On Sample Based Explanation Methods for NLP: Efficiency, Faithfulness, and Semantic Evaluation

Wei Zhang *
Wayfair
Boston MA, USA
wzhang5@wayfair.com

Ziming Huang *
Sogou Inc
Beijing, China
hzmyouxiang@gmail.com

Yada Zhu
MIT-IBM Watson AI Lab
IBM Research, NY, USA
yzhu@us.ibm.com

Guangnan Ye
IBM Research
New York, USA
gye@us.ibm.com

Xiaodong Cui
IBM Research
New York, USA
xcui@us.ibm.com

Fan Zhang
IBM Data and AI
Littleton MA, USA
fzhang@us.ibm.com

Abstract

In the recent advances of natural language processing, the scale of the state-of-the-art models and datasets is usually extensive, which challenges the application of sample-based explanation methods in many aspects, such as explanation interpretability, efficiency, and faithfulness. In this work, for the first time, we can improve the interpretability of explanations by allowing arbitrary text sequences as the explanation unit. On top of this, we implement a hessian-free method with a model faithfulness guarantee. Finally, to compare our method with the others, we propose a semantic-based evaluation metric that can better align with humans' judgment of explanations than the widely adopted diagnostic or retraining measures. The empirical results on multiple real data sets demonstrate the proposed method's superior performance to popular explanation techniques such as Influence Function or TracIn on semantic evaluation.

1 Introduction

As complex NLP models such as the Transformers family (Vaswani et al., 2017; Devlin et al., 2019) become an indispensable tool in many applications, there are growing interests to explain the working mechanism of these "black-box" models. Among the vast of existing techniques for explaining machine learning models, Influence Functions (Hampel, 1974; Koh and Liang, 2017) that uses training instances as explanations to a model's behavior have gained popularity in NLP very recently. Different from other methods such as using input erasure (Li et al., 2016), saliency maps or attention matrices (Serrano and Smith, 2019; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019) that only look at how a specific input or input sequence impacts the model decision, explaining with training instances can cast light on the knowledge a model has encoded about a problem, by answering questions like '*what knowledge did the model capture from which training instances so that it makes decision in such a manner during test?*'. Very recently, the method has been applied to explain BERT-based (Devlin et al., 2019) text classification (Han et al., 2020; Meng et al., 2020b) and natural language inference (Han et al., 2020) models, as well as to aid text generation for data augmentation (Yang et al., 2020a) using GPT-2 (Radford et al., 2019). Although useful, Influence Function may not be entirely bullet-proof for NLP applications.

First, following the original formulation (Koh and Liang, 2017), the majority of existing works use entire training instances as explanations. However, for long natural language texts that are common in many high-impact application domains (e.g., healthcare, finance, or security), it may be difficult, if not impossible, to comprehend an entire instance as an explanation. For example, a model's decision may depend only on a specific part of a long training instance.

Second, for modern NLP models and large-scale datasets, the application of Influence Functions can lead to prohibitive computing costs due to inverse Hessian matrix approximation. Although hessian-free influence score such as `TracIn` (Pruthi et al., 2020b) was introduced very recently, it may not be faithful to the model in question and can result in spurious explanations for the involvement of sub-optimal checkpoints.

Last, the evaluation of explanation methods, in particular, for the training-instance-based ones, remains an open question. Previous evaluation is either under an over-simplified assumption on the agreement of labels between training and test instances (Hanawa et al., 2020; Han et al., 2020) or

*Equal Contribution. Wei Zhang did the work while being a research scientist at IBM T.J. Watson Research Center at Yorktown Heights, NY, USA; Ziming Huang was a research scientist at IBM Research Lab at Beijing, China.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5399–5411
August 1–6, 2021.

is based on indirect or manual inspection (Hooker et al., 2019; Meng et al., 2020b; Han et al., 2020; Pruthi et al., 2020a). A method to automatically measure the semantic relations at scale and that highly correlates to human judgment is still missing in the evaluation toolset.

To address the above problems, we propose a framework to explain model behavior that includes both a set of new methods and a new metric that can measure the semantic relations between the test instance and its explanations. The new method allows for arbitrary text spans as the explanation unit and is Hessian-free while being faithful to the final model. Our contributions are:

1. We propose a new explanation framework that can use arbitrary explanation units as explanations and be Hessian-free and faithful at the same time;

2. A new metric to measure the semantic relatedness between a test instance and its explanation for BERT-based deep models.

2 Preliminaries

Suppose a model parameterized by $\hat{\theta}$ is trained on classification dataset $D = \{D^{train}, D^{test}\}$ by empirical risk minimization over D^{train}. Let $z = (\mathbf{x}, y) \in D^{train}$ and $z' = (\mathbf{x}', y') \in D^{test}$ denote a training and a test instance respectively, where $\mathbf{x}$ is a token sequence, and y is a scalar. The goal of training instance based explanation is to provide for a given test z' an ordered list of training instances as explanation. Two notable methods to calculate the influence score are `IF` and `TracIn`:

`IF` (Koh and Liang, 2017) assumes the influence of z can be measured by perturbing the loss function L with a fraction of the loss on z, and obtain

$$\mathcal{I}_{\text{pert,loss}}(z, z'; \hat{\theta}) = -\nabla_\theta L(z', \hat{\theta}) H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}), \quad (1)$$

where H is the Hessian matrix calculated on the entire training dataset, a potential computation bottleneck for large dataset D and complex model with high dimensional $\hat{\theta}$.

`TracIn` (Pruthi et al., 2020b) instead assumes the influence of a training instance z is the sum of its contribution to the overall loss all through the entire training history, and conveniently it leads to

$$\texttt{TracIn}(z, z') = \sum_i \eta_i \nabla_{\hat{\theta}_i} L(\hat{\theta}_i, z) \nabla_{\hat{\theta}_i} L(\hat{\theta}_i, z'), \quad (2)$$

where i iterates through the checkpoints saved at different training steps and η_i is a weight for each checkpoint. `TracIn` does not involve Hessian matrix and more efficient to compute. We can summarize the key differences between them according to the following desiderata of an explanation method:

Efficiency for each z', `TracIn` requires $\mathcal{O}(CG)$ where C is the number of models and G is the time spent for gradient calculation; whereas `IF` needs $\mathcal{O}(N^2G)$ where N is the number of training instances, and $N >> C$ in general. [1]

Faithfulness `IF` is *faithful* to $\hat{\theta}$ since all its calculation is based on a single final model, yet `TracIn` may be less faithful to $\hat{\theta}$ since it obtains gradients from a set of checkpoints [2].

Interpretability Both methods use the entire training instance as an explanation. Explanations with a finer-grained unit, e.g., phrases, may be easier to interpret in many applications where the texts are lengthy.

3 Proposed Method

To improve on the above desiderata, a new method should be able to: 1) use any appropriate granularity of span(s) as the explanation unit; 2) avoid the need of Hessian while maintaining faithfulness. We discuss the solutions for both in Section 3.1 and 3.2, and combine them into one formation in Section 3.3 followed by critical implementation details.

3.1 Improved Interpretability with Spans

To achieve 1), we first start with influence functions (Koh and Liang, 2017) and consider an arbitrary span of training sequence $\mathbf{x}$ to be evaluated for the qualification as explanation [3]. Our core idea is *to see how the model loss on test instance z' changes*

[1] some approximation such as hessian-inverse-vector-product (Baydin et al., 2016) may improve efficiency to $O(NSG)$ where S is the approximation step and $S < N$

[2] We may say `TracIn` is faithful to the *data* rather than to the *model*. And in the case where checkpoint averaging can be used as model prediction, the number of checkpoints may be too few to justify Eq. 2.

[3] the method can be trivially generalized to multiple spans

with the training span's importance. The more important a training span is to z', the greater this influence score should be. We derive it in three following steps.

First, we define the training span from token i to token j to be $\mathbf{x}_{ij}$, and the sequence with $\mathbf{x}_{ij}$ masked is $\mathbf{x}_{-ij} = [x_0, ..., x_{i-1}, [\text{MASK}], ..., [\text{MASK}], x_{j+1}, ...]$ and its corresponding training data is z_{-ij}. We use logit difference (Li et al., 2020) as importance score based on the empirical-risk-estimated parameter $\hat{\theta}$ obtained from D^{train} as: $\texttt{imp}(\mathbf{x}_{ij}|z, \hat{\theta}) = \text{logit}_y(\mathbf{x}; \hat{\theta}) - \text{logit}_y(\mathbf{x}_{-ij}; \hat{\theta})$, where every term in the right hand side (RHS) is the logit output evaluated at a model prediction y from model $\hat{\theta}$ right before applying the SoftMax function. *This equation tells us how important a training span is.* It is equivalent to the loss difference

$$\texttt{imp}(\mathbf{x}_{ij}|z; \hat{\theta}) = \mathcal{L}(z_{-ij}; \hat{\theta}) - \mathcal{L}(z; \hat{\theta}), \quad (3)$$

when the cross entropy loss $\mathcal{L}(z; \theta) = -\sum_{y_i} \mathcal{I}(y = y_i)\text{logit}_{y_i}(\mathbf{x}; \theta)$ is applied.

Then, we measure $\mathbf{x}_{ij}$'s influence on model $\hat{\theta}$ by adding a fraction of $\texttt{imp}(\mathbf{x}_{ij}|z; \hat{\theta})$ scaled by a small value ϵ to the overall loss and obtain $\hat{\theta}_{\epsilon,\mathbf{x}_{ij}|z} := \text{argmin}_\theta E_{z_i \in D^{train}}[\mathcal{L}(z_i, \theta)] + \epsilon\mathcal{L}(z_{-ij}; \theta) - \epsilon\mathcal{L}(z; \theta)$. Applying the classical result in (Cook and Weisberg, 1982; Koh and Liang, 2017), the influence of up-weighing the importance of x_{ij} on $\hat{\theta}$ is

$$\frac{d\hat{\theta}_{\epsilon,\mathbf{x}_{ij}|z}}{d\epsilon}\Big|_{\epsilon=0} = H_{\hat{\theta}}^{-1}(\nabla_{\hat{\theta}} L(z; \hat{\theta}) - \nabla_{\hat{\theta}} L(z_{-ij}; \hat{\theta})).$$

Finally, applying the above equation and the chain rule, we obtain the influence of $\mathbf{x}_{ij}$ to z' as:

$$\texttt{IF}^{+}(\mathbf{x}_{ij}|z, z'; \hat{\theta}) := \nabla_\epsilon L(z'; \hat{\theta}_{\epsilon,\mathbf{x}_{ij}|z})|_{\epsilon=0}$$
$$= \nabla_\theta L(z'; \hat{\theta}) H_{\hat{\theta}}^{-1}(\nabla_\theta L(z; \hat{\theta}) - \nabla_\theta L(z_{-ij}; \hat{\theta})).$$

$\texttt{IF}^{+}$ measures the influence of a training span on an entire test sequence. Similarly, we also measure the influence of a training span to a test span x'_{kl} by applying Eq. 3 and obtain

$$\begin{aligned}&\texttt{IF}^{++}(\mathbf{x}_{ij}|z, \mathbf{x}'_{kl}|z'; \hat{\theta}) \\ :=&\nabla_\epsilon L(z'_{-kl}; \hat{\theta}_{\epsilon,\mathbf{x}_{ij}|z}) - \nabla_\epsilon L(z'; \hat{\theta}_{\epsilon,\mathbf{x}_{ij}|z})|_{\epsilon=0} \\ =&(\nabla_\theta L(z'_{-kl}; \hat{\theta}) - \nabla_\theta L(z'; \hat{\theta})) \\ &H_{\hat{\theta}}^{-1}(\nabla_\theta L(z; \hat{\theta}) - \nabla_\theta L(z_{-ij}; \hat{\theta})).\end{aligned}$$

The complete derivation can be found in Appendix.

On the choice of Spans Theoretically, $\texttt{IF}^{+}$ and $\texttt{IF}^{++}$ can be applied to any text classification problem and dataset with an appropriate choice of the span. If no information about valid span is available, shallow parsing tools or sentence split-tools can be used to shatter an entire text sequence into chunks, and each chunk can be used as span candidates. In this situation, the algorithm can work in two steps: 1) using masking method (Li et al., 2020) to determine the important test spans; and 2) for each span we apply $\texttt{IF}^{++}$ to find training instances/spans as explanations.

Usually, we can choose top-K test spans, and even can choose K=1 in some cases. In this work, we look at the later case without loss of generality, and adopt two aspect-based sentiment analysis datasets that can conveniently identify a deterministic span in each text sequence, and frame the span selection task as a Reading Comprehension task (Rajpurkar et al., 2016). We discuss the details in Section 5. Note that the discussion can be trivially generalized to the case where K>1 using Bayesian approach such as $\text{imp}(x_{ij}) = \mathbf{E}_{P(x'_{kl})}[\text{imp}(x_{ij}|x_{kl})']$ which can be explored in future work.

3.2 Faithful & Hessian-free Explanations

To achieve 2), we would start with the method of `TracIn` (Pruthi et al., 2020b) described in Eq. 2 which is Hessian free by design. `TracIn` defines the contribution of a training instance to be the sum of its contribution (loss) throughout the entire training life cycle, which eradicated the need for Hessian. However, this assumption is drastically different from `IF`'s where the contribution of z is obtained solely from the final model $\hat{\theta}$. By nature, `IF` is a *faithful method*, and its explanation is *faithful to* $\hat{\theta}$, and `TracIn` in its vanilla form is arguably not a faithful method.

Proposed treatment Based on the assumption that the influence of z on $\hat{\theta}$ is the sum of influences of all variants close to $\hat{\theta}$, we define a set of "faithful" variants satisfying the constraint of $\{\hat{\theta}_i | 1 > \delta >> ||\hat{\theta}_i - \hat{\theta}||_2\}$, namely δ-faithful to $\hat{\theta}$. The smaller δ is, the more faithful the explanation method is. Instead, the δ for `TracIn` can be arbitrary large without faithfulness guarantees, as some checkpoints can be far from the final $\hat{\theta}$. Thus, we construct a δ-faithful explanation method that

mirrors TracIn as:

$$\texttt{TracInF}(z, z') = \sum_i \nabla_{\hat{\theta}+\delta_i} L(\hat{\theta}+\delta_i, z) \nabla_{\hat{\theta}+\delta_i} L(\hat{\theta}+\delta_i, z').$$

The difference between TracIn and TracInF is that the checkpoints used in TracIn are correlated in time whereas all variants of TracInF are conditionally independent. Finding a proper δ_i can be tricky. If ill-chosen, δ_i may diverge $\hat{\theta}$ so much that hurts gradient estimation. In practice, we estimate $\delta_i = \eta_i g(z_i|\hat{\theta})$ obtained from a single-step gradient descent $g(z_i|\hat{\theta})$ with some training instance z_i on model $\hat{\theta}$, scaled by an i-specific weighting parameter η_i, which in the simplest case is uniform for all i. Usually η_i should be small enough so that $\hat{\theta}+\delta_i$ can stay close to $\hat{\theta}$. In this paper we set η as the model learning rate for proof of concept.

Is TracInF faithful? First, any $\hat{\theta}+\delta_i$ is close to $\hat{\theta}$. Under the assumption of Lipschitz continuity, there exists a $k \in \mathbb{R}^+$ such that $\nabla L(\hat{\theta}+\delta_i, z)$ is bounded around $\nabla L(\hat{\theta}, z)$ by $k|\eta_i g^2(z_i|\hat{\theta})|$, the second derivative, because $|\nabla L(\hat{\theta}+\delta_i, z) - \nabla L(\hat{\theta}, z)| < k|\eta_i g^2(z_i|\hat{\theta})|$. A proper η_i can be chosen so that the right hand side (RHS) is sufficiently small to bound the loss within a small range. Thus, the gradient of loss, and in turn the TracInF score can stay δ-faithful to $\hat{\theta}$ for an sufficiently small δ, which TracIn can not guarantee.

3.3 The Combined Method

By combining the insights from Section 3.1 and 3.2, we obtain a final form named $\texttt{TracIn}^{++}$:

$$\texttt{TracIn}^{++}(x'_{kl}|z', x_{ij}|z; \hat{\theta}) = \sum_i \big[\nabla\mathcal{L}(\hat{\theta}+\delta_i, z'_{-kl}) - \nabla\mathcal{L}(\hat{\theta}+\delta_i, z')\big] \big[\nabla\mathcal{L}(\hat{\theta}+\delta_i, z) - \nabla\mathcal{L}(\hat{\theta}+\delta_i, z_{-ij})\big].$$

This ultimate form mirrors the $\texttt{IF}^{++}$ method, and it satisfies all of our desiderata on an improved explanability method. Similarly, $\texttt{TracIn}^{+}$ that mirrors $\texttt{IF}^{+}$ is

$$\texttt{TracIn}^{+}(z', x_{ij}|z; \hat{\theta}) = \sum_i \nabla\mathcal{L}(z'; \hat{\theta}+\delta_i) \big[\nabla\mathcal{L}(\hat{\theta}+\delta_i, z) - \nabla\mathcal{L}(\hat{\theta}+\delta_i, z_{-ij})\big].$$

3.4 Additional Details

Since the RHS of IF, $\texttt{IF}^{+}$ and $\texttt{IF}^{++}$ equations all involve the inverse of Hessian Matrix, here we discuss the computation challenge. Following (Koh and Liang, 2017), we adopt the vector-Hessian-inverse-product (VHP) with stochastic estimation (Baydin et al., 2016). The series of stochastic updates, one for each training instance, is performed by the vhp() function in the `torch.autograd.functional` package and the update stops until convergence. Unfortunately, we found that naively applying this approach leads to VHP explosion due to large parameter size. To be specific, in our case, the parameters are the last two layers of `RoBERTa-large` (Liu et al., 2019) plus the output head, a total of 12M parameters per gradient vector. To stabilize the process, we take three approaches: 1) applying gradient clipping (set to 100) to avoid accumulating the extreme gradient values; 2) adopting early termination when the norm of VHP stabilizes (usually < 1000 training instances, i.e., the depth); and 3) slowly decaying the accumulated VHP with a factor of 0.99 (i.e., the damp) and update with a new vhp() estimate with a small learning rate (i.e., the scale) of 0.004. Please refer to our code for more details. Once obtained, the VHP is first cached and then retrieved to perform the dot-product with the last term. The complexity for each test instance is $\mathcal{O}(dt)$ where d is the depth of estimation and t is the time spent on each vhp() operation. The time complexity of different IF methods only vary on a constant factor of two.

For each of TracIn, $\texttt{TracIn}^{+}$ and $\texttt{TracIn}^{++}$, we need to create multiple model variants. For TracIn, we save three checkpoints of the most recent training epochs; For $\texttt{TracIn}^{+}$ or $\texttt{TracIn}^{++}$, we start with the same checkpoint and randomly sample a mini-batch 3 times and perform one-step training (learning rate 1E-4) for each selection to obtain three variants. We do not over-tune those hyper-parameters for replicability concerns.

4 Evaluation Metrics

This section introduces our semantic evaluation method, followed by a description of two other popular metrics for comparison.

4.1 Semantic Agreement (Sag)

Intuitively, a rational explanation method should rank explanations that are semantically related to the given test instance relatively higher than the less relevant ones. *Our idea is to first define the*

semantic representation of a training span $\mathbf{x}_{ij}$ *of* z *and measure its similarity to that of a test span* $\mathbf{x}'_{kl}$ *of* z'. Since our method uses `BERT` family as the base model, we obtain the embedding of a training span by the difference of x and its span-masked version x_{ij} as

$$emb(x_{ij}) = emb(x) - emb(x_{-ij}), \quad (4)$$

where emb is obtained from the embedding of sentence start token such as "[CLS]" in BERT (Devlin et al., 2019) at the last embedding layer. To obtain embedding of the entire sequence we can simply use the $emb(x)$ without the last term in Eq. 4. Thus, all spans are embedded in the same semantic space and the geometric quantities such as cosine or dot-product can measure the similarities of embeddings. We define the semantic agreement `Sag` as:

$$\texttt{Sag}(z', \{z\}|_1^K) = \frac{1}{K}\sum_z cos(emb(x_{ij}|z), emb(x'_{kl}|z')), \quad (5)$$

Intuitively, the metric measures the degree to which top-K training spans align with a test span on semantics.

4.2 Other metrics

Label Agreement (`Lag`) label agreement (Hanawa et al., 2020) assumes that the label of an explanation z should agree with that of the text case z'. Accordingly, we retrieve the top-K training instances from the ordered explanation list and calculate the label agreement (`Lag`) as follows:

$$\texttt{Lag}(z', \{z\}|_1^N) = \frac{1}{K}\sum_{k\in[1,K]} \mathcal{I}(y' == y_k),$$

where $\mathcal{I}(\cdot)$ is an indicator function. `Lag` measures the degree to which the top-ranked z agree with z' on class label, e.g., if the sentiment of the test z' and explanation z agree.

Re-training Accuracy Loss (`Ral`) `Ral` measures the loss of test accuracy after removing the top-K most influential explanations identified by an explanation method (Hanawa et al., 2020; Hooker et al., 2019; Han et al., 2020). The assumption is that the higher the loss the better the explanation method is. Formally,

$$\texttt{Ral}(f, \hat{\theta}) = Acc(\hat{\theta}) - Acc(\hat{\theta}'),$$

where $\hat{\theta}'$ is the model re-trained by the set $D^{train}/\{z\}|_1^K$. Notice the re-training uses the same set of hyper-parameter settings as training (Section 6.1). To obtain $\{z\}|_1^K$, we combine the explanation lists for all test instances (by score addition) and then remove the top-K from this list.

5 Data

Our criteria for dataset selection are two folds: 1. The dataset should have relatively high classification accuracy so that the trained model can behave rationally; and 2. The dataset should allow for easy identification of critical/useful text spans to compare span-based explanation methods. We chose two aspect-based sentiment analysis (ABSA) datasets; one is ATSA, a subset of **MAMS** (Jiang et al., 2019) for product reviews, where aspects are the terms in the text. The other is **sentihood** (Saeidi et al., 2016) of location reviews. We can identify the relevant span of an aspect term semi-automatically and train models with high classification accuracy in both datasets. (see Section 6.1 for details). Data statistics and instances are in Table 1 and 2.

	Train	Dev	Test
MAMS	11186	1332	1336
sentihood	2977	747	1491

Table 1: Data Statistics. Note that we regard each training instance as aspect-specific, i.e., the concatenation of aspect term and the text x as model input.

Automatic Span Annotation As shown in the colored text in Table 2, we extract the spans for each term to serve as explanation units for $\texttt{IF}^{+}$, $\texttt{IF}^{++}$, $\texttt{TracIn}^{+}$ and $\texttt{TracIn}^{++}$. To reduce annotation effort, we convert span extraction into a question answering task (Rajpurkar et al., 2016) where we use aspect terms to formulate questions such as "How is the *service*?" which concatenates with the text before being fed into pre-trained machine reading comprehension (RC) models. The output answer is used as the span. When the RC model fails, we use heuristics to extract words before and after the term word, up to the closest sentence boundary. See appendix for more details. We sampled a subset of 100 annotations and found that the RC model has about 70% of Exact Match (Rajpurkar et al., 2016) and the overall annotation has a high recall of over 90% but low EM due to the involvement of heuristics.

Dataset	Text	Aspect	Sentiment
MAMS	the *service* was impeccable, the *menu* traditional but inventive and presentation for the most part excellent but the *food* itself came up short.	service menu food	+ + -
sentihood	i live in location2 and i love it location1 just stay away from location1 lol.	location1 location2	- +

Table 2: Dataset instances. In text, each aspect has a supporting span which we annotate semi-automatically. We choose a subset where test instances

(Not) Mitigating the Annotation Error Wrongly-annotated spans may confuse the explanation methods. For example, as shown in 2, if the span of *location2* is annotated as "I love it", span-based explanation methods will use it to find wrong examples for explanation. Thus test instances with incorrectly annotated spans are omitted, i.e., no tolerance to annotation error for test instances. To the contrary, for training instances, we do not correct the annotation error. The major reason is the explanation methods have a chance to rank the wrongly annotated spans lower (its importance score `imp`() of Eq. 3 can be lower and in turn for its influence scores.) Also, It is labor-intensive to do so.

6 Experiments

6.1 Model Training Details

We train two separate models for MAMS and sentihood. The model's input is the concatenation of the aspect term and the entire text, and the output is a sentiment label. The two models share similar settings: 1. they both use RoBERTa-large (Liu et al., 2019) from Huggingface (Wolf et al., 2019) which is fed into the `BertForSequenceClassification` function for initialization. We fine-tune the parameters of the last two layers and the output head using a batch size of 200 for ATSA and 100 for sentihood and max epochs of 100. We use AdamW optimizer (Loshchilov and Hutter, 2019) with weight decay 0.01 and learning rate 1E-4. Both models are written in Pytorch and are trained on a single Tesla V100 GPU and took less than 2 hours for each model to train. The models are selected on dev set performance, and both trained models are state-of-the-art: 88.3% on MAMS and 97.6% for sentihood at the time of writing.

6.2 Comparing Explanation Methods

We compare the six explanation methods on two datasets and three evaluation metrics in Table 3 from which we can draw the following conclusions:

1) `TracIn` family outperforms `IF` family according to `Sag` and `Lag` metrics. We see that both metrics are robust against the choice of K. It it worth noting that `TracIn` family methods are not only efficient, but also effective for extracting explanations compared to `IF` family as per `Sag` and `Lag`.

2) Span-based methods (with +) outperform Vanilla methods (w/o +). It is good news because an explanation can be much easier to comprehend if we can highlight essential spans in text, and $\texttt{IF}^{++}$ and $\texttt{TracIn}^{++}$ shows us that such highlighting can be justified by their superiority on the evaluation of `Sag` and `Lag`.

3) `Sag` and `Lag` shows a consistent trend of $\texttt{TracIn}^{++}$ and $\texttt{IF}^{++}$ being superior to the rest of the methods, while `Ral` results are inconclusive, which resonates with the findings in (Hooker et al., 2019) where they also observed randomness after removing examples under different explanation methods. This suggests that the re-training method may not be a reliable metric due to the randomness and intricate details involved in the re-training process.

4) The `Sag` measures $\texttt{TracIn}^{+}$ differently than `Lag` shows that `Lag` may be an over-simplistic measure by assuming that label y can represent the entire semantics of $\mathbf{x}$, which may be problematic. But `Sag` looks into the $\mathbf{x}$ for semantics and can properly reflect and align with humans judgments.

The Impact of K on Metrics One critical parameter for evaluation metrics is the choice of K for `Sag` and `Lag` (We do not discuss K for `Ral` due to its randomness). Here we use 200 MAMS test instances as subjects to study the influence of K, as shown in Figure 1.

		IF	IF$^+$	IF^{++}	TracInF	TracIn$^+$	TracIn^{++}
Faithful to $\hat{\theta}$?		✓	✓	✓	✓	✓	✓
Hessian-free?		✗	✗	✗	✓	✓	✓
Interpretable explanations?		✗	✓	✓✓	✗	✓	✓✓
MAMS	Sag(K=10)	14.22	17.17	21.74	15.89	22.65	**23.92**
	Sag(K=100)	14.65	15.10	19.83	15.97	19.54	**21.32**
	Lag(K=10)	21.63	25.66	65.41	38.20	08.60	**78.03**
	Lag(K=100)	26.07	25.66	62.52	43.19	06.27	**75.02**
	Ral(- top 20%)	09.80	05.64	03.55	09.80	11.89	**16.05**
	Ral(- top 50%)	**28.55**	01.47	18.14	22.30	05.64	18.14
sentihood	Sag(K=10)	04.69	04.75	22.54	03.07	00.98	**26.21**
	Sag(K=50)	03.56	07.82	22.21	01.78	01.61	**23.43**
	Lag(K=10)	53.00	41.91	61.96	55.91	18.22	**66.65**
	Lag(K=50)	56.38	44.05	63.16	59.66	17.49	**66.72**
	Ral(- top 20%)	10.56	**16.21**	06.91	09.23	06.91	09.23
	Ral(- top 50%)	16.21	18.53	11.05	**27.83**	9.23	4.58

Table 3: Performance of difference explanation methods on 200 test cases on each dataset. For Sag and Lag we set $K \in \{10, 100\}$; for Ral we set $K \in \{20\%, 50\%\}$, and Ral we consider removing the top 20% or 50% from the ordered training instance list. Computation time for IF family is about 20 minutes per test instance with recursion depth 1000 (the minimal value to guarantee convergence) on a Tesla V100 GPU. The time for TracIn family only depends on gradient calculation, which is trivial compared to IF family.

We found that as K increases, all methods, except for IF and TracInF, decrease on Sag and Lag. The decrease is favorable because the explanation method is putting useful training instances before less useful ones. In contrast, the increase suggests the explanation method fails to rank useful ones on top. This again confirms that span-based explanation can take into account the useful information in $\mathbf{x}$ and reduce the impact of noisy information involved in IF and TracInF.

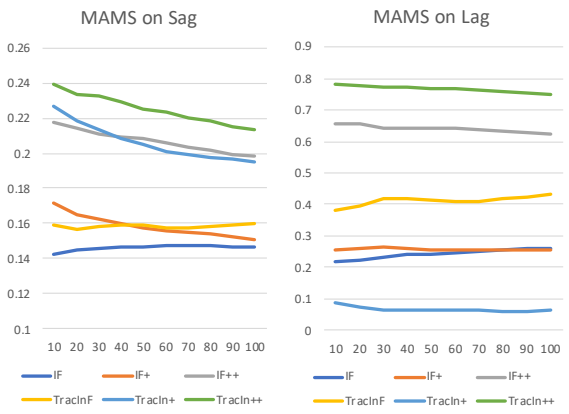

Figure 1: Sag and Lag v.s. K values on 200 MAMS test instances.

6.3 Comparing Faithfulness

How faithful our proposed TracIn^{++} to $\hat{\theta}$? To answer this question, we first define the notion of *strictly faithful explanation* and then test an explanation method's faithfulness against it. Note that none of the discussed methods is strictly faithful, since IF^{++} used approximated inverse-Hessian and TracIn^{++} is a δ away from being strictly faithful. To obtain ground truth, we modify TracIn^{++} to use a single checkpoint $\hat{\theta}$ as the "ultimately faithful" explanation method [4]. Then, we obtain an explanation list for each test instance and compute its Spearman Correlation with the list obtained from the ground truth. The higher the correlation, the more faithful the method is.

In Table 4 we discovered that TracIn^{++} has similar mean as IF^{++} but has a much lower variance, showing its stability over IF^{++}. This aligns with the finding of Basu et al. (2021) which argues that in deep non-convex networks, influence function usually is non-stable across test instances. TracIn family arguably may be a promising direction to stability. Both methods are more faithful to Ground truth than Control that uses checkpoints,

[4]The choice of ground truth can also be the exact computation of inverse-Hessian in IF (our future work). Faithfulness does not equal to correctness; there is no guarantee the ground truth is a valid explanation method, but it can be a valid benchmark for faithfulness

Method	Spearman Mean	Var.
Control	55.11	4.84
TracIn^{++}	60.14	3.57
IF^{++}	59.37	20.50

Table 4: Comparison of Correlation with Ground truth. The experiment is run 5 times each; "Control" is only different from TracIn^{++} on the models used: "control" uses three checkpoints of the latest epochs, but TracIn^{++} uses three δ-faithful model variants.

showing that the model "ensemble" around $\hat{\theta}$ may be a better choice than "checkpoint averaging" for model explanations. Further explorations may be needed since there are many variables in this comparison.

7 A Case Study

Table 5 demonstrate the differences of explanation methods. In action, TracIn^{++} shows both the test span and explanation span to a user; TracIn^{+} shows only the training span, and TracIn does not show spans. Interestingly we can observe the top-1 explanation found by TracIn^{++} is more semantically related than others in the example, a common pattern among the test cases.

8 Related Work

Popular explanation methods include gradient-based (Sundararajan et al., 2017), attention-based (Clark et al., 2019; Jain and Wallace, 2019; Wiegreffe and Pinter, 2019), as well as sample-based (Koh and Liang, 2017; Yeh et al., 2018; Pruthi et al., 2020b) methods.

Major Progress on Sample-based Explanation Methods There have been a series of recent efforts to explain black-box deep neural nets (DNN), such as LIME (Ribeiro et al., 2016) that approximates the behavior of DNN with an interpretable model learned from local samples around prediction, Influence Functions (Koh and Liang, 2017; Koh et al., 2019) that picks training samples as explanation via its impact on the overall loss, and Exemplar Points (Yeh et al., 2018) that can assign weights to training samples. TracIn (Pruthi et al., 2020b) is the latest breakthrough that overcomes the computational bottleneck of Influence Functions with the cost of faithfulness.

The Discussion of Explanation Faithfulness in NLP The issue of Faithfulness of Explanations was primarily discussed under the explanation generation context (Camburu et al., 2018) where there is no guarantee that a generated explanation would be faithful to a model's inner-workings (Jacovi and Goldberg, 2020). In this work, we discuss faithfulness in the sample-based explanations framework. The faithfulness to model either can be guaranteed only in theory but not in practice (Koh and Liang, 2017) or can not be guaranteed at all (Pruthi et al., 2020b).

Sample-based explanation methods for NLP Han et al. (2020) applied IF for sentiment analysis and natural language inference and also studied its utility on detecting data artefacts (Gururangan et al., 2019). Yang et al. (2020b) used Influence Functions to filter the generated texts. The one closest to our work is (Meng et al., 2020a) where a single word is used as the explanation unit. Their formation uses gradient-based methods for single words, while ours can be applied to any text unit granularity using text masking.

Explanation of NLP Models by Input Erasure Input erasure has been a popular trick for measuring input impact for NLP models by replacing input by zero vector (Li et al., 2016) or by marginalization of all possible candidate tokens (Kim et al., 2020) that arguably dealt with the out of distribution issue introduced by using zero as input mask. Similar to (Kim et al., 2020; Li et al., 2020; Jacovi and Goldberg, 2021) we also use "[MASK]" token, with the difference that we allow masking of arbitrary length of an input sequence.

Evaluations of Sample-based Methods A benchmark of evaluating sample-based explanation methods has not been agreed upon. For diagnostic purposes, Koh et al. (2017) proposed a self-explanation method that uses the training instances to explain themselves; Hanawa et al. (2020) proposed the label and instance consistency as a way of model sanity check. On the non-diagnostic setting, sample removal and re-training (Han et al., 2020; Hooker et al., 2019) assumes that removing useful training instances can cause significant accuracy loss; input enhancement method assumes useful explanations can also improve model's decision making at model input side (Hao, 2020), and manual inspections (Han et al., 2020; Meng et al., 2020a) were also used to examine if the

Test Case	been here a few times and food has always been good but service really suffers when it gets crowded.	+
TracIn^{++}	expected there to be more options for tapas the food was mediocre but the service **was pretty good.**	+
TracIn^{+}	decor **is simple yet functional** and although the staff are not the most attentive in the world, ...	+
TracInF	this place is the tourist fav of chinese food in the city, the service was fast, but the taste of the food is average, too much starch ...	0
IF^{++}	... the host was rude to us as we walked in, we stayed because the decor is **charming** and we wanted french food.	+
IF^{+}	the scene **a dark refurbished** dining car hosts plenty of hipsters in carefully selected thrift-store clothing.	+
IF	an unpretentious sexy atmosphere lends itself to the above average wine-list and a menu that can stand-up to any other restaurant ...	+

Table 5: Showcasing Top-1 Explanations. Aspect terms are in blue, and the spans are in bold font. TracInF do not highlight either training or testing span; TracIn^{+} highlights training span; TracIn^{++} highlights both training and test spans. TracIn^{++} and IF^{++} can help users understand which span of z influenced which span of z', which TracInF and IF do not provide.

meanings of explanations align with that of the test instance. In this paper, we automate this semantic examination using the embedding similarities.

9 Future Work

TracIn^{++} opens some new questions: 1) how can we generalize TracIn^{++} to cases where test spans are unknown? 2) Can we understand the connection between IF and TracIn which may spark discoveries on sample-based explanation methods? 3) How can we apply TracIn^{++} to understand sequence generation models?

Acknowledgement

This work is supported by the MIT-IBM Watson AI Lab. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies. We thank anonymous reviewers for their valuable feedback. We also thank your family for the support during this special time.

References

Samyadeep Basu, Philip Pope, and Soheil Feizi. 2021. Influence functions in deep learning are fragile. *ICLR*.

Atılım Güneş Baydin, Barak A Pearlmutter, and Jeffrey Mark Siskind. 2016. Tricks from deep learning. *arXiv preprint arXiv:1611.03777*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *NIPS*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of bert's attention. *BlackBoxNLP*, abs/1906.04341.

R Dennis Cook and Sanford Weisberg. 1982. *Residuals and influence in regression*. New York: Chapman and Hall.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational pretraining for semi-supervised text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5880–5894, Florence, Italy. Association for Computational Linguistics.

Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *ACL*.

Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. 2020. Evaluation criteria for instance-based explanation. *arXiv preprint arXiv:2006.04528*.

Yiding Hao. 2020. Evaluating attribution methods using white-box LSTMs. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 300–313, Online. Association for Computational Linguistics.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Alon Jacovi and Yoav Goldberg. 2021. Aligning Faithful Interpretations with their Social Attribution. *Transactions of the Association for Computational Linguistics*, 9:294–310.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. 2019. A challenge dataset and effective models for aspect-based sentiment analysis. In *EMNLP-IJCNLP*, pages 6281–6286.

Siwon Kim, Jihun Yi, Eunji Kim, and Sungroh Yoon. 2020. Interpretation of NLP models through input marginalization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3154–3167, Online. Association for Computational Linguistics.

Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. 2019. On the accuracy of influence functions for measuring group effects. *CoRR*, abs/1905.13289.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*, pages 1885–1894.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. *EMNLP*.

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. *CoRR*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Fanyu Meng, Junlan Feng, Danping Yin, Si Chen, and Min Hu. 2020a. A structure-enhanced graph convolutional network for sentiment analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 586–595, Online. Association for Computational Linguistics.

Yuxian Meng, Chun Fan, Zijun Sun, Eduard Hovy, Fei Wu, and Jiwei Li. 2020b. Pair the dots: Jointly examining training history and test stimuli for model interpretability. *arXiv preprint arXiv:2010.06943*.

Phiyodr. 2020. roberta-large-finetuned-squad2. `https://huggingface.co/phiyodr/bart-large-finetuned-squad2`. [Online; accessed 19-Dec-2020].

Danish Pruthi, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C Lipton, Graham Neubig, and William W Cohen. 2020a. Evaluating explanations: How much do explanations from the teacher aid students? *arXiv preprint arXiv:2012.00893*.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. 2020b. Estimating training data influence by tracking gradient descent. In *NIPS*.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Marzieh Saeidi, Guillaume Bouchard, Maria Liakata, and Sebastian Riedel. 2016. Sentihood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods. In *COLING*.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. *CoRR*, abs/1703.01365.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

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

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020a. G-daug: Generative data augmentation for commonsense reasoning. *arXiv preprint arXiv:2004.11546*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020b. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics.

Chih-Kuan Yeh, Joon Sik Kim, Ian E.H. Yen, and Pradeep Ravikumar. 2018. Representer point selection for explaining deep neural networks. In *NIPS*.

A Span extraction details

The model we apply the huggingface (Wolf et al., 2019) pre-trained RC model "phiyodr/roberta-large-finetuned-squad2" (Phiyodr, 2020) which is chosen based on our comparison to a set of similar models on SQuAD 2.0 dataset. We use the SQuAD 2.0-trained model instead of 1.0 because the data is more challenging since it involves multiple passages, and the model has to compare valid and invalid passages for answer span extraction, a case similar to the dataset we use. Templates we used are: The heuristics

How is the X?
How was the X?
How are the X?
How were the X?
How do you rate the X?
How would you rate the X?
How do you think of the X?
What do you think about the X?
What do you say about the X?
What happened to the X?
What did the X do?

Table 6: Templates for RC model

when the RC model fails: 1) We consider RC model fails when no span is extracted, or the entire text is returned as an answer. 2) We identify the location of the term in the text and expand the scope from the location both on the left and on the right, and when sentence boundary is found, we stop and return the span as the span for the term. Note that we do find cases where the words around a term do not necessarily talk about the term. However, we found such a case to be extremely rare.

B Derivation of IF^{++}

$$
\begin{aligned}
&\mathcal{I}_{\text{pert,loss}}(X_{ij}, z_{-kl}; \hat{\theta}) \\
&\quad := \nabla_\epsilon \text{imp}(X_{ij}|X; \hat{\theta}_{\epsilon, z_{-ij}, z})\Big|_{\epsilon=0} \\
&\quad = \frac{d\text{imp}(X_{ij}|X;\hat{\theta})}{d\hat{\theta}} \left(\frac{d\hat{\theta}_{\epsilon,z_{-kl},z}}{d\epsilon}\Big|_{\epsilon=0}\right) \\
&\quad = (\nabla_\theta O_y(X, \hat{\theta}) - \nabla_\theta O_y(X_{-ij}, \hat{\theta}))\left(\frac{d\hat{\theta}_{\epsilon,z_{-kl},z}}{d\epsilon}\Big|_{\epsilon=0}\right) \\
&\quad = -(\nabla_\theta O_y(X, \hat{\theta}) - \nabla_\theta O_y(X_{-ij}, \hat{\theta}))H_{\hat{\theta}}^{-1}(\nabla_\theta L(z_{-kl}, \hat{\theta}) - \nabla_\theta L(z, \hat{\theta}))
\end{aligned}
$$

C Derivation of TracIn^{+} and TracIn^{++}

Similar to IF(Koh and Liang, 2017) and TracIn(Pruthi et al., 2020b), we start from the Taylor expansion on point $\hat{\theta}_t$ around z' and z'_{-ij} as

$$
\begin{aligned}
\mathcal{L}(\hat{\theta}_{t+1}, z') &\sim \mathcal{L}(\hat{\theta}_t, z') + \nabla\mathcal{L}(\hat{\theta}_t, z')(\hat{\theta}_{t+1} - \hat{\theta}_t) \\
\mathcal{L}(\hat{\theta}_{t+1}, z'_{-ij}) &\sim \mathcal{L}(\hat{\theta}_t, z'_{-ij}) + \nabla\mathcal{L}(\hat{\theta}_t, z'_{-ij})(\hat{\theta}_{t+1} - \hat{\theta}_t)
\end{aligned}
$$

If SGD is assumed for optimization for simplicity, $(\hat{\theta}_{t+1} - \hat{\theta}_t) = \lambda\nabla\mathcal{L}(\hat{\theta}_t, z)$. Thus, putting it in above equations and perform subtraction, we obtain

$$
\mathcal{L}(\hat{\theta}_{t+1}, z') - \mathcal{L}(\hat{\theta}_{t+1}, z'_{-ij}) \sim \mathcal{L}(\hat{\theta}_t, z'_{-ij}) - \mathcal{L}(\hat{\theta}_t, z') + [\nabla\mathcal{L}(\hat{\theta}_t, z') - \nabla\mathcal{L}(\hat{\theta}_t, z'_{-ij})]\lambda\nabla\mathcal{L}(\hat{\theta}_t, z)
$$

And,

$$\text{imp}(x'_{ij}|z';\hat{\theta}_{t+1}) - \text{imp}(x'_{ij}|z';\hat{\theta}_t) \sim [\nabla\mathcal{L}(\hat{\theta}_t, z'_{-ij}) - \nabla\mathcal{L}(\hat{\theta}_t, z')]\lambda\nabla\mathcal{L}(\hat{\theta}_t, z)$$

So, the left term is the change of importance by parameter change; we can interpret it as the change of importance score of span x_{ij} w.r.t the parameter of networks. Then, we integrate over all the contributions from different points in the training process and obtain

$$\texttt{TracIn}^{+}(x'_{ij}|z', z) = \sum_t [\nabla\mathcal{L}(\hat{\theta}_t, z'_{-ij}) - \nabla\mathcal{L}(\hat{\theta}_t, z')]\lambda\nabla\mathcal{L}(\hat{\theta}_t, z)$$

The above formation is very similar to TracInwhere a single training instance z is evaluated as a whole. But we are interested in the case where an meaning unit x_{kl} in z can be evaluated for influence. Thus, we apply the same logic of the above equation to z_{-kl}, the perturbed training instance where token k to l is masked, as

$$\texttt{TracIn}^{+}(x'_{ij}|z', z_{-kl}) = \sum_t [\nabla\mathcal{L}(\hat{\theta}_t, z'_{-ij}) - \nabla\mathcal{L}(\hat{\theta}_t, z')]\lambda\nabla\mathcal{L}(\hat{\theta}_t, z_{-kl})$$

Then, the difference $\texttt{TracIn}^{+}(x'_{ij}|z', z) - \texttt{TracIn}^{+}(x'_{ij}|z', z_{-kl})$ can indicate how much impact a training span x_{kl} on test span x'_{ij}. Formally, the influence of x_{kl} on x'_{ij} is

$$\texttt{TracIn}^{++}(x'_{ij}, x_{-kl}|z', z) = \lambda \sum_t [\nabla\mathcal{L}(\hat{\theta}_t, z'_{-ij}) - \nabla\mathcal{L}(\hat{\theta}_t, z')][\nabla\mathcal{L}(\hat{\theta}_t, z) - \nabla\mathcal{L}(\hat{\theta}_t, z_{-kl})]$$

We denote that such a form is very easy to implement, since each item in summation requires only four (4) gradient estimates.