Explaining Text Matching on Neural Natural Language Inference

YOUNGWOO KIM, MYUNGHA JANG, AND JAMES ALLAN, University of Massachusetts Amherst

Natural language inference (NLI) is the task of detecting the existence of entailment or contradiction in a given sentence pair. Although NLI techniques could help numerous information retrieval tasks, most solutions for NLI are neural approaches whose lack of interpretability prohibits both straightforward integration and diagnosis for further improvement. We target the task of generating token-level explanations for NLI from a neural model. Many existing approaches for token-level explanation are either computationally costly or require additional annotations for training. In this paper, we first introduce a novel method for training an explanation generator that does not require additional human labels. Instead, the explanation generator is trained with the objective of predicting how the model's classification output will change when parts of the inputs are modified. Second, we propose to build an explanation generator in a multi-task learning setting along with the original NLI task so that the explanation generator can utilize the model's internal behavior. The experiment results suggest that the proposed explanation generator outperforms numerous strong baselines. In addition, our method does not require excessive additional computation at prediction time, which renders it an order of magnitude faster than the best-performing baseline.

CCS Concepts: • **Information systems** → *Clustering and classification*; • **Computing methodologies** → Information extraction; **Neural networks**.

Additional Key Words and Phrases: natural language inference, neural network explanation, rationale, interpretable machine learning

ACM Reference Format:
Youngwoo Kim, Myungha Jang, and James Allan. 2020. Explaining Text Matching on Neural Natural Language Inference. *ACM Transactions on Information Systems* 1, 1, Article 1 (January 2020), 23 pages. https://doi.org/10.1145/3418052

1 INTRODUCTION

Natural language inference (NLI) is the task of detecting the existence of entailment or contradiction in an input sentence pair [7]. The NLI task has been expected to assist many information retrieval (IR) tasks, as the notion of relevance is closely related to the notion of entailment [10, 31]. For example, in question answering applications, an NLI component could be used to retrieve semantically-equivalent questions or answer passages [21]. Once the NLI model determines a relationship between two sentences to be *entailment* or *contradiction*, downstream applications would then benefit from additional information about the decision: What makes the relationship between the sentences *entailment*? If the relationship is identified as *contradiction*, what is contradictory about the two

Author's address: Youngwoo Kim, Myungha Jang, and James Allan, University of Massachusetts Amherst, Amherst, MA 01003, {youngwookim,mhjang,allan}@cs.umass.edu.

1046-8188/2020/1-ART1 $15.00
https://doi.org/10.1145/3418052

sentences? To explain these, the entailed or contradictory information should be localized in given sentence pairs.

Existing work in other text classification task have been accompanied with a token-level explanation to explain models' predictions [4, 27, 28]. For NLI task, additional token-level information would be helpful in many downstream tasks such as the following applications:

- Entailment functions are adopted as a search component in retrieval-based question answering (QA) systems [21, 35] by retrieving questions or answers that are entailed by users' queries. In practice, the system would need to know which tokens in the candidate text entail the query, so that it can highlight the words that entail the query or re-rank the candidate texts based on the contextual information such as conversation history.
- For diversification and duplicate detection in short texts, entailment-based text clustering is used as an alternative to the conventional bag-of-word features [20]. Knowing which terms in short texts are actually entailed could be a used for a stronger similarity function for clustering.
- Identifying contradictory information at the token-level is important for contradiction-aware summarization systems, such as opinion summarizers [26] and summarizations for systematic reviews [1, 43].

In this work, we also formalize our challenge as a token-tagging problem. Specifically, we aim to explain the identified relation between input pairs by (1) identifying which tokens in each sentence can be semantically aligned across the pair of sentences, and (2) uncovering reasons for contradictions by identifying tokens that represent two pieces of information that cannot be true at the same time. For this, we define three types of tag: "match" to denote aligned tokens that convey the same information, "conflict" to denote tokens that present contradictory information, and "mismatch" to denote tokens that could not be aligned (Table 1).

There are number of approaches that are applicable to our problem. First, generic neural network explanation methods could be used to score the importance of input tokens [4, 38, 42, 51, 52]. In these methods, importance scores are typically calculated by one of the following three types of signals or combination of them: (1) changes in outputs with respect to input perturbations [38], (2) gradients of the outputs with respect to the inputs [42, 51, 52], and (3) activated weights in the neural network [5]. While these methods do not incur additional annotation effort, we found that many of them are too computationally inefficient to be used in product deployment settings. Moreover, most methods are only investigated in very generic settings that are not specific to a particular task or architecture, and thus there should be much room to improve the accuracy by specializing on the particular problem and models.

Secondly, when it comes to NLI task specific approach, Thorne et al.'s work is most closely related to our work as they also targeted token-level explanation for NLI [44]. They proposed an unsupervised explanation model based on attention weights. However, it was shown that their attention-based

Table 1. Three example sentence pairs from MNLI dataset with the corresponding classification labels (entailment, contradiction and neutral) and token-level tags : conflict (red), match (blue) and mismatch (yellow). In each row, the text on the left corresponds to the premise and the text on the right corresponds to the hypothesis.

There is nothing more to be done here, I think, unless, at the dead ashes in the grate.	entailment	he stared There isn't anything left to do.
yeah i mean just when uh the they military paid for her education	contradiction	The military didn't pay for her education.
uh-huh well I've enjoyed talking to you.	neutral	I liked talking to you about sports.

method is not as effective as generic method LIME [38], which only uses model's output changes from perturbed inputs to generate an explanation. This result is supported by other studies that show standard attention modules do not provide meaningful explanations [25]. One possibility is that we may need more complex mechanism to generate an explanation from the model's hidden variables such as attention weights.

In this study, we aim to build an explanation generator to generate token-level explanations by considering the model's internal behaviors. We specifically address the following research questions:

RQ1: How can we train an effective token-level explanation generator without annotation effort?
RQ2: How can an explanation generator benefit from the model's hidden variables?

As an answer to the first research question, we propose a novel weakly-supervised method to train the explanation task without using any additional human-labeled data. The training objective of the explanation generator is to predict the model's output changes in response to perturbations. Specifically, given a sentence pair, a number of tokens are randomly removed and the changes in the classification outcomes are measured. Tokens that cause larger changes in the classification when removed are taken as a weak supervision signal, and the explanation generator is trained to generate scores that can predict the model's behavior in response to perturbations. The resulting generator can then be used as a token-level explanation generator.

Regarding the second research question, we show that explanation performance can be improved by building a multi-task learning model that simultaneous predict both the original NLI classification and its explanation. Multi-task learning has been shown to be effective for tasks with similar characteristics. In this explanation problem, multi-task learning provides significant benefits because the explanation generator can access most of the hidden variables of the original task network.

We apply our approach to the Multi-Genre NLI Corpus (MNLI) [47] and the Stanford Natural Language Inference (SNLI) Corpus [7]. The methods are evaluated by comparing the model's outputs with human-annotated token-level explanations. For the MNLI dataset [47], we collect human-annotated token-level explanations based on our definition of the three types of tag (section 3). For the SNLI dataset, we used the token-level annotation collected by Camburu et al. [8]. Overall, our paper makes the following contributions:

- We introduce a weak-supervision training method to train an explanation generator for the classification problems.
- We show that training an explanation generator using multi-task learning with the original task network improves the explanation quality even beyond the original perturbation signal.
- We introduce token-label tags and collection labels to explain the NLI problem. We describe the performance of various explanation approaches on the proposed dataset and the existing explanation dataset. The experiments on both datasets showed our method to be not only more computationally efficient than perturbation methods, but also more precise than a number of strong baselines.

2 RELATED WORK

2.1 Explaining natural language inference

Among the work aimed at obtaining explanations for NLI models, one notable example is the e-SNLI dataset [8], which adds large scale explanation annotations to the well-known NLI dataset SNLI [7]. It contains human-written explanation sentences and token-level annotations that represent the important tokens for the decision [8]. They reported that it is challenging to create quality explanations and evaluate generated sentences, because it is not easy come up with clear criteria to define a good explanation. For a method, they also proposed to train a neural network that generates explanation sentences. Using e-SNLI, [44] have investigated whether the attention component of the

neural network can be used to generate token-level explanations [44]. The results were not positive: they showed that the explanation score derived from the attention score is less effective than the generic machine learning explanation method LIME [38]. In this paper, we show our approach is more precise than LIME and the attention component based approach.

2.2 Neural network explanation methods

Certain neural explanation methods are called explanation generators if they incorporate additional machine learning components beyond the original models and have their own parameters to be trained [18]. A number of explanation generation approaches have used human-written explanations as supervised labels [8, 22, 24]. If the target data are accompanied by explanations such as textual descriptions of images, supervised approaches can be an effective and easy way to build an explanation generator [22]. However, it is less likely for the text data to be accompanied by additional text explaining its meaning. Moreover, some tasks, including NLI, are hard for crowd-workers to annotate, because these tasks concern too primitive levels of textual understanding. Annotating explanations at the token-level could be much more challenging and costly than the original task.

In contrast, other neural network explanation approaches aim to be more generic and unsupervised. To provide an explanation for a particular instance, these methods assign an importance or salience score to the input by examining the current gradients or the currently active weights, which are mostly decided by activation of non-linearity units. We refer to these approaches as *gradient-driven methods*. Initially, the gradient from the input to output was used as the importance score [41]. Later approaches used complex combinations of the gradient at multiple points [42]. Another strategy of these instance-wise approaches is to hide part of the input and measure the sensitivity of the output changes [51, 52]. The layerwise relevance propagation (LRP) method explains the contribution of the input tokens by recursively distributing the contribution of an upper layer's neurons to the lower layer's neurons based on the weights at the particular input instance [6]. This method has been employed to explain a number of text classification problems – for example, sentiment analysis using recurrent neural network [5] and document classification using convolutional neural networks [4]. It shows which input words are important for a particular word-generation or classification decision. Ancona et al. have compared a number of these attribution approaches [2].

One potential pitfall of gradient-driven methods is that they may not be reliable outside the small faithful locality. Many methods only examine the gradients (or weights) at single inputs, which makes it challenging to capture a larger view. For example, we found that the impact of negation such as "not" is often underestimated, and yet its deletion may change the classification decision from entailment to contradiction. Our approach and other explanation generation approaches are more robust in handling this problem compared to gradient-driven methods, as they are trained to generate larger locality during training.

One easy way to arrive at a larger view of neural network behavior is to change part of the input and examine the changes to the output and the network [42, 51]. Such approaches are referred to as perturbation-based approaches [14]. A major drawback of perturbation-based approaches is computational cost. Moreover, the effect of removing multiple tokens simultaneously might be very different from the effect if they are removed independently. In addition to the cost of executing exponential permutation candidates, translating the permutation behavior into a localization decision is not trivial. In this paper, we suggest to use perturbations to train an explanation generator, so that we can obtain explanations with lower computational costs at the inference time with the acceptable increase of the cost in the training time.

Specifying rationales as part of the input has been used for a number of text classification tasks [9, 27, 28]. Those approaches are similar to our approach in that an explanation generator is trained by selecting an important subset of the input that would result in similar decisions. Our

experiments do not have direct comparison with these approaches, because applying them for the NLI task was unduly complex. In contrast to those approaches, which use deleting part of the input as signal, our approach is more flexible, because it can be trained with other types of modification of the inputs.

For our problem, some explanation methods are difficult to apply. Model-wise explanation approaches' strategy is to generate simpler proxy models, which can be decision-trees [12] or sets of rules [3]. Although proxy models of networks are easier to interpret than the original models, it is not trivial to use them for instance-wise explanation, or to handle word embeddings. Some other approaches are to ensure models to be interpretable by forcing the models to obey some constraints [32, 39]. However, it is unclear whether the prevalent neural network components such as a sentence encoding from a sequence of word embeddings can be achieved with the proposed strategies without sacrificing performance.

2.3 Evaluating explanations for neural network

Although there are numerous generic approaches to explain the predictions of a neural network, it is difficult to declare one method superior to another because the methods were compared with different evaluations and it is difficult to define a standard for the evaluations. One of the popular evaluation methods is to delete or hide a part of the inputs that is explained as contributing significantly to the prediction and check whether the output changes more than if other parts had been deleted or hidden [2, 4]. Other approaches incorporate human into evaluation, where the most intuitive way is to compare human written explanation with the model's explanation [8, 28]. It is possible to ask people to achieve some goals by using model provided explanations. In Ribeiro et al.'s work, people were shown explanation examples for two classifiers and they were asked to predict which classifiers would perform better or which features would be good for classification [38]. In our evaluation, we take the most common approach to compare human written explanation with model's explanation.

2.4 Natural language inference

While there has a been long history of research in area of textual entailment, recently used NLI task definitions are largely affected by SNLI dataset, which is a large dataset with 570,000 pairs with manual annotations[7]. It was followed by more generalized dataset Multi-Genre NLI Corpus (MNLI), which is one of the most representative dataset for NLI task. Currently, the best approach for NLI task is to fine-tune the neural model from pre-trained language models such as BERT [13, 50]. Recent improvements on MNLI tasks are done by improving pre-trained language models, mostly by increasing the model size and training data for pre-training [36, 37]. The first pre-trained language model on Transformer showed accuracy of 82.1. Soon Devlin et al. showed their BERT_{BASE} model and BERT_{LARGE} model with accuracy of 84.6 and 86.6, where only difference is the size of the model [13]. XLNet followed with accuracy of 89.8 [50] and T5-11B showed 91.7 [37].

Unfortunately, it is a non-trivial challenge to apply them to tasks that are similar but slightly different from one in which they were trained. To benefit from the NLI corpus and the proposed approaches, various transfer learning approaches were proposed [11, 34]. For example, an MNLI-trained model was proposed as a sentence encoder to improve other tasks [11]. Multi-task learning is another effective way to benefit from a large corpus such as the MNLI dataset. Liu et al. built a multi-task model by training many natural language understanding tasks and sharing all intermediate transformer parameters except the last feed-forward layers [29]. Especially, the accuracy on the task with small datasets have huge improvements. Limitations of such transfer learning approaches are that they require additional training data for the target task and they are difficult to be used in applications that are not based on machine learning.

Table 2. All relevant tags displayed. Blue is for match, red is for conflict, and yellow is for mismatch. In each row, the text on the left corresponds to the premise and the text on the right corresponds to the hypothesis.

An older and younger man smiling.	neutral	Two men are smiling and laughing at the cats playing on the floor.
yeah i mean just when uh the they military paid for her education	contradiction	The military didn't pay for her education.

To see how the NLI task can benefit other tasks, it worth to have a look at the methods before the neural approaches appear. Earlier methods for NLI can be found in the PASCAL Recognizing Textual Entailment (RTE) challenge [17, 23]. Many entailment systems in RTE challenges were based on curated lexical resources such as WordNet [33] and ParaphraseDB [16]. They were often composed of separable modules, and it was reasonably clear what the role of each modules was and the systems could be applied to similar tasks [21].

One example of textual entailment systems that is less reliant on machine learning techniques is the Excitement Open Platform [30]. This system provides a textual entailment engine based on the edit distance between premise and hypothesis. This system was used as a feature for a semantic similarity task [46] and a relation extraction task [15]. The system was easy to be utilized because it reveals token alignment information and does not require additional training data for the downstream task.

3 TASK DEFINITION

The original natural language inference task is a sentence pair classification problem. Two sentences, a *premise* and a *hypothesis*, are given. The goal of the task is to classify their relationship into either *entailment*, *neutral*, or *contradiction* [7].

We define NLI explanation as a sequence-tagging problem. To provide a clear definition for the token-level annotation, we defined three tags, each of which indicates the role of the tokens in the sentences with regard to the inference decision. Given a pair of input sentences, our goal is to compute a score for each token in the sentences based on how relevant it is to each tag: *match*, *mismatch*, or *conflict*.

Match. The **match** tag in the hypothesis denotes a token whose meaning can be inferred from the premise. A token in the premise is tagged as *match* if it is required to infer the meaning that appears in hypothesis. A sentence pair that is labeled as an entailment implies that all the meanings that the tokens in the hypothesis imply should be inferred from tokens in the premise. Thus, we expect all tokens in the hypothesis to be tagged as *match* and some of the tokens in the premise – those that correspond to the tokens in the hypothesis – to be tagged as *match*. As a result, many tokens are tagged as *match* including ones that are trivially same across the sentences.

Even if a sentence pair is labeled as neutral, some information in the hypothesis could be inferred from the premise. In the first example of Table 2 (labelled neutral), the text "Two men are smiling" in the hypothesis can still be inferred from the premise, so we can annotate these tokens as *match* even though the classification label is not *entailment*. Similarly *match* can be used for a sentence pair whose label is *contradiction*.

Mismatch. A token is tagged **mismatch** if it is in the hypothesis but cannot be inferred from the premise. For example, in Table 1, "about sports" in the third row cannot be inferred from the premise, and hence is annotated as *mismatch*. A neutral relationship can be clearly explained by indicating

which tokens are considered mismatched. *Mismatch* is the opposite of *match*. We distinguish them by considering *mismatch* only for cases whose classification labels are neutral.

Conflict. A token is tagged **conflict** if it is a critical token that renders the corresponding concept untrue. We chose to apply the *conflict* tag only to critical tokens that produce a contradiction, rather than tagging all the tokens of a contradictory concept. In the annotation process, the annotators were instructed not to include tokens that are trivially identical across sentences. The negations and antonym pairs that are relevant to a contradiction are always included.

Drawing a clear border for contradicting concepts is a difficult problem due to their ambiguity. The ambiguity was more apparent for conflicts than it was for matches or mismatches. In the first example of Table 3, all the annotators included "same" and "differ" as conflict tokens, but some also included "is the" and "tended to" as conflict tokens. We accept this discrepancy as an inevitable limitation of token-level explanations.

Our tagging definition differs from that of e-SNLI [8, 27] in that we are not only seeking to identify important tokens, but also to identify the roles of the tokens. e-SNLI was generated by asking crowd-workers to write explanation sentences for each instance of the SNLI dataset. During the annotation process the crowd-workers were also asked to tag which tokens are important for the explanation. The difference between their guideline and ours is particularly prominent for the entailment sentence pairs. In this work, entailment sentences are explained by annotating which tokens would be tagged *match* with the definition above. In contrast, e-SNLI asked the crowd-workers to tag tokens that are semantically related but not trivially equal across the sentence pairs (these are often paraphrased texts or a hyponymy and hypernymy pairs). This difference resulted in our dataset having more tokens tagged for entailment sentences pairs then e-SNLI dataset.

4 METHOD

We model token tagging as a ranking problem. For a given sentence pair, the model generates three scores for each token. Each of the scores represents how likely the token is to be tagged as match, mismatch, and conflict. Note that the three tags are not mutually exclusive. Although it is reasonable to expect correlations between tags in a single token, modeling some relations – e.g., between conflict and match – could be very subtle sometimes. Thus, we don't explicitly capture correlation in our modeling.

Let $f : x \rightarrow z_c$ be the original classification function implemented by the neural network, where x is a sequence of token IDs that are fed to the network. Another function $g : x \rightarrow y_t$ is added to generate an explanation vector y_t for a tag t, where the number of dimensions for y_t is equal to number of tokens in x. Our goal is to train g, so that the score of the $y_{t,i}$ (i-th element of y_t) indicates how likely the corresponding input token x_i should be tagged with the particular tag.

Table 3. Two examples showing disagreements between annotators. Each example shows annotations from two annotators, where one annotator tagged only the deep red colored tokens as conflict while the other annotator also included the light red colored tokens as conflict.

However, the specific approaches to executing those principles tended to differ among the various sectors.	contradiction	Specific approaches to each principle is the same in each sector.
What you say about Lawrence is a great surprise to me , I said.	contradiction	I knew that about Lawrence all along.

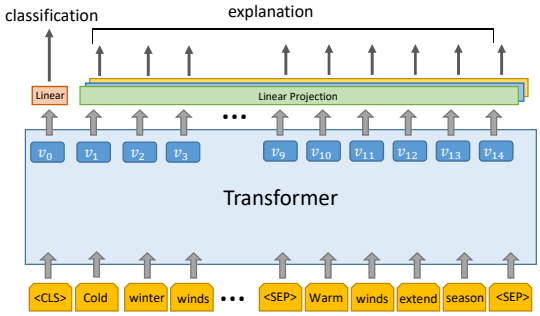

Fig. 1. The structure of our Transformer-based model. Thin arrows represent final outputs for classifications and explanations. The red "Linear" in the left represents the linear-projection layer for sentence pair-level classification, and three linear-projection layers on the right are used to generate the explanation scores for each tags. Linear layers with the same color share the same parameter.

4.1 Transformer network for classification

Transformer-based neural networks [45] have been shown to be effective for NLI problems, especially with language model pre-training [13]. We propose a multi-task learning model that can be built on the Transformer structure. In our experiment, we used pre-trained BERT [13], though our explanation method is applicable to other Transformer based models as well.

Figure 1 shows the structure of our model. The model takes as input a sequence of token ids, which will be transformed into dense word embedding in the beginning of the network. It has multiple blocks, each of which has multi-head attention layers and feed-forward layers as components. The Transformer generates an h-sized vector, v_i, as a representation for each token, where i is the index of the token in the sequence.

For the classification task, we use the same approach used in previous work [13]. The input sequence is represented as "[CLS] premise tokens [SEP] hypothesis tokens [SEP]", where [CLS] and [SEP] are special tokens. For each token i, the Transformer outputs a vector $v_i \in \mathbb{R}^h$. To produce classification probabilities z_c for three classes (entailment, neutral, and contradiction), the output vector for the [CLS] token, v_0 is passed to a single linear projection layer and a softmax layer.

$$z_c = W_c^T v_0 + b_c \tag{1}$$

For other tokens, the corresponding output vectors v_i are used to obtain explanation scores. The vectors are fed into another three different linear projection layers. Finally, we obtain the explanation score $y_{t,i}$, which is the score of the i-th token for tag t

$$y_{t,i} = W_t^T v_i + b_t. \tag{2}$$

Here, we use two different loss functions for training, so the final output has slightly different format. When using cross-entropy loss, each linear projection layer produces two values from each token. The two values are fed into a softmax layer to obtain probability scores, which is same as in binary classification. The one of the two probability scores is used as the final score. As we have three linear projection layers, we obtain three scores for each token in the sequence. Each of three scores represents the probability that the corresponding token be tagged as a particular class. When using a correlation loss function, each linear projection layer produces a single value for each token. As there are three linear projection layers, we again obtain three scores for each of the tokens.

4.2 Training to explain

For each sentence pair in the training data, we select a weak supervision label for each tag and train the explanation generation with it. First, random perturbations (deleting) are applied to the sentence pair to generate several perturbed inputs. The generated inputs and the original input are then fed into the classification network to obtain the classification probability for the perturbed input. For each of the tags, we select the perturbation that resulted in the most *informative* output changes compared to the output from the original input. We use tokens that were modified in the select perturbation as a weak label to train the network. Figure 2 shows a high-level overview of the explanation training. In this section, we describe the details of each step of this training.

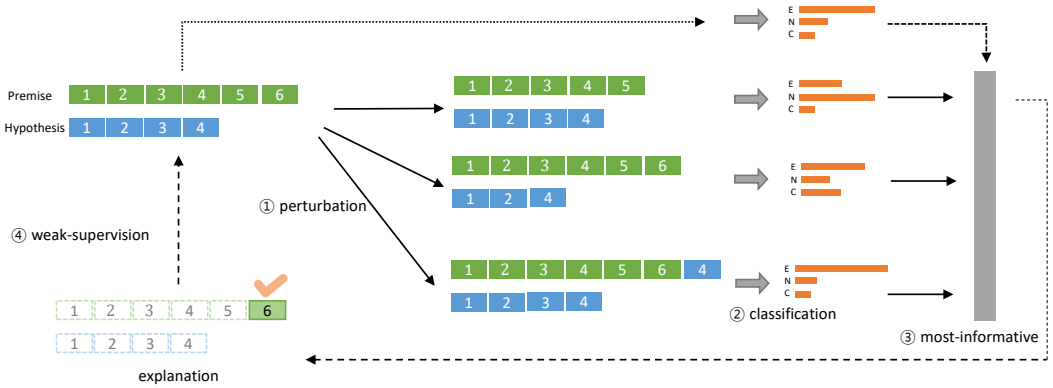

Fig. 2. Explanation training procedure with weak-supervision. This diagram shows the procedure to obtain a signal for one of the tags.

Algorithm 1: Selecting Informative Instance

input : Input text sequence $\mathbf{x}=\{x_1, x_2, ..., x_n\}$
Three signal functions $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ for each of tags. Each $\mathcal{S}_t$ is defined in terms of f
(hyper-parameter) m : the number of perturbed inputs
(hyper-parameter) p : parameter for perturbed sequence length

Output: Most informative instances $x^{(s_t)}$ for each tag t

for $i \leftarrow 1$ **to** m **do**
  L : Location of the last non-padding token. $j \leftarrow$ Sample from $\{0, 1, ..., L\}$
  $l_i \leftarrow$ Sample from $G(p = 0.5)$
  `// Delete tokens from` j `until` $j + l_i$
  $x^{(i)} \leftarrow \{x_1, x_2, ...x_{j-1}, x_{j+l_i}, ..., x_n\}$;

Evaluate signal functions $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ for $x, x^{(1)}, ...x^{(m)}$

for *tag* $t \leftarrow 1$ **to** 3 **do**
  for $i \leftarrow 1$ **to** m **do**
    `// Length penalty`
    $\mathcal{D}(x, x^{(i)}) \leftarrow max\{0.1 \cdot (l_i - 3), 0\}$
    `// Informative score` $\mathcal{I}_i$
    $\mathcal{I}_i^{(t)} \leftarrow \mathcal{S}_t(x) - \mathcal{S}_t(x^{(i)}) - \mathcal{D}(x, x^{(i)})$
  $s_t = \arg\max_i \mathcal{I}_i^{(t)}$

return $x^{(s_{t_1})}$, $x^{(s_{t_2})}$, $x^{(s_{t_3})}$

Perturbations are applied to each of input sentence pairs by the following procedures: (1) a token index j is randomly selected to start the deletion; (2) from a geometric distribution with $p = 0.5$, the length of the sequence to be deleted, $l \sim G(p = 0.5)$, is sampled; and, (3) l tokens from location j to $j+l$ are deleted. The tokens after the deleted tokens are shifted forward and the end of the sequence is filled with padding tokens. Multiple perturbed instances are generated from a single training instance in this way. Here, the tokens come from a concatenation of the premise and hypothesis, so that both the premise and hypothesis have a chance to be perturbed.

From each input instance x (indices of the sentence pair), a set of perturbed instances $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$ is generated. These perturbed instances are fed into the network to produce a corresponding classification probability (softmax) output.

Among the perturbed instances, we want to **select** the one that changes the output the most in a way that we are interested in. We will call such instance as the **most informative instance**. We define and measure the degree of informativeness by what we refer to as a signal function. We expect that modeling each of the tags separately would help to represent the different aspects of the textual understanding (match, mismatch and conflict). For each target tag, we define a corresponding signal function $\mathcal{S}_L$ as follows:

$$\begin{aligned} \mathcal{S}_{match}(x) &= f_e(x) \\ \mathcal{S}_{conflict}(x) &= f_c(x) \\ \mathcal{S}_{mismatch}(x) &= f_n(x), \end{aligned} \tag{3}$$

where f_e, f_c, and f_n are softmax probability outputs for entailment, contradiction and neutral, respectively.

Because long sequences of deleted tokens are likely to result in larger output changes that are less meaningful, we penalize any perturbation with a large number of token changes by introducing a size penalty:

$$\mathcal{D}(x, x^{(k)}) = \max\{0.1 \cdot (d - 3), 0\}, \tag{4}$$

where d is the number of modified tokens between x and $x^{(k)}$. Thus, if two different perturbations cause similar changes in the signal function, the shorter one would be preferred. The numbers in the penalty term were heuristically designed to match the scale of $\mathcal{S}_t(x) - \mathcal{S}_t(x^{(k)})$ which is in the range [-1, 1].

Equation 5 shows the final informative score for each perturbed instance $x^{(k)}$, where $\mathcal{S}_t$ is one of the signal functions in Equation 3:

$$\mathcal{I}_t(x, x^{(k)}) = \mathcal{S}_t(x) - \mathcal{S}_t(x^{(k)}) - \mathcal{D}(x, x^{(k)}). \tag{5}$$

For each tag t, the most informative instance $\hat{x}$ is selected from the perturbed instances:

$$\hat{x}_{(t)} = \underset{k \in [1,m]}{\arg\max}\, \mathcal{I}_t(x, x^{(k)}), \tag{6}$$

where m is the number of perturbed instances. As a result, we obtain three instances, one for each of the tags. For some input instances, it is possible that even the largest change of the signal function (the model's output) is very small. For example, consider the case where a sentence pair is classified as entailment and there is very low probability for contradiction. It is possible that any deletion does not change the contradiction probability much. In this case, the most informative instance for conflict would not be meaningful enough as the magnitude of change is very small. We handle this problem by setting a minimum threshold on the informative score, so that the instance for the particular tag is rejected and the training is skipped when the most informative perturbation is not of adequate quality.

We selected a threshold of 0.3, because this is approximately the average value for the probability of each label (three probabilities that total 1).

Finally, **weak-supervision** signal is decided by the most informative instance $\hat{x}_{(t)}$. We take the tokens that were modified from x to $\hat{x}_{(t)}$ as the weak label for the tag t. We denote the weak label for tag t as $\hat{y}_t$.

Let y_t be a vector representing the model output where each dimension $y_{t,j}$ is the score for j-th token to be important for the tag t. The size of y_t equals the number of tokens in the input sequence. Given a weak supervision label $\hat{y}_t$, the cross-entropy loss for each tag t is given as

$$\mathcal{L}_t = -\sum_j \hat{y}_{(t),j} \log y_{t,j}, \tag{7}$$

where the label $\hat{y_{t,j}}$ is 1 if j-th token was modified in the perturbation and 0 otherwise. Our final loss is sum of the loss for each tag

$$\mathcal{L} = \sum_t \mathcal{L}_t. \tag{8}$$

As an alternative to classical cross-entropy function, we suggest using Pearson's correlation coefficient as a loss function. The loss function that we refer to as correlation loss is given as

$$\mathcal{L}_t = -\frac{\sum_j (\hat{y}_{(t),j} - \bar{\hat{y}}_{(t)})(y_{(t),j} - \bar{y_t})}{\sigma_{\hat{y}_t}\sigma_{y_t}}, \tag{9}$$

where the label $\hat{y_{t,j}}$is 1 if j-th token was modified and -1 otherwise [1]. σ_{y_t} and $\bar{y_t}$ are the standard deviation and the mean of the values in vector y_t

$$\bar{y_t} = \frac{\sum_j y_{t,j}}{|y_t|} \tag{10}$$

$$\sigma_{y_t} = \sqrt{\frac{\sum_j (y_{t,j} - \bar{y_t})^2}{|y_t| - 1}}. \tag{11}$$

$|y_t|$ is the size of the explanation vector, which is equal to the maximum sequence length. $\sigma_{\hat{y}_t}$ and $\bar{\hat{y}}_t$ are defined similarly for the vector $\hat{y}_t$. We adopted this loss function as we expect this could be more robust than cross-entropy loss with noisy signal. This correlation loss function satisfies the conditions for an effective list-wise loss function [48]. The effect of the loss function is discussed in section 5.3.

5 EXPERIMENTS

We evaluated our method and the baseline approaches by comparing them against human annotated sequence tagging. Our experiments were mainly conducted on the model trained on the MNLI dataset [47], which we annotated based on the definition in section 3. For comparison with previous work, we also conducted the experiment on the e-SNLI dataset [8]. The annotation definition of the e-SNLI is slightly different from ours, because they did not explicitly define the role of the tokens as match, mismatch or conflict. We observed that most methods were applicable to both the e-SNLI data and our dataset.

5.0.1 Implementation. Currently, most state-of-the-art models for the NLI tasks are built by fine-tuning a pre-trained language model [29, 37]. We used the pre-trained uncased BERT model with 12 layers and fine-tuned the entire network.

We first trained 2.5 epochs only for the NLI classification task. We then began training both the classification and explanation modules. We alternately processed classification training steps and

[1] Using 1 and 0 would be effectively the same

explanation training steps. Both the classification and explanation was trained only on the training split. The explanation training lasted for 0.5 epochs, which is roughly 12,000 steps with a batch size of 16. For each training instance, 20 perturbed inputs were generated, from which the most informative pair were selected. Our training was done with single M40 GPU. The training with explanation took roughly 33 hours to be trained. The training without explanation took 21 hours to be trained.

A parameter update was performed using the Adam optimizer with weight decay. Linear decay of the learning rate and warm-up steps were applied as they are in the original implementation of the BERT model. For the initial learning rate we used $2 \cdot 10^{-5}$. The maximum sequence length was set to 300 tokens.

During the explanation training we also trained the classification module by alternating the two tasks every step. For the explanation training, we used a smaller learning rate than we did for the classification training (0.3 times the learning rate for the classification training).

5.1 Evaluation

5.1.1 Metrics. The metrics we used in the evaluation were accuracy, mean average precision (MAP), and precision at 1 (P@1). To evaluate accuracy, we tuned the cut-off threshold on the development set to maximize accuracy. Accuracy was measured over all tokens in the test set. MAP and P@1 do not require add cut-off threshold.

5.1.2 Data annotation. *Conflict* was labeled only for sentence pairs whose gold label was contradiction. Similarly, *match* was labeled for the sentence pairs with entailment label and *mismatch* for neutral label.

Because the "entailment" label implies that the content of the hypothesis can be inferred from premise, if the label is "entailment", all tokens in the hypothesis should be labeled *match*. Thus, we evaluated the *match* label only on the premise sentences. For *mismatch*, we evaluated tokens only in the hypothesis.

Forty percent of the data were annotated by three annotators. When the annotators produced different annotations, the annotation that is more similar to the others was selected. Thus if two annotators made similar decisions and the other made a different decision, one of the two similar decisions was selected.

From the validation split of MNLI, we annotated 700 instances for each tag, resulting in a total of 2,100 instances. For each tag, 100 instances were used as a development set, and 600 instances were used as a test set. Kohen's κ for token-level agreement was 0.74.

5.1.3 Baselines. To show the characteristics of the dataset with trivial baselines, we included random and inverse document frequency (Idf) approaches. The random method assigns a random score to each token. The Idf method assigns each token a score of $(1/df)$ where df is the number of sentences in the collection that contain the corresponding word. P@1 of the random method is approximately the proportion of true label.

LIME [38] is a generic classifier explanation method that has been shown to be the best-performing method in previous work on the e-SNLI dataset [44]. Given an input, the LIME method generates numerous perturbed variations of the input. It evaluates the model's outputs for these variations and builds a linear classifier that can predict the model's output near the given point. This method requires a large number of perturbed instances for each input. For the number of perturbed inputs, we selected the proposed value from the implementation.[2]

[2]https://github.com/marcotcr/lime.

We considered three gradient-driven approaches: Saliency [41], Grad*Input [40], and Integrated Gradient (IntGrad) [42]. The Saliency method evaluates the score of each input as the absolute value of the input's gradient toward the output. The Grad*Input method obtains the score by multiplying each input dimension by the gradient. The IntGrad method evaluates the score by numeric integration of the gradient over the input changes from starting value to current input value. These three methods were implemented based on the DeepExplain library [2].[3] Modifications were applied to each method to support word embedding.

Saliency, Grad*Input, IntGrad and LIME were designed to generate scores for each dimension of the input. As our explanation is token level, we sum the score for each dimension of the token's embedding. Taking a maximum was also considered, but the results from the development data showed that the maximum is similar to or worse than the sum.

We used two perturbation-based methods. The *Sensitivity* method assigns each token a score according to the change in the output when the token is deleted [51]. *Sensitivity (M)* deletes multiple tokens simultaneously. As it is infeasible to try all possible deletions, this method samples the location and length of the sequence to delete. Each token's score is assigned by the maximum change of outputs among the attempted deletions. For comparison, we allowed an equal number of runs for Sensitivity and Sensitivity (M).

As our model uses sub-word tokens, the scores of sub-word tokens were translated into a token-level score by taking the maximum of each token's sub-word tokens' scores.

5.2 Results

We refer to our method as **SE-NLI** (Self-Explaining NLI). In Tables 4 and 5, SE-NLI (CO) and SE-NLI (CE) denote our methods with different loss functions: Pearson's correlation coefficient (Equation 9) and cross-entropy loss (Equation 7), respectively. In the remaining parts of this paper, SE-NLI without any notation refers to SE-NLI (CO).

5.2.1 Performance on original NLI task. In this subsection, we demonstrate the performance of our model on the original NLI classification task to show that our multi-task learning for explanation approach does not have negative effect on the performance in the original task. As discussed in the related work (section 2.4), recent improvement in the NLI task has been mostly driven by improved language model pre-training. Thus, newer models are not particularly different from the perspective of the NLI task itself. Following existing work [19, 49], we include a comparison of models trained from the same BERT_{BASE} checkpoint. Table 4 shows the accuracy of the classification-only model and our multi-task trained models on the MNLI dataset, all having the same BERT_{BASE} as a starting point. The models show little difference in the classification.

Table 4. Original NLI task (entailment, contradiction and neutral) accuracy of the models trained with our explanation generator and the model that was only trained for the classification task. All three models used the same BERT_{base} model for parameter initialization. The numbers are accuracy on MNLI-matched split. The accuracy difference between runs 1, 2, and 3 are not significant, showing P-values of 0.60 (1 vs 2), 0.41 (1 vs 3) and 0.19 (2 vs 3).

	Model	Accuracy
1	Classification only	84.4
2	SE-NLI (CO)	84.5
3	SE-NLI (CE)	84.2

[3] https://github.com/marcoancona/DeepExplain.

Table 5. Experiment on token-level tagging done on MNLI. For each column the highest value is marked with bold text. If the highest value is significantly better than all the other methods it is marked with ▲(p = 0.01). Average # of Runs represents the number of neural network required to explain a single instance.

	Conflict			Match			Mismatch			Avg
Method	P@1	MAP	Acc	P@1	MAP	Acc	P@1	MAP	Acc	#Runs
Random	0.289	0.431	0.762	0.593	0.673	0.509	0.537	0.623	0.519	-
Idf	0.364	0.504	0.762	0.703	0.710	0.508	0.478	0.609	0.517	-
Saliency	0.705	**0.733**	0.762	0.813	0.793	0.524	0.798	0.761	0.530	1
Grad*Input	0.426	0.486	0.761	0.737	0.703	0.507	0.598	0.639	0.523	1
IntGrad	0.559	0.582	0.786	0.868	0.744	0.506	0.652	0.689	0.539	300
LIME	0.637	0.618	0.799	0.905	0.777	0.597	0.735	0.731	0.601	5,000
Sensitivity	0.601	0.598	0.780	0.950	0.795	0.590	0.653	0.674	0.542	39.8
Sensitivity (M)	0.398	0.520	0.762	0.658	0.728	0.508	0.723	0.764	0.523	39.8
SE-NLI (CO)	**0.750**▲	0.723	**0.800**	**0.965**	**0.903**▲	**0.760**▲	**0.817**	**0.830**▲	**0.714**▲	1
SE-NLI (CE)	0.551	0.599	0.783	0.932	0.874	0.739	0.803	0.803	0.657	1

Table 6. Comparison of our method with the reported best methods on e-SNLI dataset. For Thresholded Attention and LIME, the numbers are as presented in the previous work [44]. Our own experiments on LIME on BERT based model showed similar numbers to the previous work on LIME. For the comparison we used the same metric as the previous work.

	Premise			Hypothesis		
	P	R	F1	P	R	F1
Thresholded Attention	0.192	0.262	0.222	0.534	0.630	0.578
LIME (LSTM+GloVe based)	0.656	0.483	0.537	0.570	0.669	0.616
LIME (BERT based)	0.376	1.000	0.547	0.460	0.834	0.593
SE-NLI	0.525	0.726	0.609	0.492	1.000	0.660

5.2.2 Comparison with alternative explanation methods. Table 5 shows the results of the token-level explanation tagging conducted on MNLI. In most cases, SE-NLI (CO) is the best-performing method. The cross entropy version, SE-NLI (CE) is often comparable to the other methods, but it does not perform as well as SE-NLI (CO). It is surprising to find that SE-NLI performs much better than Sensitivity and Sensitivity (seq), because SE-NLI was trained on signals that are similar to those methods. Note that none of the methods were supervised with the explanation annotation. Each tag shows different levels of difficulty mainly due to the different number of positive labels in a single sentence pair. *Match* has the most positive tokens, which resulted in P@1 and MAP being higher than for the other two tags. *Conflict* is the most difficult of all tags.

Among the other methods, Saliency, Sensitivity, and LIME tend to perform better than the other baselines, but none of them performs exceptionally. It is noteworthy that the Saliency method is among the highest performing methods; it has the simplest implementation and the lowest computational cost.

Comparison with the previous work [44] on the e-SNLI dataset is shown in Table 6. The Thresholded Attention method uses attention weights to generate an explanation. We thresholded all the models to maximize the F1 score. Thus, the differences of precision and recall are the results of threshold selection. The scores for Thresholded Attention are from the model built using LSTM and GloVe embeddings. As this model did not benefit from pre-trained contextualized embeddings, such as BERT, it is not directly comparable to SE-NLI. Instead, LIME could be a baseline for the

comparison, as our implementation of LIME on BERT showed similar results to the LIME on an LSTM and GloVe based model. On the e-SNLI dataset, SE-NLI out-performed the LIME method by F1. We would not expect Thresholded Attention to be better than SE-NLI, considering that many studies claimed that attention weight alone is insufficient as an explanation [25].

5.2.3 Computational requirements. Table 5 shows on the right the average number of neural network runs required for each method. Saliency and Grad*Input need to compute one forward run and one backward run (gradients to input) to compute the token-level scores. The IntGrad method uses numeric integration over the multiple points of gradients and outputs, and so it requires a large number of computations: the default parameter from the implementation is 300. The LIME method requires many outputs of perturbed inputs to build a linear classifier: 5,000 is also from the default parameter of the implementation. The Sensitivity method deletes each token in the input one by one, and 39.8 is the average number of tokens in the evaluation data. We did not count forward runs and backward runs as separate runs if they used the same input. Saliency, Grad*Input, and IntGrad require both forward runs and backward runs; the other methods use only forward runs. Along with two other methods, SE-NLI has the lowest computational requirements, requiring only a single run to generate an explanation. If we assume that computing both forward runs and backward runs is more expensive than only computing forward runs, SE-NLI has the lowest computational requirement of all the methods during the prediction time.

Compared to the other methods, SE-NLI requires additional computation during training. However, the additional computational cost are of reasonable amount. In our implementation, we trained the explanation generator for only 0.5 epochs, whereas the whole training procedure for NLI lasted over 3 epochs.

SE-NLI requires additional computation to get outputs from a number of perturbations. However, these additional computations are still affordable, because forward runs for perturbations are much faster than back-propagation and parameter updates.

5.2.4 Effect of loss functions. It is notable that the model trained with cross-entropy loss (SE-NLI (CE)) dramatically fails on *conflict* tags. In the early stage of our experiment, we observed that using the cross-entropy converges slower than the correlation loss. The difference in the final accuracy (precision) between cross-entropy loss and correlation loss was not as significant when we used a much larger learning rate without decaying. However, that configuration had an observable negative effect on the original classification task.

The motivations of using the correlation loss was that cross-entropy loss would penalize the predictions (location in the sequence) that are not in the weak label (most informative instances) more harshly than correlation loss does. However, cross-entropy loss exhibits better accuracy, suggesting that correlation loss is better for ranking metrics.

5.3 Analysis

5.3.1 Fidelity. We evaluated the fidelity of our explanation with a deletion experiment, which is commonly used in attribution analysis papers [4]. We selected 2,000 sentence pairs whose gold labels were contradiction. We deleted tokens in decreasing order of conflict tag scores and measured how much the average accuracy changed. Figure 3 shows that SE-NLI is good at predicting the tokens that will make the system's accuracy plummet much faster when deleted.

5.3.2 Multi-task learning. It is unclear whether NLI knowledge is actually needed for the token-tagging task. To investigate this question, we trained the explanation generator in a separate network. For training data, we recorded weak-supervision input from the training of SE-NLI and applied it to the target network. We tested two cases: in one, the model was initialized with pre-trained BERT

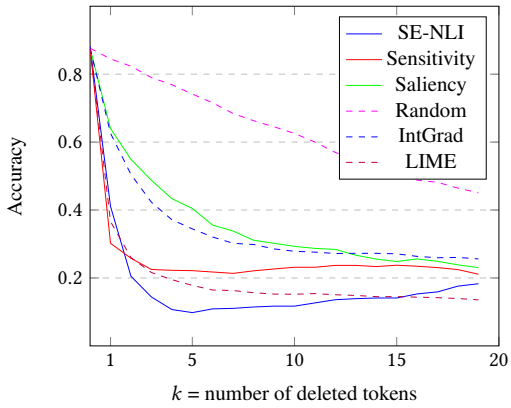

Fig. 3. Sentence classification accuracy changes as tokens are deleted in the order of decreasing scores of explanation prediction. Rapid accuracy drops are considered evidence of a good explanation [4].

Table 7. Effect of multi-task learning. MTL with NLI is the same as the model SE-NLI (CO) in Table 5. BERT start and cold start were trained using the same supervision as the training of SE-NLI but they were trained on vanilla BERT or a random initialization rather than on an NLI-trained model.

	Conflict			Match			Mismatch		
Model	P@1	MAP	Acc	P@1	MAP	Acc	P@1	MAP	Acc
MTL with NLI	0.750	0.723	0.800	0.965	0.903	0.760	0.817	0.830	0.714
BERT start	0.625	0.640	0.798	0.965	0.890	0.754	0.783	0.791	0.688
Cold start	0.484	0.544	0.775	0.700	0.711	0.584	0.537	0.628	0.523

Model	Sentences
MTL with NLI	**P:** I don' t know um do you do a lot of camping **H:** I know exactly.
BERT start	**P:** I don' t know um do you do a lot of camping **H:** I know exactly.

Table 8. Comparison of conflict prediction of MTL model (MTL with NLI) and baseline model (BERT start). **P** stands for premise and **H** stands for hypothesis.

(BERT start), whereas in the other, the parameters were randomly initialized (Cold start). Table 7 shows the results of alternative models. The BERT start model shows comparable performance for the match tag, but it does not reach the performance of the original model for the other two tags. Thus, we conclude that there is meaningful gain in using multi-task learning for explanation generator.

Table 8 shows a case that highlights two models with different levels of language understanding. Although the word "camping" does not carry conflicting meaning, the model without NLI knowledge (BERT start) assigns it a high score. Moreover, this model assigns a lower score to the token "know" in the hypothesis than it does to the "know" token in the premise.

5.3.3 Hyper-parameters. In this subsection, we demonstrate the change in the model's performance as the hyper-parameter values changes.

When the tokens in the inputs are deleted for perturbations, the number of deleted tokens for each perturbation is sampled from a geometric distribution. We found that deleting a flexible number of tokens is superior to deleting only one token. Figure 4 shows that the MAP changes as this parameter changes. The score decreases if too many tokens (0.9) or too few tokens are deleted. The value of p being 0 implies that always single token is deleted, and, in this setting, MAP score for the *match* tag drops. We expect that the score drop is much larger on the match tag because it requires that a greater number of tokens be tagged. Specifically, for the match tag, 63% of the tokens in a sentence pair are tagged, whereas for conflict, only 30% are tagged. We expect that only deleting a single token causes the model to generate only a few "most important" tokens, which are insufficient to select 60% of tokens for the *match* tag.

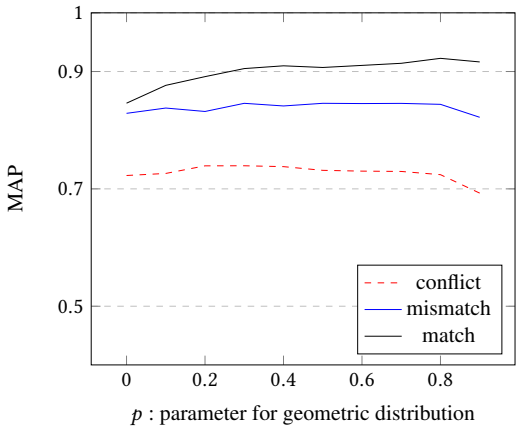

Fig. 4. Changes in MAP as the parameter p of geometric distribution changes, which decides the length of deleted sequence. Larger p values would result in a longer sequence being deleted.

Another hyper-parameter is the number of perturbations generated when selecting the most informative instance. For the numbers reported above, our method was trained by generating 20 perturbed inputs for each instance. If only a small number of perturbations are considered, even the most informative instance could result in a small difference in outputs. We used the strategy of rejecting the instance and skipping the training when the most informative score is below the threshold (Equation 5). This strategy helps the training succeed even when we use small numbers of perturbations, as fewer perturbations lead to more instances having low informative scores. Figure 5 shows the MAP scores for the *conflict* tag as the number of perturbations changes. As expected, if the number of perturbations is fewer than five, the accuracy is reduced, and the difference increases when no threshold is applied.

5.3.4 Qualitative analysis. We examined the proposed method's actual output to understand the model's behavior. We considered 18 instances. The NLI task has three labels, so the confusion matrix for the prediction has $3 \times 3 = 9$ entries. Two examples are presented for each of nine entries. We list the first appearing instances in the dataset that matches the entries of the confusion matrix. All the instances were from the validation (matched) split. Tables 9, 10 and 11 each present six examples. Table 9 contains examples whose gold label is "contradiction", Table 10 contains examples for "entailment" and Table 11 contains examples for "neutral".

Although each of three tags could show complementary information, it is difficult to list all three scores for all token in a simple format. Thus, for each example, we presented the single tag that

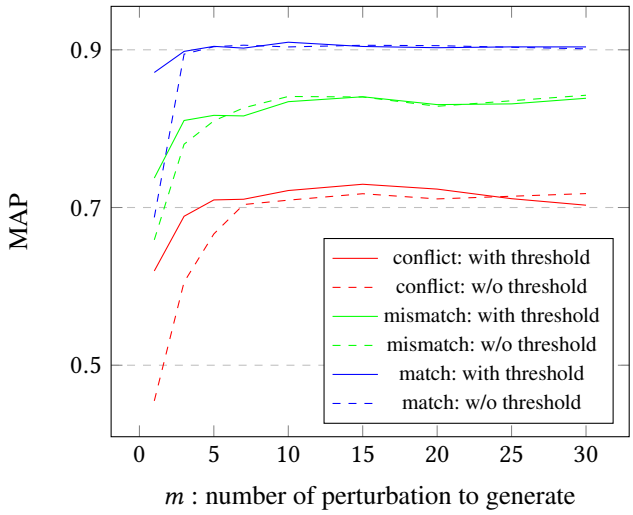

Fig. 5. Changes in MAP for the *conflict* tag in number of perturbations per step changes.

is most relevant to the model's prediction. *Match* (blue) tag scores are displayed for entailment, *conflict* (red) for contradiction and *mismatch* (green) for neutral. Scores are linearly normalized for presentation by color. As there are negative scores, the tokens with white backgrounds could have negative scores.

In the first example of Table 9, the model predicted the label to be entailment, and the gold label is entailment. The token "all" in the hypothesis is not considered to be "match". We can at least expect that the model did not consider "all" to match any of the tokens in the premise, but simply treated it as unimportant token. In the third example of Table 9, the model assigns a high score to the token "long". We can expect that the model failed to infer that this expression is contradictory to "has never really let".

Similarly, in the third example of Table 10, the model assigns a high mismatch score to the token "cold," implying that it concluded that this token contains information that cannot be inferred from the premise.

6 CONCLUSION

We investigated an approach to generate an explanation for a neural natural language inference method by defining token tags that show the role of the token in the language inference. We described a new weak-supervision training method to build a neural model that contains both an explanation-generating function and a sentence classification function in a shared network. We showed that our proposed model outperforms strong baselines, while our model has the least computational cost of those considered.

We expect a few directions for future work. The current work defines only three possible roles for each tokens. These definitions should be supplemented to provide more detailed information. Also, we would expect to see more fine-grained grouping of tokens, so that tokens for same entity or same actions could be aligned across the sentences. We hope that this work will help researchers to investigate and use neural models for information retrieval tasks.

Prediction (Label)	Sentences
entailment (contradiction)	**P:** The most important directions are simply up and up leads eventually to the cathedral and fortress commanding the hilltop, and down inevitably leads to one of three gates through the wall to the new town. **H:** Go downwards to one of the gates, all of which will lead you into the cathedral.
entailment (contradiction)	**P:** But uh these guys were actually on the road uh two thousand miles from from home when they had to file their uh their final exams and send them in **H:** These men filed their midterm exams from home.
neutral (contradiction)	**P:** What' s truly striking, though, is that Jobs has never really let this idea go. **H:** Jobs never held onto an idea for long.
neutral (contradiction)	**P:** Even if you' re the kind of traveler who likes to improvise and be adventurous, don' t turn your nose up at the tourist offices. **H:** There' s nothing worth seeing in the tourist offices.
contradiction	**P:** This site includes a list of all award winners and a searchable database of Government Executive articles. **H:** The Government Executive articles housed on the website are not able to be searched.
contradiction	**P:** Yeah i i think my favorite restaurant is always been the one closest you know the closest as long as it' s it meets the minimum criteria you know of good food **H:** My favorite restaurants are always at least a hundred miles away from my house.

Table 9. Our model's explanation score output for examples whose gold labels are **contradiction**. Different tags are shown depending on the model's actual prediction. If the model's prediction is entailment, the scores for the *match* tag are highlighted blue. For neutral predictions, the *mismatch* scores are highlighted green. For contradiction predictions, the *conflict* scores are highlighted red. **P** stands for premise, and **H** stands for hypothesis.

ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-1813662. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Abdulaziz Alamri and Mark Stevenson. 2016. A corpus of potentially contradictory research claims from cardiovascular research abstracts. *Journal of biomedical semantics*, 7(1):36.

[2] Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. 2018. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations (ICLR 2018)*.

[3] Robert Andrews, Joachim Diederich, and Alan B Tickle. 1995. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6):373–389.

Prediction (Label)	Sentences
entailment	**P:** Uh i don' t know i i have mixed emotions about him uh sometimes i like him but at the same times i love to see somebody beat him **H:** I like him for the most part, but would still enjoy seeing someone beat him.
entailment	**P:** You and your friends are not welcome here, said severn. **H:** Severn said the people were not welcome there.
neutral (entailment)	**P:** I' m not sure what the overnight low was **H:** I don' t know how cold it got last night.
neutral (entailment)	**P:** Mortifyingly enough, it is all the difficulty, the laziness, the pathetic formlessness in youth, the round peg in the square hole, the whatever do you want? **H:** Many youth are lazy.
contradiction (entailment)	**P:** And uh as a matter of fact he' s a draft dodger **H:** They dodged the draft, i' ll have you know.
contradiction (entailment)	**P:** I' m kind of familiar with the weather out that way in west Texas but not in not in lewisville **H:** I do not know the weather conditions in lewisville.

Table 10. Our model's explanation score output for examples whose gold labels are **entailment**. Different tags are shown depending on the model's actual prediction. If the model's prediction is entailment, the scores for the *match* tag are highlighted blue. For neutral predictions, the *mismatch* scores are highlighted green. For contradiction predictions, the *conflict* scores are highlighted red. **P** stands for premise, and **H** stands for hypothesis.

[4] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. " what is relevant in a text document?": An interpretable machine learning approach. *PloS one*, 12(8):e0181142.

[5] Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. *EMNLP 2017*, page 159.

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

[7] Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

[8] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NIPS 2018, Montreal, Canada, December 3-8, 2018*.

[9] Samuel Carton, Qiaozhu Mei, and Paul Resnick. 2018. Extractive adversarial networks: High-recall explanations for identifying personal attacks in social media posts. In *EMNLP*.

[10] Daniel Cohen, Brendan O'Connor, and W Bruce Croft. 2018. Understanding the representational power of neural retrieval models using nlp tasks. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 67–74. ACM.

[11] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

[12] Mark Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30.

Prediction (Label)	Sentences
entailment (neutral)	**P:** Tuppence rose. **H:** Tuppence floated into the air.
entailment (neutral)	**P:** What changed? **H:** What was unique?
neutral	**P:** The new rights are nice enough **H:** Everyone really likes the newest benefits
neutral	**P:** Calcutta seems to be the only other production center having any pretensions to artistic creativity at all, but ironically you' re actually more likely to see the works of satyajit Ray or mrinal Sen shown in Europe or North America than in India itself. **H:** Most of mrinal Sen' s work can be found in European collections.
contradiction (neutral)	**P:** Um- hum um- hum yeah well uh i can see you know it' s it' s it ' s it' s kind of funny because we it seems like we loan money you know we money with strings attached and if the Government changes and the country that we loan the money to um i can see why the might have a different attitude towards paying it back it' s a lot us that you know we don' t really loan money to to countries we loan money to governments and it' s the **H:** We don' t loan a lot of money.
contradiction (neutral)	**P:** I' m not opposed to it but when its when the time is right it will probably just kind of happen you know **H:** I cannot wait for it to happen.

Table 11. Our model's explanation score output for examples whose gold labels are **neutral**. Different tags are shown depending on the model's actual prediction. If the model's prediction is entailment, the scores for the *match* tag are highlighted blue. For neutral predictions, the *mismatch* scores are highlighted green. For contradiction predictions, the *conflict* scores are highlighted red. **P** stands for premise, and **H** stands for hypothesis.

[13] 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.

[14] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. 2018. Towards explanation of dnn-based prediction with guided feature inversion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1358–1367. ACM.

[15] Kathrin Eichler, Feiyu Xu, Hans Uszkoreit, and Sebastian Krause. 2017. Generating pattern-based entailment graphs for relation extraction. In **SEM*.

[16] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

[17] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

[18] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data*

Science and Advanced Analytics (DSAA), pages 80–89. IEEE.

[19] Aditya Gupta and Greg Durrett. 2019. Effective use of transformer networks for entity tracking. 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 759–769.

[20] Anand Gupta, Manpreet Kaur, Shachar Mirkin, Adarsh Singh, and Aseem Goyal. 2014. Text summarization through entailment-based minimum vertex cover. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (* SEM 2014)*, pages 75–80.

[21] Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912. Association for Computational Linguistics.

[22] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer.

[23] Andrew Hickl and Jeremy Bensley. 2007. A discourse commitment-based framework for recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 171–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

[24] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2018. Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8779–8788.

[25] 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.

[26] Hyun Duk Kim and ChengXiang Zhai. 2009. Generating comparative summaries of contradictory opinions in text. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 385–394. ACM.

[27] Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117.

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

[29] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

[30] Bernardo Magnini, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Günter Neumann, Tae-Gil Noh, Sebastian Pado, Asher Stern, and Omer Levy. 2014. The excitement open platform for textual inferences. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 43–48.

[31] Christopher Manning. 2016. Understanding human language: Can nlp and deep learning help? In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1–1. ACM.

[32] David Alvarez Melis and Tommi Jaakkola. 2018. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pages 7786–7795.

[33] George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

[34] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136.

[35] Chen Qu, Feng Ji, Minghui Qiu, Liu Yang, Zhiyu Min, Haiqing Chen, Jun Huang, and W Bruce Croft. 2019. Learning to selectively transfer: Reinforced transfer learning for deep text matching. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 699–707. ACM.

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

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

[38] 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*, pages 1135–1144. ACM.

[39] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2662–2670.

[40] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. page 3145–3153.

[41] K Simonyan, A Vedaldi, and A Zisserman. 2014. Deep inside convolutional networks: visualising image classification models and saliency maps.

[42] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org.

[43] Noha S Tawfik and Marco R Spruit. 2018. Automated contradiction detection in biomedical literature. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 138–148. Springer.

[44] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2019. Generating token-level explanations for natural language inference. 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 963–969.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[46] Ngoc Phuoc An Vo, Simone Magnolini, and Octavian Popescu. 2015. Paraphrase identification and semantic similarity in twitter with simple features. In *SocialNLP@NAACL*.

[47] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

[48] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM.

[49] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv*, pages arXiv–2004.

[50] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

[51] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

[52] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. 2017. Visualizing deep neural network decisions: Prediction difference analysis. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.