Accelerating the Global Aggregation of Local Explanations

Alon Mor, Yonatan Belinkov, Benny Kimelfeld

Technion – Israel Institute of Technology, Haifa, Israel
almr16@campus.technion.ac.il, belinkov@technion.ac.il, bennyk@cs.technion.ac.il

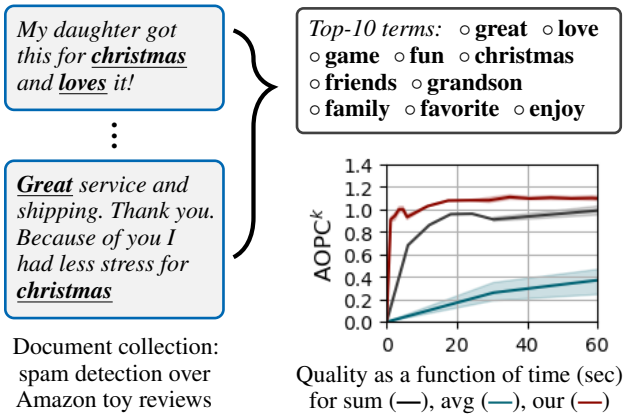
Abstract

Local explanation methods highlight the input tokens that have a considerable impact on the outcome of classifying the document at hand. For example, the Anchor algorithm applies a statistical analysis of the sensitivity of the classifier to changes in the tokens. Aggregating local explanations over a dataset provides a global explanation of the model. Such aggregation aims to detect words with the most impact, giving valuable insights about the model, like what it has learned in training and which adversarial examples expose its weaknesses. However, standard aggregation methods bear a high computational cost: a naïve implementation applies a costly algorithm to each token of each document, and hence, it is infeasible for a simple user running in the scope of a short analysis session.

We devise techniques for accelerating the global aggregation of the Anchor algorithm. Specifically, our goal is to compute a set of top-k words with the highest global impact according to different aggregation functions. Some of our techniques are lossless and some are lossy. We show that for a very mild loss of quality, we are able to accelerate the computation by up to 30×, reducing the computation from hours to minutes. We also devise and study a probabilistic model that accounts for noise in the Anchor algorithm and diminishes the bias toward words that are frequent yet low in impact.

Introduction

A particular paradigm for local explanations consists of algorithms that compute a numeric estimate of each token's contribution to the decision of the model, also known as input attribution (Danilevsky et al. 2020a). Many of the common techniques are computationally intensive. For instance, the score that Anchor (Ribeiro, Singh, and Guestrin 2018) assigns to a token is the probability that the model keeps its decision intact when the document undergoes random perturbations, as long as the token remains unchanged. LIME (Ribeiro, Singh, and Guestrin 2016) derives a linear bag-of-words predictor of the model's outcome in a small area around the document and scores each token by its learned coefficient. The SHAP score (Lundberg and Lee 2017) views the tokens as players in the cooperative game of forming the model's decision, and applies the well-known Shapley value to attribute a portion of the profit to each player. As a running illustration, Figure 1 shows documents from a spam-detection task of Amazon reviews on toys and games, with words marked as *anchors*, meaning that their score by Anchor exceeds a threshold.



Document collection: spam detection over Amazon toy reviews

Quality as a function of time (sec) for sum (—), avg (—), our (—)

Figure 1: Left: local explanations in a collection of documents for a spam detection task. Top-right: top-10 terms resulting from the global aggregation of (local) anchors. Bottom-right: global aggregation quality; our approach leads to higher quality at a fraction of the computation time.

In turn, an approach to global explanation is the *aggregation* of the attribution scores computed by methods as the aforementioned, intended to quantify the impact of each term on the model (Arras et al. 2016). More precisely, we begin with a set of instances to classify (e.g., the training set or the test set of a learning setup), compute a score for each token, and then estimate the impact of every word by aggregating the scores that it obtained in its occurrence in documents. Figure 1 shows top-10 impactful words based on the anchors; to illustrate an insight, "Christmas" is one of the most impactful terms that encourage the model (Bert, Devlin et al. 2018) to classify a document as spam. Like any form of a global explanation, we would like it to be used effectively by users of varying skills to gain valuable insights on the model: what it learned in training, which biases it has towards specific phrases, which adversarial examples can shed light on possible weaknesses, and so on. For instantiation, we would like to present the top-k impactful terms in online analysis frameworks such as the FIND system (Lertvittayakumjorn, Specia, and Toni 2020) that provides attribution-based insights.

Yet, the global aggregation of a local attribution, like the above ones, entails critical challenges since the latter is designed to explain a single decision by the model. For one, how exactly should we aggregate the scores? The local scores are inherently noisy, and so, if we simply sum up the scores of all occurrences, we end up giving an advantage to frequent words that are irrelevant to the model (e.g., a function word like "the" is likely to receive enough noisy scores that sum up to a high global score). If we normalize by the number of occurrences, then we promote rare terms that matter only in insignificant cases (e.g., a word that only appeared once in the dataset would be scored highly). Either way, the aggregation falls short of the expectation to explain the model.

Another major challenge, regardless of which aggregation is used, is the *computational cost*. The statistical approaches are arguably designed to be practical as local explanations, where we wish to provide an online answer for a single instance; yet, global aggregation should, in principle, apply the costly computation to every token of every document. For instance, a direct application of the release of Anchor on a collection of 10k documents took us almost two days, which were shortened to more than an hour after code optimizations. Again, such performance casts the usage of global aggregation as ineffective for online analysis pipelines.

We focus on the global aggregation of the Anchor explanations and tackle the above challenges by making the following contributions. First, we design a probabilistic model that targets the importance of words *as explanations* and, at the same time, accounts for noise and irrelevant occurrences. Second, we develop several optimizations of global aggregation over Anchor towards an effective implementation. Some of the optimizations are specific to Anchor, while others apply to every global aggregation of contribution scores. Specifically, we design an *anytime* algorithm that maintains a set of top-k candidates, and approaches the top score after a fraction of the naïve time, with an improvement of up to 30x. Third, we design and conduct an experimental study that shows the effectiveness of our solutions. For that, we adapt the evaluation method of Guidotti et al. (2018) to the task of top-k impactful terms. This is illustrated in Figure 1: after a few seconds, our anytime algorithm produced a set of words that outperforms mere sum and average (that terminate in ten minutes).

Related work. The local attribution scores we discussed fall in the category of *perturbation-based* explanations (Anchor) or *simplification-based* explanations (LIME, SHAP). An earlier perturbation-based explanation is occlusion (Zeiler and Fergus 2014). Another common category is *gradient-based* explanations, such as Saliency (Simonyan, Vedaldi, and Zisserman 2014), InputXGradient (Shrikumar et al. 2016), and others (Pruthi et al. 2020; Sundararajan, Taly, and Yan 2017); these assume white-box access to the model, while perturbation-based methods can operate with any model in a black-box manner. See Atanasova et al. (2020) for a comparative analysis. Other scores are derived from an *attention mechanism* incorporated in the model (Lu et al. 2019; Xie et al. 2017). Layer-wise Relevance Propagation (LRP) extracts from the network a linear model where, similarly to LIME, the coefficients are used as scores (Arras et al. 2016).

Aggregation of local attributions was studied by Ebert, Jakobovits, and Filippova (2022), who applied aggregation to the gradient-based local attribution of Bastings et al. (2022). Aggregation of LRP was done by Lertvittayakumjorn, Specia, and Toni (2020) and Gholizadeh and Zhou (2021). These publications focused on representing the whole space of local attributions (e.g., via clustering, word clouds, and representative embeddings), and did not focus on the challenge of execution cost. Later in the paper, we refer to work on the aggregation of LIME scores (Ribeiro, Singh, and Guestrin 2016; van der Linden, Haned, and Kanoulas 2019) that also did not focus on the computational aspects.

Token scoring is one of a plethora of explanation forms proposed for machine-learning models. Other popular approaches derive from the original model different kinds of insights such as nearby *interpretable* (or *surrogate*) models, deterministic *rules*, and *examples* that highlight nuances of the model. These can be found in relevant surveys such as Guidotti et al. (2019), Danilevsky et al. (2020b) and Atanasova et al. (2020), to name a few.

Formal Setup

A *document* d is a finite sequence $w_1, \ldots, w_m$ of *words*. For $i = 1, \ldots, m$, we call the pair (w_i, i) a *token* of d. By a slight abuse of notation, we may identify d with its set of tokens; hence, $(w, i) \in d$ means that the i'th word of d is w. We denote by $\mathbf{D}$ the set of all documents. By a *predictor* we refer to any function $f : \mathbf{D} \to C$ that maps documents to some domain C. In particular, a binary classifier is a predictor where $C = \{0, 1\}$. In our evaluation, we will make the (conventional) assumption that $f(d)$ is the most likely class according to an associated probability function $\hat{f} : \mathbf{D} \times C \to [0, 1]$ that defines a distribution over C for each document d.

Anchors. The anchor concept has been defined for general modalities and "predicates" (properties of the input instance) that can take the role of an explanation of a prediction (Ribeiro, Singh, and Guestrin 2018). We define it in the textual domain where the predicates are token memberships.

We assume that every document d is associated with a *perturbator*, which is a distribution Δ_d over $\mathbf{D}$. We later discuss the actual perturbation used in the public anchor implementation. We also assume a numerical threshold $\tau \in [0, 1]$. Let f be a predictor, and let $(w, i) \in d$ be a token of d. We say that (w, i) is an *anchor* (of d w.r.t. f) if

$$\Pr_{x \sim \Delta_d}[f(x) = f(d) \mid (w, i) \in x] \geq \tau \,. \tag{1}$$

In words, (w, i) is an anchor if the document retains the same prediction under perturbation, with high probability, as long as the word w is kept at the ith position in the perturbation. This probability is referred to as the *precision* of (w, i). We denote by $\mathsf{Anc}(d)$ the set of anchors of a document d.

The algorithm uses a Masked Language Model (MLM) as the perturbator of d. Samples from $\mathbf{D}$ are created by masking tokens of d, and later unmasking using DistilBert (Sanh et al. 2019). The output of the MLM on a masked (w, i) is a distribution of words that can fit the mask. Out of the ζ (500 here) words with the highest probability, a word is sampled according to its probability. If there are multiple masks, then the process repeats iteratively until all replacements are applied.

Global aggregation. Let S be a finite collection of documents, and let $f : \mathbf{D} \to C$ be a predictor that we wish to explore in the context of S. (For instance, S can be the training set that is used for the construction of f.) We denote by $W(S)$ the set of words that occur in S. By *global aggregation* we refer to any numerical function $\mathcal{G}$ that maps every word $w \in W(S)$ and $c \in C$ to a number $\mathcal{G}(w, c)$. Intuitively, high $\mathcal{G}(w, c)$ means that w has high impact on f taking the value of c on a given document. We denote by $S[c]$ the collection of documents in S classified as c. We also denote by $A^+(w, c)$ and $A^-(w, c)$ the number of occurrences of a word $w \in W(S)$ where w is considered an anchor for c and a non-anchor for c, respectively:

$$A^+(w, c) \stackrel{\text{def}}{=} \sum_{d \in S[c]} |\{i \mid (w, i) \in \mathsf{Anc}(d)\}| \quad (2)$$

$$A^-(w, c) \stackrel{\text{def}}{=} \sum_{d \in S[c]} |\{i \mid (w, i) \in d \setminus \mathsf{Anc}(d)\}| \quad (3)$$

Ribeiro, Singh, and Guestrin (2016) proposed the function:

$$\mathcal{G}_{\mathsf{sq}}(w, c) \stackrel{\text{def}}{=} \sqrt{A^+(w, c)} \quad (4)$$

Several aggregations have been proposed by van der Linden, Haned, and Kanoulas (2019). The one that performed best on binary classification is the *Global Average Importance* that, in our context, is normalizing $A^+(w, c)$ by the number occurrences of w:

$$\mathcal{G}_{\mathsf{av}}(w, c) \stackrel{\text{def}}{=} \frac{A^+(w, c)}{A^+(w, c) + A^-(w, c)} \quad (5)$$

Note that $\mathcal{G}_{\mathsf{av}}(w, c)$ does not distinguish between rare words that occur as anchors and words that are frequently anchors. In fact, the maximal score is obtained by a word that occurs once, and in that occurrence, it is an anchor. On the other hand, $\mathcal{G}_{\mathsf{sq}}(w, c)$ is sensitive to the noise of the anchor algorithm since it rewards frequent words (e.g., stop words) that are occasionally identified as anchors.

Another aggregation that van der Linden, Haned, and Kanoulas (2019) proposed is $\mathcal{G}_{\mathsf{h}}(w, c)$, which weighs a word by its score for different classes. The idea is that the multiplicity of classes entails a penalty since it indicates low relevance to the specific class. Let $\widetilde{h}(w, c)$ be the normalized $\mathcal{G}_{\mathsf{sq}}(w, c)$, that is, $\widetilde{h}(w, c) \stackrel{\text{def}}{=} \mathcal{G}_{\mathsf{sq}}(w, c) / \sum_{c' \in C} \mathcal{G}_{\mathsf{sq}}(w, c')$. Note that $\widetilde{h}(w, c)$ can be viewed as a distribution of w's importance across the classes. The Shannon entropy of this distribution is $H_w \stackrel{\text{def}}{=} -\sum_{c \in C} \widetilde{h}(w, c) \log\left(\widetilde{h}(w, c)\right)$. Low H_w implies that w impacts a specific class. Let $H_{\min}$ and $H_{\max}$ be the minimum and maximum $H_{w'}$ across all words w'. Since rare words might occur in a specific class (thus having low entropy), $\mathcal{G}_{\mathsf{h}}$ is defined using H_w and $\mathcal{G}_{\mathsf{sq}}$ as follows:

$$\mathcal{G}_{\mathsf{h}}(w, c) \stackrel{\text{def}}{=} \left(1 - \frac{H_w - H_{\min}}{H_{\max} - H_{\min}}\right) \mathcal{G}_{\mathsf{sq}}(w, c) \quad (6)$$

Hence, a high entropy will have a low factor over $\mathcal{G}_{\mathsf{sq}}$, and rare words will have low $\mathcal{G}_{\mathsf{sq}}$, both resulting in low $\mathcal{G}_{\mathsf{h}}$.

While $\mathcal{G}_{\mathsf{h}}(w, c)$ aims to address the problems of $\mathcal{G}_{\mathsf{av}}(w, c)$ and $\mathcal{G}_{\mathsf{sq}}$, we have found that anchors commonly appear in at most one class, thus the entropy is very low for most words to begin with. In the next section, we will propose a new aggregation that aims to overcome these weaknesses.

Problem definition: top-k terms. We consider the following computational task. We have a set S of documents, a predictor f, an aggregation function $\mathcal{G}$ for anchors, and a number k. The goal is to find, for a given class c of f, a set T_c of k words in $W(S)$ with maximal $\mathcal{G}(w, c)$. We will refer to these words as the *top terms*. While we wish to find T_c in interactive time, a naive aggregation can be prohibitively slow. We might be willing to settle for an approximation, that is, a set T'_c of k words has *similar quality* compared to T_c. Next, we discuss how this quality can be measured.

For evaluating global aggregations, van der Linden, Haned, and Kanoulas (2019) proposed $\mathsf{AOPC}_{\mathsf{global}}$ that adapts the *Area Over the Perturbation Curve* of Samek et al. (2015). The idea is to measure how removal of high-score terms impacts the model's predictions. In our terms, $\mathsf{AOPC}_{\mathsf{global}}$ is calculated on an aggregation $\mathcal{G}$ by progressively removing from each document the top-k terms w in that document by decreasing $\mathcal{G}(w, c)$. A curve shows a better performance than another if it is higher and its initial slope is steeper (indicating better identification of the top influencing terms).

Our goal is to measure the quality of the set T_c of top-k terms. Hence, in our experiments, we adapt $\mathsf{AOPC}_{\mathsf{global}}$ so that we remove from each document d *only the terms in* T_c; in particular, documents that do not intersect with T_c remain unchanged (in contrast to $\mathsf{AOPC}_{\mathsf{global}}$ that removes k terms from each document). Formally, let $f : \mathbf{D} \to C$ be a predictor, $c \in C$ a class, and T_c the set of top-k terms found by the aggregation $\mathcal{G}$ and a class c. Let $w_1, \ldots, w_k$ be the words in T_c ordered by decreasing $\mathcal{G}(\cdot, c)$. For a document d, let d^i be the document d with every occurrence of a word from $w_1, \ldots, w_k$ removed. We define

$$\mathsf{AOPC}^k(\mathcal{G}, c) \stackrel{\text{def}}{=} \frac{1}{k+1} \cdot \operatorname*{avg}_{d \in S[c]} \Big(\sum_{i=1}^{k} \hat{f}(d, c) - \hat{f}(d^i, c) \Big) \quad (7)$$

where $\hat{f}$ is the probability that the classification model of f assigns to c for the document d. In words, $\mathsf{AOPC}^k(\mathcal{G}, c)$ is the average drop in the probability of the class c, over all documents and prefixes of T_c.

Probability-Based Global Aggregation

Consider a document collection S, a predictor $f : \mathbf{D} \to C$, and a class $c \in C$. Let $d = (w_1, \ldots, w_\ell)$ be a document in $S[c]$. We consider a simple generative model that produces a random document $X_d = (w'_1, \ldots, w'_\ell)$ of the same length as d, assuming that each word is generated randomly, yet anchor words and non-anchor words are taken from different distributions. More precisely, each w'_i is selected randomly and independently using the following process:

- If $(w_i, i) \in \mathsf{Anc}(d)$, then do as follows. With probability α, select a word $w \in W(S)$ with the *anchor probability* $q(w, c)$; with probability $1 - \alpha$ (that the anchor is wrong), select a word w from $W(S)$ with the probability $p(w, c)$.
- Otherwise, select $w \in W(S)$ with probability $p(w, c)$.

Here, α is a parameter. The probabilities $p(w,c)$ and $q(w,c)$ are unknown and chosen to maximize the probability of S:

$$(p^*, q^*) \stackrel{\text{def}}{=} \operatorname*{argmax}_{p,q} \Big(\prod_{d \in S[c]} \Pr[d = X_d] \Big) \tag{8}$$

Intuitively, words w with high $q^*(w,c)$ are likely to be used as anchors but not necessarily as non-anchors. Hence, we use q^* as our global aggregation:

$$\mathcal{G}_{\mathsf{pr}}(w,c) \stackrel{\text{def}}{=} q^*(w,c) \tag{9}$$

We estimate q^* and p^* via Maximum Likelihood Estimation using Lagrange multipliers (Sargent 2000) for local maxima. See the long version of this paper (Mor, Belinkov, and Kimelfeld 2023). The resulting estimations $\tilde{q}$ and $\tilde{p}$ are as follows. Recall that α is a parameter, and recall $A^+(w,c)$ and $A^-(w',c)$ from Equations (2) and (3), respectively.

$$\tilde{q}(w,c) = \frac{1}{\alpha} \cdot \frac{A^+(w,c)}{\Sigma_{w'} A^+(w',c)} - (\frac{1}{\alpha} - 1) \cdot \frac{A^-(w,c)}{\Sigma_{w'} A^-(w',c)}$$
$$\tilde{p}(w,c) = \frac{A^-(w,c)}{\Sigma_{w'} A^-(w',c)} \tag{10}$$

The resulting $\tilde{q}$ is not necessarily equal to q^* since it is not necessarily positive. To get a probability (which may be different from q^*), we apply *Laplace smoothing* (Chen and Goodman 1999) with the absolute minimum value of $\tilde{q}$ as the smoothing parameter. Let $q_{\mathsf{min}} \stackrel{\text{def}}{=} \min_{w \in W(S[c])} \tilde{q}(w,c)$ and $\mathcal{L}(g,\beta)$ be the Laplace smoothing over the function g with a smoothing parameter β. We then define $\tilde{q}^*(w,c) = \mathcal{L}(\tilde{q}, |q_{\mathsf{min}}|)$ and $\tilde{p}^*(w,c) = \tilde{p}(w,c)$. Laplace smoothing is monotone, thus it preserves word ordering: $\tilde{q}^*(w_1,c) \geq \tilde{q}^*(w_2,c)$ whenever $\tilde{q}(w_1,c) \geq \tilde{q}(w_2,c)$.

Runtime Optimizations

In this section, we propose algorithms and optimization techniques for accelerating the computation of the top-k terms.

Incremental Evaluation (Anytime)

Our goal is to retrieve the top-k words in the dataset. Thus, we adopt an incremental evaluation that maintains the best top-k found in each step of the algorithm. This allows us to provide the user with informative early results in a *pay-as-you-go* (*anytime*) manner. Yet, some of the scores are not known during the computation, since they may require the processing of the entire dataset. Hence, we use a heuristic *pseudo-score* $\tilde{\mathcal{G}}(w,c)$ instead of each exact aggregation $\mathcal{G}(w,c)$.

We traverse the documents in a predefined order (that we discuss next). Denote the word with the lowest pseudo-score in the top-k group as w_{min}. A word w enters the top-k group if $\tilde{\mathcal{G}}(w,c) > \tilde{\mathcal{G}}(w_{\mathsf{min}},c)$. Since the results of the anchor algorithm are unknown for all documents during its running, the pseudo-score when reaching document d_i is calculated with respect to only the first i documents. Specifically, denote by $A_i^+(w,c)$ and $A_i^-(w,c)$ the values $A^+(w,c)$ and $A^-(w,c)$, respectively, restricted to the first i documents in $S[c]$. For $\mathcal{G}_{\mathsf{sq}}, \mathcal{G}_{\mathsf{av}}$ and $\mathcal{G}_{\mathsf{h}}$, the pseudo-score is calculated simply by replacing A with A_i in the definitions. In the case of $\mathcal{G}_{\mathsf{pr}}$, we use the estimates $\tilde{q}$ and $\tilde{p}$ of Equation (10) and, again, apply the replacement of A with A_i. As for the order, we traversed the documents by descending confidence of the classification, assuming that these encourage anchors early on. (We also tried other orders and got similar results.)

Candidate filtering. By utilizing the maintained set of top candidates, we can avoid the expensive computation for an unlikely candidate, by applying an optimistic, fast-to-compute upper bound on its score: a word is filtered out if its upper bound is lower than the minimal pseudo-score in the current set of candidates. The upper bound for a word w considers the occurrences (w,i) of the word in the remaining documents d, and it is the pseudo-score under the assumption that $(w,i) \in \mathsf{Anc}(d)$ for every such d and i. While this bound might appear optimistic, this optimization accounts for 30%-40% reduction of the computation's running time.

Accelerating Anchor

Recall that a major challenge in computing the top terms is the expensive computation of anchors. Yet, we are interested in an aggregation over many applications of the Anchor algorithm, and so, we might settle for lower accuracy and confidence. Hence, we change the hyperparameters of the Anchor algorithm, as follows.

Masking. Recall that Anchor generates documents by masking words in the instance and filling them using a DistilBert MLM model. Each mask is filled out of 500 MLM suggestions. We observed that *decreasing* this number (to just 50 in our experiments) reduced the execution cost without harming the quality, and sometimes even improving it.

Anchor precision and confidence. The precision threshold τ of Equation (1) determines the number of documents that we sample using the replacement alternatives produced by the masking. During runtime, we reduce τ for a word to reflect its past occurrences as an anchor. Consequently, we generate fewer documents to decide whether the word is an anchor. Instead of τ, we use

$$\tilde{\tau}(w,c) = \tau - \omega \cdot \tilde{\mathcal{G}}_{\mathsf{pr}}(w,c)/N_w \tag{11}$$

where N_w is the number of occurrences of w in the dataset, and ω is a hyperparameter. We use $\omega = 0.4$ in our experiments so that $\tilde{\tau}(w,c)$ does cannot get below 0.55.

In addition, the anchor implementation by Ribeiro, Singh, and Guestrin (2018) estimates, for each anchor candidate w, a *confidence score* that the inequality of Equation (1) is correct. To be an anchor, this confidence needs to exceed $1-\delta$. Higher values of δ lead to a reduction in the number of needed samples. Hence, since we aggregate results, we can settle for higher values of δ.

Additional Optimizations

In addition to the optimizations described in this section, we applied several accelerations that are fairly straightforward and standard. First, we filtered out *stop-words* from the set of top-term candidates, as we perceive them as non-informative terms regarding the explanation of the model. We also filtered out *rare words*, where we defined a word to be rare if it occurs

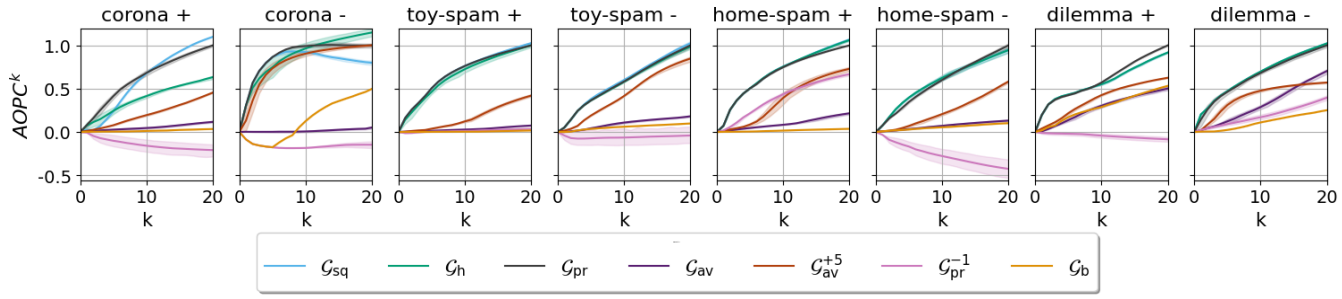


Figure 2: $\mathrm{AOPC}^k(\mathcal{G}, c)$ of different aggregation functions $\mathcal{G}$ (y-axis) for varying k (x-axis).

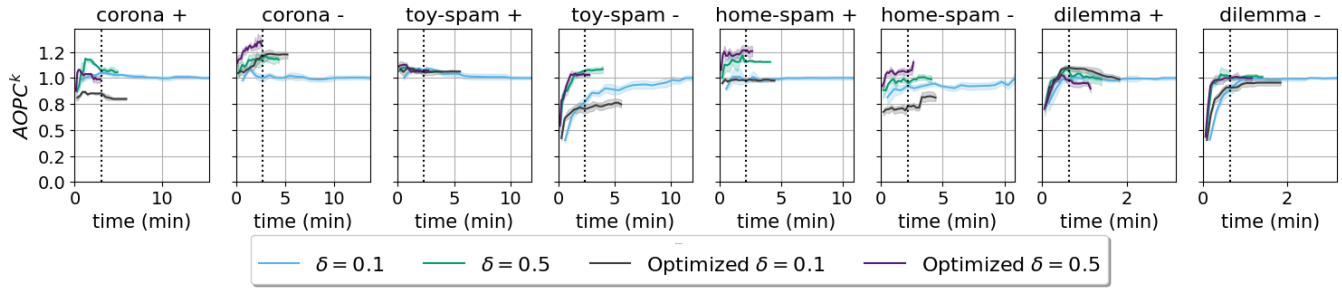


Figure 3: $\mathrm{AOPC}^k(\mathcal{G}, c)$ for different versions $\mathcal{G}$ of $\mathcal{G}_{\mathsf{pr}}$ and $k = 20$, as a function of the computation time. "Optimized" refers to the combination of all runtime optimizations described in the paper.

fewer times than some threshold (5 in our experiments). Our experiments confirm that this filtering provides considerable acceleration for negligible loss of quality. Finally, we also experiment with the application of the entire algorithm on a *sample* of the documents rather than the entire training set.

Experiments and Results

In our experimental study, we aim to understand the effectiveness of the aggregations and optimizations that we proposed. More precisely, we investigate, empirically, how well and how fast each alternative finds top-k terms. The code and data are available at https://github.com/alonm16/anchor.

Setup

We find the top-k terms for several classification tasks. We use $k = 20$. In each task, the documents are organized into three collections: training, validation, and test. The predictor f is a classifier trained on the training set, chosen as the model checkpoint with the best validation set accuracy. We experimented with three models: logistic regression, Bert (Devlin et al. 2018), and DeBERTa (He et al. 2020). We report the results only for DeBERTa; the other two behave similarly and are discussed in the long version (Mor, Belinkov, and Kimelfeld 2023). The set S of documents that we aggregate over is the test set, as in the experimental study of van der Linden, Haned, and Kanoulas (2019).

All experiments were conducted on a machine with 96 of Intel Xeon Gold 6336Y 2.40GHz CPUs with 24 cores, 512GB RAM, 8 of 50GB Nvidia A40 GPUs running Ubuntu 20.04 LTS. The algorithms were programmed in Python 3.10 with the libraries CUDA 11.6, PyTorch 2.0, and Numpy 1.23.

Classification tasks. We restricted each dataset to documents of at most 200 characters, since the influence of each token on a longer text diminishes, and we got fewer anchors and less meaningful results.[1] We used the following tasks from Kaggle. URL references to the tasks can be found in the long version (Mor, Belinkov, and Kimelfeld 2023).

Coronavirus tweets (sentiment). Tweets classified into five sentiments: extremely negative, negative, neutral, positive, and extremely positive. We combined extremely negative and negative, and extremely positive and positive since there were too few anchors that distinguish between the extreme and normal classes. The dataset consists of 16,000, 4200, and 12,500 documents (training, validation, test).

The Social Dilemma tweets (sentiment). Tweets classified into three sentiments: positive, negative, and neutral. The dataset consists of 3200, 1000, and 3000 documents.

Amazon reviews: Toys & Games (spam). Amazon product reviews in the Toy and Games category. The reviews are classified into *spam* and *non-spam*. The dataset consists of 15,000, 3800, and 11,000 documents.

Amazon reviews: Home & Kitchen (spam). Similar to the previous one, but now in the category Home and Kitchen. The dataset consists of 12,000, 3000, and 9000 documents.

[1] The length bound also influences the execution cost; our experiments show that the runtime grows linearly with this bound.

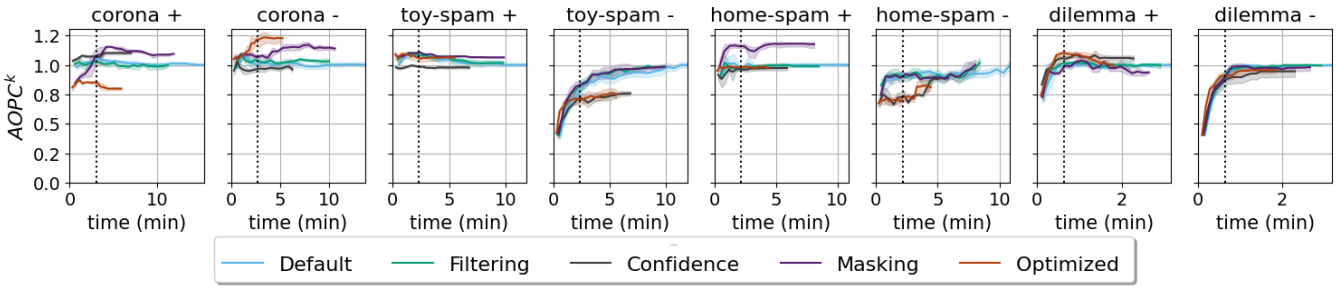


Figure 4: $\mathsf{AOPC}^k(\mathcal{G}, c)$ for different optimizations of $\mathcal{G}_{\mathsf{pr}}$ and $k = 20$, as a function of the computation time. "Optimized" refers to the combination of all optimizations.

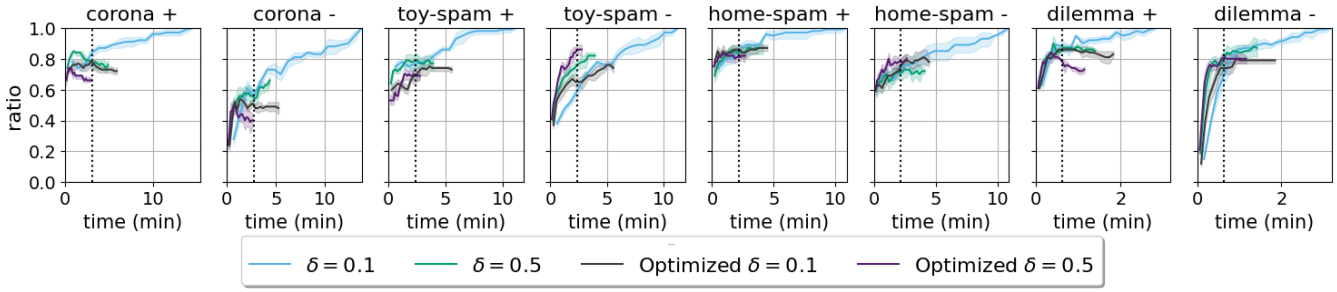


Figure 5: Ratio of shared terms for different versions of $\mathcal{G}_{\mathsf{pr}}$.

Compared Aggregations. We compare the aggregation functions proposed in the paper, as well as their optimized versions. The list of functions includes $\mathcal{G}_{\mathsf{sq}}$ (Equation (4)), $\mathcal{G}_{\mathsf{av}}$ (Equation (5)), $\mathcal{G}_{\mathsf{h}}$ (Equation (6)), and $\mathcal{G}_{\mathsf{pr}}$ (Equation (9)) with $\alpha = 0.5$. As for $\mathcal{G}_{\mathsf{av}}$, we also experiment with a *quick fix* of its weakness of promoting rare words: the function $\mathcal{G}^{+5}_{\mathsf{ave}}$ is the same as $\mathcal{G}_{\mathsf{av}}$, except that every word is filtered out if it has fewer than five occurrences in the training set.

As baselines, we use two aggregations $\mathcal{G}(w, c)$. The first ignores the Anchor algorithm and simply measures the probability of class c among the documents that include w.

$$\mathcal{G}_{\mathsf{base}}(w, c) \stackrel{\text{def}}{=} \frac{|\{d \in S[c] \mid w \in d\}|}{|\{d \in S \mid w \in d\}|} \tag{12}$$

(Formally, $w \in d$ means $(w, i) \in d$ for some i.) The second is $\mathcal{G}^{-1}_{\mathsf{pr}} \stackrel{\text{def}}{=} 1/\mathcal{G}_{\mathsf{pr}}$, the inverse of $\mathcal{G}_{\mathsf{pr}}$, used as a sanity check.

Results

Figure 2 shows the results for the aggregation functions. Each box corresponds to one task and one class c, and it shows $\mathsf{AOPC}^k(\mathcal{G}, c)$ for the top terms at different times. Each line includes a shaded error band based on repetitions with 5 seeds. Observe that $\mathcal{G}_{\mathsf{pr}}$ consistently begins with the steepest curve, and its overall height exceeds the baselines. Also note that $\mathcal{G}_{\mathsf{av}}$ is inferior to the rest. In contrast, $\mathcal{G}_{\mathsf{sq}}$ and $\mathcal{G}_{\mathsf{h}}$ perform similarly to $\mathcal{G}_{\mathsf{pr}}$. While so, we later inspect the three in several case studies, and argue that the results of $\mathcal{G}_{\mathsf{pr}}$ are more useful, even though it is not captured by AOPC^k.

Figures 3 and 4 show the quality of T_c when applying the aggregation in an anytime manner. We measure the quality of the set of k candidates at different times during the computation, until completion. Hence, each chart shows the change of AOPC^k over time as well as the total running time.

Figure 3 shows that increasing δ improves *both the running time and quality* of $\mathcal{G}_{\mathsf{pr}}$. An exception is "dilemma +" where we get reduced runtime but a slight decrease in quality. This suggests that the aggregation compensates for the reduction in confidence (number of samples) for the Anchor algorithm. It also confirms the conjecture that the default value of δ, namely 0.1, is too strict within aggregation. Combining the remaining optimizations shortens the overall running time; while it reduced the quality in the "corona +" experiment, it generally did not impair and sometimes improved the quality (e.g., "corona -" and "home-spam -/+"). This experiment shows that, overall, the optimizations are highly beneficial.

Figure 4 shows that each optimization alone accelerates the computation. The *confidence* optimization (Equation (11)) incurs the most significant reduction in quality. In contrast, the *masking* optimization consistently improves the overall quality. Applying all optimizations together yields the shortest running time, but generally at the cost of lowering the AOPC^k scores (with the exception of "corona -" where the impact on the score is positive).

In Figure 5, we measure the percentage of shared terms between T_c of $\mathcal{G}_{\mathsf{pr}}$ to that of its optimizations. The intersection is generally around 80%. Note that a drop in the intersection does not necessarily imply a drop in quality, since different terms can have similar quality (as shown in Figures 3 and 4).

toy-spam -	toy-spam +
small, broke, disappointed, waste, would, smaller, cheap, thought, money, poor	great, love, loves, fun, favorite, awesome, wonderful, loved, classic, perfect

Table 1: Top-10 terms: negative vs. positive (no stop words).

toy-spam -	home-spam +
$\mathcal{G}_{\mathsf{pr}}$: not, but, disappointment, broke, small, it, would, waste, only, price	$\mathcal{G}_{\mathsf{pr}}$: great, love, best, excellent, perfect, loves, wonderful, happy, pleased, good
$\mathcal{G}_{\mathsf{av}}$: narrower, tipping, expanding, mouths, stains, tone, files, obscure, faint, health	$\mathcal{G}_{\mathsf{av}}$: castle, boys, visited, wars, kumar, vibrant, lavender, implement, farewell, silky
$\mathcal{G}_{\mathsf{sq}}$: not, but, it, the, disappointment, to, this, broke, was, small	$\mathcal{G}_{\mathsf{sq}}$: great, love, best, good, and, my, I, perfect, loves, ever

Table 2: Comparing different aggregations of the toy-spam and home-spam datasets.

Case Studies

We now discuss several case studies from our experiments. Table 1 shows the top terms for the toy-spam dataset under $\mathcal{G}_{\mathsf{pr}}$. We can see that terms for spam reviews are mostly positive (promoting products). On the other hand, non-spam reviews use more negative adjectives and commonly include customer complaints. (As an exception, "thought" occurs in non-spams in 80% of the cases, and is typically used negatively to describe disappointment from a product.)

Table 2 shows the results for the toy-spam (negative) and home-spam (positive) datasets. We can see that $\mathcal{G}_{\mathsf{sq}}$ has many common insignificant words, such as "my," "it," "and," "the," and so on. The function $\mathcal{G}_{\mathsf{av}}$ selects rare words such as "mouths," "files," "visited," and "wars." The function $\mathcal{G}_{\mathsf{pr}}$ is balanced and selects different terms such as "wonderful" and "happy" (as positives) or "broke" and "waste" (as negatives).

Table 3 shows the top terms of the corona dataset for $\mathcal{G}_{\mathsf{pr}}$ and $\mathcal{G}_{\mathsf{sq}}$, each with its position for the other aggregation. We can see, for example, that indices of common words like "corona" and "19" are dropped for $\mathcal{G}_{\mathsf{pr}}$, while less common words that appeared more as anchors stay at the top. As insights on the dataset, the reader can see that "hand" and "help" are significant to the positive class, where the former is typically used in the context of hand sanitizing.

$\mathcal{G}_{\mathsf{pr}}$: hand, like, help, good, safe, please, thank, great (0-7), free (9), support (11), thanks (12), well (13), best (15), positive (18), better (21), care (20), love (22), safety (25), relief (26)
$\mathcal{G}_{\mathsf{sq}}$: hand, like, help, good, please, safe, thank, great (0-7), 19 (48), free (8), co (751), support (9), thanks (10), well (11), corona (752), best (12), store (33), grocery (42), positive (13)

Table 3: Top terms of the corona dataset under $\mathcal{G}_{\mathsf{pr}}$ and $\mathcal{G}_{\mathsf{sq}}$. Each term is attached the position by the other function.

Dataset	Addition	Drop (%)
toy-spam -	I was a bit (**disappointed** / unsatisfied) with this game's performance.	32/2
toy-spam -	It is just a (**small** / usual) item.	60/8
toy-spam +	This store contained (**classic** / board) games.	52/0
toy-spam +	That game's theme is (**love** / animals).	44/6
corona +	I bought (**hand** / -) sanitizers.	44/2
corona +	People should (**support** / back) others more.	62/1
corona -	People shouldn't fret over this (**crisis** / situation).	62/2
corona (-)	The pandemic affected (**crude** / -) oil prices.	57/7

Table 4: Sentences with top-10 anchors by $\mathcal{G}_{\mathsf{pr}}$ (in bold). Each sentence is appended to all documents with the opposite label. The average accuracy is shown before and after the change.

Counterfactual examples. In Table 4, we ran the following experiment, inspired by the work of Wallace et al. (2019) on the impact of concatenated text on the model's performance. For various tasks and classes, we manually generated short sentences with their top-10 terms of $\mathcal{G}_{\mathsf{pr}}$ (Optimized). Each sentence is appended to all documents with an opposite label. We then measured the drop in the overall accuracy of the classifier due to the change; that is, we compared the accuracy before to after the change. We then repeated the measurement when replacing the term with another word of a similar nature. Importantly, the word replacement is such that the meaning of the sentence should barely impact that classification, so one could expect similar drops. Nevertheless, we can see that applying this change reduces the accuracy by a considerable amount ($\sim 50\%$). This suggests that our method indeed finds significant terms for the model.

Conclusions

We studied the problem of identifying the top-k terms under an aggregation of their identification as anchors or non-anchors in the dataset. We proposed the probabilistic aggregation $\mathcal{G}_{\mathsf{pr}}$ as a way of accounting for both the frequency of words and their treatment likelihood of being anchors. Global aggregation over the Anchor explanations incurs a prohibitive computational cost. We proposed techniques for considerably accelerating the identification of the top-k terms and showed experimentally that we obtain an anytime solution that is much more useful for online analysis, reducing the time from hours to minutes and seconds. We focused on single-word terms, and it is left for future work to study the case of multiple words, where the challenge is bigger due to the number of candidates for the top-k terms. Finally, while the runtime optimizations were applied to Anchor, the general framework can be adapted to any method of local attribution scores, and such adaptations are also left for future work.

Acknowledgments

This research was supported by the German-Israeli Foundation for Scientific Research and Development (grant I-1502-407.6/2019), the German Research Foundation (project 412400621), the Israel Science Foundation (grant 448/20), an Azrieli Foundation Early Career Faculty Fellowship, and an AI Alignment grant from Open Philanthropy.

References

Arras, L.; Horn, F.; Montavon, G.; Müller, K.; and Samek, W. 2016. Explaining Predictions of Non-Linear Classifiers in NLP. In *Rep4NLP@ACL*, 1–7. Association for Computational Linguistics.

Atanasova, P.; Simonsen, J. G.; Lioma, C.; and Augenstein, I. 2020. A Diagnostic Study of Explainability Techniques for Text Classification. In *EMNLP (1)*, 3256–3274. Association for Computational Linguistics.

Bastings, J.; Ebert, S.; Zablotskaia, P.; Sandholm, A.; and Filippova, K. 2022. "Will You Find These Shortcuts?" A Protocol for Evaluating the Faithfulness of Input Salience Methods for Text Classification. In *EMNLP*, 976–991. Association for Computational Linguistics.

Chen, S. F.; and Goodman, J. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4): 359–394.

Danilevsky, M.; Qian, K.; Aharonov, R.; Katsis, Y.; Kawas, B.; and Sen, P. 2020a. A Survey of the State of Explainable AI for Natural Language Processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 447–459. Suzhou, China: Association for Computational Linguistics.

Danilevsky, M.; Qian, K.; Aharonov, R.; Katsis, Y.; Kawas, B.; and Sen, P. 2020b. A Survey of the State of Explainable AI for Natural Language Processing. In *AACL/IJCNLP*, 447–459. Association for Computational Linguistics.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Ebert, S.; Jakobovits, A. S.; and Filippova, K. 2022. Understanding Text Classification Data and Models Using Aggregated Input Salience. *CoRR*, abs/2211.05485.

Gholizadeh, S.; and Zhou, N. 2021. Model Explainability in Deep Learning Based Natural Language Processing. *CoRR*, abs/2106.07410.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2019. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.*, 51(5): 93:1–93:42.

Guidotti, R.; Monreale, A.; Turini, F.; Pedreschi, D.; and Giannotti, F. 2018. A Survey Of Methods For Explaining Black Box Models. *CoRR*, abs/1802.01933.

He, P.; Liu, X.; Gao, J.; and Chen, W. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *CoRR*, abs/2006.03654.

Lertvittayakumjorn, P.; Specia, L.; and Toni, F. 2020. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. In *EMNLP (1)*, 332–348. Association for Computational Linguistics.

Lu, J.; Zhang, C.; Xie, Z.; Ling, G.; Zhou, T. C.; and Xu, Z. 2019. Constructing Interpretive Spatio-Temporal Features for Multi-Turn Responses Selection. In *ACL (1)*, 44–50. Association for Computational Linguistics.

Lundberg, S. M.; and Lee, S. 2017. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874.

Mor, A.; Belinkov, Y.; and Kimelfeld, B. 2023. Accelerating the Global Aggregation of Local Explanations. arXiv:2312.07991.

Pruthi, G.; Liu, F.; Kale, S.; and Sundararajan, M. 2020. Estimating Training Data Influence by Tracing Gradient Descent. In *NeurIPS*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR*, abs/1602.04938.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Samek, W.; Binder, A.; Montavon, G.; Bach, S.; and Müller, K. 2015. Evaluating the visualization of what a Deep Neural Network has learned. *CoRR*, abs/1509.06321.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Sargent, R. 2000. Optimal control. *Journal of Computational and Applied Mathematics*, 124(1): 361–371. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

Shrikumar, A.; Greenside, P.; Shcherbina, A.; and Kundaje, A. 2016. Not Just a Black Box: Learning Important Features Through Propagating Activation Differences. *ArXiv*, abs/1605.01713.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR*, abs/1312.6034.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic Attribution for Deep Networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, 3319–3328. PMLR.

van der Linden, I.; Haned, H.; and Kanoulas, E. 2019. Global Aggregations of Local Explanations for Black Box models. *CoRR*, abs/1907.03039.

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *EMNLP-IJCNLP*, 2153–2162. Hong Kong, China: Association for Computational Linguistics.

Xie, Q.; Ma, X.; Dai, Z.; and Hovy, E. H. 2017. An Interpretable Knowledge Transfer Model for Knowledge Base Completion. In *ACL (1)*, 950–962. Association for Computational Linguistics.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and Understanding Convolutional Networks. In *ECCV (1)*, volume 8689 of *Lecture Notes in Computer Science*, 818–833. Springer.