fAML	**0.218**	**-2.981**	**0.745**	**0.168**	**0.398**	0.263	**-3.740**	0.614	**0.155**	0.332
	pAML	0.284	-2.214	0.608	0.194	0.336	0.545	-1.848	0.334	0.274	0.191
	SIG	0.469	-1.879	0.584	0.272	0.302	0.273	-2.572	0.620	0.174	0.336
	LIFT	0.661	-1.563	0.456	0.370	0.252	0.629	-1.697	0.367	0.344	0.204
	SHAP	0.632	-1.316	0.386	0.344	0.216	0.451	-2.001	0.452	0.255	0.251
	GXI	0.663	-1.566	0.457	0.369	0.252	0.629	-1.683	0.367	0.344	0.205
	LLM	0.701	-0.773	0.204	0.364	0.124	0.561	-1.306	0.303	0.281	0.185

Table 2: Evaluation results on the Llama2 and Mistral models.

5 Conclusion

In this paper, we introduced AML - a self-supervised optimization framework for explaining LMs. By introducing an auxiliary attribution model, AML navigates the complexity of explaining model predictions through a dual-masking approach. Notably, AML presents a flexible loss facilitating the creation of a metric-adapted explanation. Our extensive evaluation, encompassing 13 explanation methods, 4 datasets, 5 LMs, and 5 explanation metrics, consistently demonstrates the superior performance of AML compared to other state-of-the-art methods.

6 Limitations and Future Work

While AML has demonstrated remarkable performance, we acknowledge certain limitations that warrant further investigation. In Sec. 3, we proposed several methodologies for employing masking operations. Although we have shown in this work that employing option 1 in AML yields state-of-the-art results, it would be worthwhile to explore the potential of each of the other options as well. For instance, a recent study by (Zhao and Shan, 2024) suggests that option 3, which replaces the token to be masked with a token predicted based on the surrounding context, may offer optimal performance. Option 3 also has an advantage in cases where models do not support <MASK> or <unk> tokens, as the predicted token always belongs to the vocabulary. However, option 3 also necessitates the use of a designated prediction model. Lastly, this work focuses on assessing explainability via objective evaluation metrics and agreement with human rationales (DeYoung et al., 2020). A promising avenue for future research would be to explore other facets of explainability (Krishna et al., 2022), including stability and fairness.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631.

Arthur Mensch Chris Bamford Devendra Singh Chaplot Diego de las Casas Florian Bressand Gianna Lengyel Guillaume Lample Lucile Saulnier Lélio Renard Lavaud Marie-Anne Lachaux Pierre Stock Teven Le Scao Thibaut Lavril Thomas Wang Timothée Lacroix William El Sayed Albert Q. Jiang, Alexandre Sablayrolles. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. 2023. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629.

Oren Barkan. 2017. Bayesian neural word embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Oren Barkan, Omri Armstrong, Amir Hertz, Avi Caciularu, Ori Katz, Itzik Malkiel, and Noam Koenigstein. 2021a. Gam: Explainable visual similarity and classification via gradient activation maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 68–77.

Oren Barkan, Yuval Asher, Amit Eshel, Yehonatan Elisha, and Noam Koenigstein. 2023a. Learning to explain: A model-agnostic framework for explaining black box models. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 944–949. IEEE.

Oren Barkan, Veronika Bogina, Liya Gurevitch, Yuval Asher, and Noam Koenigstein. 2024. A counterfactual framework for learning and evaluating explanations for recommender systems. In *Proceedings of the ACM on Web Conference 2024*, pages 3723–3733.

Oren Barkan, Avi Caciularu, Ori Katz, and Noam Koenigstein. 2020a. Attentive item2vec: Neural attentive user representations. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3377–3381. IEEE.

Oren Barkan, Avi Caciularu, Idan Rejwan, Ori Katz, Jonathan Weill, Itzik Malkiel, and Noam Koenigstein. 2020b. Cold item recommendations via hierarchical item2vec. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 912–917. IEEE.

Oren Barkan, Avi Caciularu, Idan Rejwan, Ori Katz, Jonathan Weill, Itzik Malkiel, and Noam Koenigstein. 2021b. Representation learning via variational bayesian networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 78–88.

Oren Barkan, Yehonatan Elisha, Yuval Asher, Amit Eshel, and Noam Koenigstein. 2023b. Visual explanations via iterated integrated attributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2073–2084.

Oren Barkan, Yehonatan Elisha, Jonathan Weill, Yuval Asher, Amit Eshel, and Noam Koenigstein. 2023c. Deep integrated explanations. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 57–67.

Oren Barkan, Yehonatan Elisha, Jonathan Weill, Yuval Asher, Amit Eshel, and Noam Koenigstein. 2023d. Stochastic integrated explanations for vision models. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE.

Oren Barkan, Yonatan Fuchs, Avi Caciularu, and Noam Koenigstein. 2020c. Explainable recommendations via attentive multi-persona collaborative filtering. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 468–473.

Oren Barkan, Edan Hauon, Avi Caciularu, Ori Katz, Itzik Malkiel, Omri Armstrong, and Noam Koenigstein. 2021c. Grad-sam: Explaining transformers via gradient self-attention maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2882–2887.

Oren Barkan, Roy Hirsch, Ori Katz, Avi Caciularu, and Noam Koenigstein. 2021d. Anchor-based collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2877–2881.

Oren Barkan, Ori Katz, and Noam Koenigstein. 2020d. Neural attentive multiview machines. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3357–3361. IEEE.

Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Oren Barkan, Noam Koenigstein, Eylon Yogev, and Ori Katz. 2019a. Cb2cf: a neural multiview content-to-collaborative filtering model for completely cold item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 228–236.

Oren Barkan, Noam Razin, Itzik Malkiel, Ori Katz, Avi Caciularu, and Noam Koenigstein. 2020e. Scalable attentive sentence pair modeling via distilled sentence

embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3235–3242.

Oren Barkan, Tal Reiss, Jonathan Weill, Ori Katz, Roy Hirsch, Itzik Malkiel, and Noam Koenigstein. 2023e. Efficient discovery and effective evaluation of visual perceptual similarity: A benchmark and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20007–20018.

Oren Barkan, Idan Rejwan, Avi Caciularu, and Noam Koenigstein. 2020f. Bayesian hierarchical words representation learning. In *"Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics"*, pages 3871–3877.

Oren Barkan, Tom Shaked, Yonatan Fuchs, and Noam Koenigstein. 2023f. Modeling users' heterogeneous taste with diversified attentive user profiles. *User Modeling and User-Adapted Interaction*, pages 1–31.

Oren Barkan, Shlomi Shvartzman, Noy Uzrad, Almog Elharar, Moshe Laufer, and Noam Koenigstein. 2023g. Inversynth ii: Sound matching via self-supervised synthesizer-proxy and inference-time fine-tuning. ISMIR.

Oren Barkan and David Tsiris. 2019. Deep synthesizer parameter estimation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3887–3891. IEEE.

Oren Barkan, David Tsiris, Ori Katz, and Noam Koenigstein. 2019b. Inversynth: Deep estimation of synthesizer parameter configurations from audio signals. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2385–2396.

Shay Ben-Elazar, Gal Lavee, Noam Koenigstein, Oren Barkan, Hilik Berezin, Ulrich Paquet, and Tal Zaccai. 2017. Groove radio: A bayesian hierarchical model for personalized playlist generation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 445–453.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Hila Chefer, Shir Gur, and Lior Wolf. 2021a. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 397–406.

Hila Chefer, Shir Gur, and Lior Wolf. 2021b. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 782–791.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. 2020. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*.

Joseph Enguehard. 2023. Sequential integrated gradients: a simple but effective method for explaining language models. *arXiv preprint arXiv:2305.15853*.

Javier Ferrando, Gerard I Gállego, and Marta R Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. *arXiv preprint arXiv:2203.04212*.

Ruth Fong, Mandela Patrick, and Andrea Vedaldi. 2019. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2950–2958.

Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437.

Keren Gaiger, Oren Barkan, Shir Tsipory-Samuel, and Noam Koenigstein. 2023. Not all memories created equal: Dynamic user representations for collaborative filtering. *IEEE Access*, 11:34746–34763.

Dvir Ginzburg, Itzik Malkiel, Oren Barkan, Avi Caciularu, and Noam Koenigstein. 2021. Self-supervised document similarity ranking via contextualized language models and hierarchical inference. *arXiv preprint arXiv:2106.01186*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

Ori Katz, Oren Barkan, Noam Koenigstein, and Nir Zabari. 2022. Learning to ride a buy-cycle: A hyper-convolutional model for next basket repurchase recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 316–326.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Satyapriya Krishna, Tessa Han, Alex Gu, Javin Pombra, Shahin Jabbari, Steven Wu, and Himabindu Lakkaraju. 2022. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv preprint arXiv:2202.01602*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zhuang Liu, Hanzi Mao, Chaozheng Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Itzik Malkiel, Oren Barkan, Avi Caciularu, Noam Razin, Ori Katz, and Noam Koenigstein. 2020. Recobert: A catalog language model for text-based recommendations. *arXiv preprint arXiv:2009.13292*.

Itzik Malkiel, Dvir Ginzburg, Oren Barkan, Avi Caciularu, Jonathan Weill, and Noam Koenigstein. 2022a. Interpreting bert-based text similarity via activation and saliency maps. In *Proceedings of the ACM Web Conference 2022*, pages 3259–3268.

Itzik Malkiel, Dvir Ginzburg, Oren Barkan, Avi Caciularu, Yoni Weill, and Noam Koenigstein. 2022b. Metricbert: Text representation learning via self-supervised triplet training. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *NAACL-HLT*.

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2023. Decompx: Explaining transformers decisions by propagating token decomposition. *arXiv preprint arXiv:2306.02873*.

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. Globenc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. *arXiv preprint arXiv:2205.03286*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "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*, pages 1135–1144.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2017. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 conference on*

empirical methods in natural language processing, pages 3687–3697.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhixue Zhao and Boxuan Shan. 2024. Reagent: Towards a model-agnostic feature attribution method for generative language models. *ReLM workshop at AAAI24*.

Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. 2018. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*.

Yilun Zhou and Julie Shah. 2023. The solvability of interpretability evaluation metrics. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2399–2415, Dubrovnik, Croatia. Association for Computational Linguistics.

A Appendix

A.1 Agreement with human rationales

In this section, we assess the explanations based on their alignment with human-annotated explanations. This evaluation examines a distinct aspect of explainability, focusing on human rationale rather than model rationale. In other words, it gauges the usefulness of the generated explanation for humans rather than its faithfulness to the model's prediction. While one might argue that faithfulness tests are more critical as they reveal the actual elements in the input that the model relies on for its prediction, for the sake of completeness, we also report results for human agreement tests on the Movie Reviews dataset[8] from the ERASER benchmark[9] (DeYoung et al., 2020), which includes ground-truth annotations provided by human annotators.

In this experiment, we employed a BERT model finetuned on the SST2 dataset. It is important to note that neither pAML nor fAML can be optimized for this type of test, as it is impermissible to expose them to ground-truth annotations. We chose to optimize pAML and fAML based on w.r.t. the A-C metric. This design choice further tests whether optimizing explanations with respect to faithfulness objectives results in explanations that align with human rationales.

Table 3 displays the Macro and Micro F1 scores (DeYoung et al., 2020) computed based on the top 25 tokens, for a selection of top-performing methods from Tabs. 1-2. We observe that both fAML and pAML surpass the other methods, despite being optimized w.r.t. the A-C metric. Therefore, the results in Tab. 3 suggest that the ability of fAML and pAML to optimize for model faithfulness also results in explanations that align with ground-truth annotations, thereby providing satisfactory explanations from a human perspective. Overall, Tab. 3 provides further evidence of the usefulness of the AML methodology in generating explanations that are faithful not only to model rationales, but to human rationales as well.

	Macro F1	Micro F1
fAML	**0.118**	**0.096**
pAML	<u>0.110</u>	<u>0.092</u>
DCMP	0.102	0.088
SIG	0.094	0.082

Table 3: Movie Reviews reasoning task. Macro and Micro F1 scores (for the top 25 tokens) are computed based on the human annotated ground-truth.

A.2 AML ablation study

To assess the individual contributions and necessity of each component in the AML loss (Eq. 4), we conducted an ablation study using the EMR dataset and the DistilBERT model. In this study, we compared fAML (pAML) with two ablated versions: **fAMLXinv** (**pAMLXinv**) and **fAMLXpred** (**pAMLXpred**), which **omit** the $\mathcal{L}_{inv}$ and $\mathcal{L}_p$ loss terms from Eq. 4, respectively.

Table 4 reports results for both Suff and Comp metrics, assessing the quality of explanations from two complementary perspectives. Consistent trends are observed for both the pretrained and finetuned cases, where fAMLXpred and fAMLXinv exhibit inferior performance compared to fAML. This observation underscores the crucial contribution gained from the simultaneous optimization of both $\mathcal{L}_{inv}$ and $\mathcal{L}_p$ in the AML framework.

Notably, we observe fAMLXpred (pAMLXpred) outperforms fAMLXinv (pAMLXinv) on the Comp metric, while the opposite trend emerges on the Suff metric. This can be attributed to the nature of their respective optimization: fAMLXpred (pAMLXpred) combines $\mathcal{L}_{inv}$ with $\mathcal{L}_a$, leading to the production of attribution maps that prioritize comprehensiveness. This is since this combination encourages a reduction in the score assigned to the class that received the highest score in the original prediction, when the input elements are masked proportionally to their assigned attribution scores. On the other hand, fAMLXinv (pAMLXinv) combines $\mathcal{L}_p$ with $\mathcal{L}_a$, aiming to preserve the original prediction while masking the less important elements in the input, thereby promoting sufficiency. Indeed, empirically, we observed that higher values of λ_{inv} and λ_p are selected when optimizing for the Comp and Suff metrics, respectively.

Overall, the complete AML loss (Eq. 4 facilitates an adaptable trade-off among all its components by configuring λ_{inv}, λ_p, and λ_a on the validation set during the pretraining phase for each metric. This approach results in optimal performance for both Suff and Comp metrics.

[8] https://www.eraserbenchmark.com/zipped/movies.tar.gz

[9] https://www.eraserbenchmark.com/

	Suff↓	Comp↑
fAML	**0.01**	**0.71**
pAML	0.07	0.57
fAMLXpred	0.02	**0.71**
pAMLXpred	0.09	0.59
fAMLXinv	**0.01**	0.67
pAMLXinv	0.06	0.55

Table 4: Ablation study results using the DistilBERT model on the EMR dataset.

	SST2	IMDB
fAML	1.5s	1.6s
pAML	**0.01**s	**0.01**s
SIG	6.5s	∼1.4m
DCMP	0.14s	1.6s
SLVX-beamsize-10	1.2s	∼35m
SLVX-beamsize-50	3.0s	∼1h
SLVX-beamsize-100	4.3s	∼1.5h

Table 5: Runtime comparison tests. 's', 'm' and 'h' stand for seconds, minutes, and hours. See Sec. A.3 for details.

A.3 Runtime comparison

In this section, we present the runtime performance of the top-performing methods: fAML, pAML, SIG, and DCMP. We also provide runtime performance for the SLVX method, which allows optimization per metric via the application of beam-search. Table 5 presents the average runtime performance based on two experimental runs using the RoBERTa model, each involving examples drawn from the SST2 and IMDB datasets. For SST2, consisting of short texts with an average length of 14 words, we used 50 examples. For IMDB, comprising longer texts with an average length of 226 words, 15 examples were used. Specifically for SLVX, we reported results across various beam sizes.

We observe the following trends: pAML exhibits significantly better runtime efficiency compared to all other methods. Notably, fAML remains nearly unaffected by text length, demonstrating consistent performance for longer texts. These findings suggest that both fAML and pAML are practical even in real-time scenarios. In contrast, SIG and SLVX exhibit runtimes that become impractical for long texts.

As a final note, it is important to highlight that the attribution model employed in fAML and pAML is not restricted to the same architecture as the model under explanation. This flexibility allows for the utilization of a much lighter attribution model compared to the explained model. For instance, in our experiments, we utilized a lighter RoBERTa model as the attribution model to explain the Llama2 model. In future research, we aim to explore even lighter architectures suitable for the attribution model.

A.4 Qualitative Results

Table 6 presents qualitative comparison of AML and the two runner-ups methods from Tabs. 1 and 2: SIG, DCMP and SLVX. The examples are taken from the SST2 and EMR datasets, and the predictions and attributions are computed using the corresponding finetuned BERT and RoBERTa models. The top three words are highlighted in bold for each example based on token importance in the attribution map produced by each explanation method. As both pAML and fAML yielded the same ranking of attribution scores, we present a single line for both of them for each example. It is evident that AML produces the most meaningful attribution. For more examples, the reader is referred to our GitHub repository[10].

A.5 Evaluation metrics

For quantitative assessment of the explanation methods, we consider the following set of metrics:

1. Log-Odds (**LO**) (Shrikumar et al., 2017) score is defined as the average difference of the negative logarithmic probabilities on the predicted class before and after masking the top $k\%$ words with <MASK> padding (Encoder-based models) and <UNK> padding (Decoder-based models). Lower scores are better. In this work, we used LO with $k = 20$.

2. Comprehensiveness (**Comp**) (DeYoung et al., 2020) score is defined as the average difference of the change in predicted class probability before and after removing the top $k\%$ features. Similar to Log-odds, this measures the influence of the top-attributed tokens on the model's prediction. For a single example

[10] https://github.com/amlconf/aml/blob/main/qualitative_examples.md

Dataset	Prediction	Model	Method	Attribution
SST2	Positive	BERT	AML	a modest **pleasure** that **accomplishes** its goals with ease and **confidence** .
			SIG	**a** modest pleasure **that** accomplishes its goals with ease and **confidence** .
			DCMP	a modest **pleasure that** accomplishes its goals with ease and **confidence** .
			SLVX	a modest **pleasure** that **accomplishes** its goals with **ease** and confidence .
SST2	Positive	BERT	AML	finds a way to tell a simple story , perhaps the **simplest** story of all , in a way that seems **compelling** and even **original** .
			SIG	**finds a** way to tell a simple story , perhaps the simplest story of all , in a way that seems **compelling** and even original .
			DCMP	finds a way to tell a simple **story** , perhaps the simplest story of all , in a way that seems **compelling and** even original .
			SLVX	finds a way to tell a simple story , perhaps the simplest story of all , **in a** way that seems **compelling** and even original .
SST2	Negative	BERT	AML	a **grim** , **flat** and **boring** werewolf movie that refuses to develop an energy level .
			SIG	a grim , flat and **boring** werewolf **movie** that **refuses** to develop an energy level .
			DCMP	a grim , flat and **boring** werewolf **movie** that refuses to develop an **energy** level .
			SLVX	a grim , flat and **boring** werewolf movie that **refuses** to **develop** an energy level .
SST2	Negative	BERT	AML	**there 's** something **fundamental missing** from this story : something or someone to care about .
			SIG	there 's something fundamental **missing** from this story : **something or** someone to care about .
			DCMP	**there 's** something fundamental **missing** from this story : something **or** someone to care about .
			SLVX	there 's something fundamental **missing** from **this story** : something or someone to care about .
EMR	Joy	RoBERTa	AML	hawke **draws** out the best from his large cast in **beautifully** articulated portrayals that are subtle and so **expressive** they can **sustain** the poetic flights in burdette 's dialogue .
			SIG	hawke draws out the best from his large cast in **beautifully** articulated portrayals that are subtle and so **expressive** they can sustain the **poetic** flights in burdette 's dialogue .
			DCMP	hawke draws out the best from his large cast in **beautifully** articulated portrayals that are subtle and so **expressive** they can sustain the **poetic** flights in burdette 's dialogue .
			SLVX	**hawke** draws out the best from his large cast in **beautifully** articulated portrayals that are subtle **and** so expressive they can sustain the poetic flights in burdette 's dialogue .
EMR	Sadness	RoBERTa	AML	i was **feeling** really **troubled** and **down** over what my dad said
			SIG	**i** was **feeling** really **troubled** and down over what my dad said
			DCMP	i was **feeling** really **troubled** and down over **what** my dad said
			SLVX	i was **feeling** really **troubled** and down over what my **dad** said

Table 6: Examples of RoBERTa attributions on several sentences from the SST2 and EMR dataset. The bold tokens represent the top 3 tokens in the sentence, according to each attribution method. See Sec. A.4 for details.

Comp is computed as:

$$p(y'|x_i) - p(y'|x_i^{(k)}),$$

where y' is the predicted class, x is the input sequence of tokens, and $x^{(k)}$ denotes the modified sequence with the top $k\%$ attributed tokens deleted from the sequence. Higher scores are better. In this work, we used Comp with $k = 20$.

3. Sufficiency (**Suff**) (DeYoung et al., 2020) score is defined as the average difference of the change in predicted class probability before and after keeping only the top $k\%$ tokens. This measures the adequacy of the top $k\%$ attributions for model's prediction. It is defined in a similar fashion as comprehensiveness, except the $x^{(k)}$ is defined as the sequence containing only the top $k\%$ tokens. Lower scores are better. In this work, we used Suff with $k = 20$.

4. Area Over the Perturbation Curves (AOPC): AOPC-Sufficiency (**A-S**) and AOPC-Comprehensiveness (**A-C**) (DeYoung et al., 2020) - are the average differences of the change in predicted class probability before and after keeping and removing the top $k\%$ tokens for Sufficiency and Comprehensiveness, respectively:

$$\text{AOPC-S} = \frac{1}{|B|} \sum_{k \in B} \text{Suff}(k),$$

$$\text{AOPC-C} = \frac{1}{|B|} \sum_{k \in B} \text{Comp}(k).$$

Here, we evaluate Comp and Suff for 5 different values of k, setting $B = \{1, 5, 10, 20, 50\}$ as suggested by (DeYoung et al., 2020). A-S and A-C measure how well a specific token ordering is scored under a model from two complementary perspectives and across the k axis.