# NLIZE: A Perturbation-Driven Visual Interrogation Tool for Analyzing and Interpreting Natural Language Inference Models

Shusen Liu, Zhimin Li, Tao Li, Vivek Srikumar, Valerio Pascucci and Peer-Timo Bremer

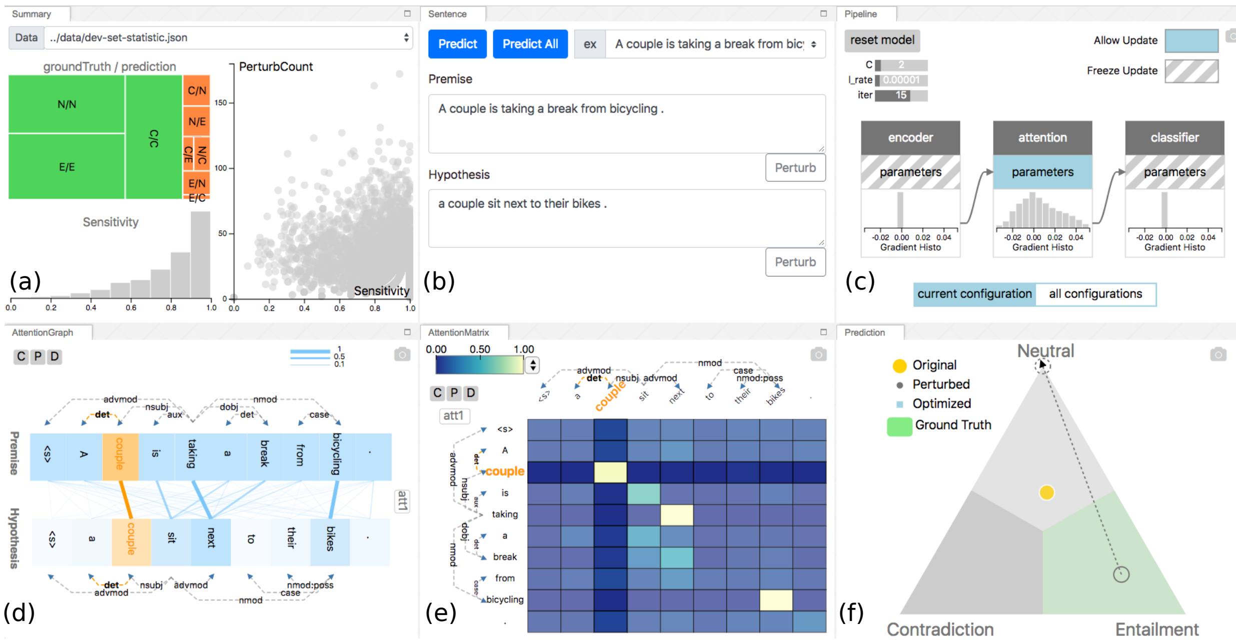


Fig. 1. The interface of the proposed system. During exploration, we can filter through a large number of sentence pairs summarized in (a). the current selected pair is displayed in (b). The model internal information (attention) is displayed in (d) and (e). The predicted probability (one of three labels: *neural*, *entailment*, *contradiction*) is shown in the barycentric coordinate in (f). Finally, the high-level model structure and the updates to the model are summarized in (c).

**Abstract**— With the recent advances in deep learning, neural network models have obtained state-of-the-art performances for many linguistic tasks in natural language processing. However, this rapid progress also brings enormous challenges. The opaque nature of a neural network model leads to hard-to-debug-systems and difficult-to-interpret mechanisms. Here, we introduce a visualization system that, through a tight yet flexible integration between visualization elements and the underlying model, allows a user to interrogate the model by perturbing the input, internal state, and prediction while observing changes in other parts of the pipeline. We use the natural language inference problem as an example to illustrate how a perturbation-driven paradigm can help domain experts assess the potential limitation of a model, probe its inner states, and interpret and form hypotheses about fundamental model mechanisms such as attention.

**Index Terms**—Natural Language Processing, Interpretable Machine Learning, Natural Language Inference, Attention Visualization

## 1 Introduction

As demonstrated by many recent successes, neural-network-based machine learning approaches have garnered increasing popularity and have been adopted in a wide variety of applications. However, researchers as well as practitioners often need to overcome many obstacles during the training, debugging, and tuning processes to realize the full potential of these models. Interpreting the internal mechanism and analyzing how predictions are made are critical for both the design and deployment of a model. More importantly, the ability to pinpoint where and how an error is made and propose hypotheses for the cause of the failure is key to identifying the limitations of and improving upon existing models. However, providing meaningful answers to these questions is challenging and has been described as impossible by many.

Recently, significant research has been developed to combat the model interpretability challenges. Both the machine learning as well as the visualization community have proposed a number of promising techniques aimed at interpreting convolution neural networks (CNN) [3, 13, 21, 22, 33, 39, 40]. At the same time, these approaches also remind us how little we truly understand about the inner mechanisms of such deep neural networks.

Compared to image classification tasks, natural language processing (NLP) systems often involve additional challenges such as the discrete nature of words. For example, *feature visualization* [21], an often deployed technique for illustrating what image features a given part of the network (e.g., neuron, layer, or channel) captures, cannot be readily generalized to natural language. An image (pixel values) corresponds to


- *Shusen Liu, Peer-Timo Bremer is with Lawrence Livermore National Laboratory. E-mail: {liu42, bremer5}@llnl.gov.*
- *Zhimin Li, Valerio Pascucci is with SCI Institute, University of Utah. E-mail: {zhimin, pascucci}@sci.utah.edu. Tao Li, Vivek Srikumar is with School of Computing, University of Utah, E-mail:{tli,svivek}@cs.utah.edu*


*Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx*

a continuous solution space. which is more accessible for optimization and human visual recognition (i.e., the existence or the absence of patterns). For natural language, even though we can encode a word as a vector [17, 26], the word embedding space is still discretely defined, in which the interpolation of two vectors (i.e., two words) does not hold clear meaning.These restrictions call for new avenues for solving the interpretation challenges of natural language models, which motivates the proposed work.

Despite a number of recent advances, most of existing techniques study the model as an invariant object, where the model's behaviors are recorded and analyzed in an offline fashion. However, the exploratory nature of the model interpretation often leads to many *"what if ..."* types of questions, such as, what if we perturbed the current input? Will the prediction be stable? What if we change one of the critical internal states of the model? How would the modification affect the prediction? What if the current prediction is wrong? How and where could we apply minimal change to the model to produce the correct result? And how would it affect the internal state we care about? These types of queries form a natural way to gain an understanding and develop hypotheses of the model mechanisms by *interrogating* how the components of a model interact with each other in a dynamic setting. Hypothetically, we can code specialized experiments for each of these scenarios. However, such a process is not only tedious but also ignores the iterative nature of the exploratory analysis. Often, these questions are not pre-determined. Instead, new exploration paths arise as we investigate and analyze previous observations.

Here, we aim to provide immediate and informative answers to these *"what if"* questions by combining the expressive power of visualization and the direct online query/optimization of the neural network model. Instead of viewing the models as invariant objects, we approach the interpretability challenge by studying them in a dynamic environment. By employing a perturbation-driven scheme, we probe the internal states of the model and examine how changes in one part of the pipeline (the input, internal states, and output prediction, see Fig. 4) affect others, which in turn provides a new perspective to address the model interpretation problem.

We materialize the goal of the perturbation-driven exploration in an interactive visualization system for natural language inference models [24]. However, the components of the visualization and the overall concept can be readily extended to other NLP tasks, such as question and answer, text summarization, etc. In its simplest form, the inference task asks whether the relationship between **sentence A** and **sentence B**: is (1) *entailment* (one can infer **B** from **A**), (2) *contradiction* (**B** disagrees with **A**) or (3) *neutral* (**A** and **B** talk about different/unrelated things). Natural language inference addresses the fundamental challenge of identifying semantic relationships between sentences and is a core NLP task (see Section 2.1 for details).

One recent advance in neural natural language process models is the introduction of *attention mechanisms* [2, 36] (Section 2.3). Intuitively, attention asks which parts of the input are deemed more important for making a prediction. Attention is often represented via weights for individual words or pairs of words (i.e., the alignment between words in different sentences). There have been many theories about how attention works in various models. The proposed tool introduces a perturbation-driven visual analytics environment, where the domain experts can study how changes in sentence input, attention, or prediction affect each other, which helps the experts develop deeper intuitive and alternative hypotheses. In addition, we propose to enhance the standard visual encoding (e.g., as a bipartite graph or as a matrix) of the attention matrix by overlaying sentence linguistic structure to allow grammar-guided simplification of the visual representation. Finally, as discussed in Section 6.4, the ability to examine how attention corresponds to the grammatical structure also enables domain experts to speculate about the potential benefits of including the linguistic structure in the design of the attention component of the model.

In summary, the key contributions of this paper are:

- A perturbation-driven exploration scheme derived from close examination of how domain experts conduct exploratory analysis on end-to-end NLP neural network models;
- The NLIZE system that enables the perturbation-driven exploration by providing an intuitive environment that allows domain experts to readily express hypotheses and obtain instantaneous feedback;
- An optimization method for correcting a failed prediction based on a natural extension of the margin-infused relaxed algorithm (MIRA) to neural networks; and
- A visual encoding of the attention by imposing sentence linguistic structure to allow grammar-guided sentence simplification.

## 2 BACKGROUND

The target audience of the proposed tool is domain experts who analyze and develop NLP models. Therefore, certain background knowledge in NLP is required to fully understand and appreciate the technique discussed in this paper. In this section, we first explain the definition of a natural language inference (NLI) task and how it fits into the grand challenges in NLP. Then, we examine the common architectural characteristics shared by many state-of-the-art neural network models. Finally, we discuss the role attention plays in the model and why attention is closely tied to model interpretability.

### 2.1 Natural Language Inference

Natural Language Inference (NLI) [7] is an important machine understanding task in NLP. The goal of NLI is to predict the relationship between a premise (**P**) sentence and a hypothesis (**H**) sentence. The prediction falls in one of three categories: *entailment* (**E**), *contradiction* (**C**), and *neutral* (**N**). A simple example is shown in Table 2.1. In this case, the premise is "A boy ate an apple". The hypothesis statement "A kid ate fruit" can be concluded from the premise. Therefore, the relationship between the premise and hypothesis is *entailment*. However, we should note that such a relationship is not necessarily reversible. Since the concept "fruit" is less restrictive than that of "apple", we cannot conclude "A boy ate an apple" from the statement "A kid ate fruit". The same logic applies to the hypothesis of "A boy ate a Fuji apple.", in which the premise neither implies nor opposes the hypothesis, and therefore, their relationship is *neutral*.

Table 1. An illustration of natural language inference.

| P / H | sentences | entail | contradict | neutral |
|---|---|---|---|---|
| premise | A boy ate an apple. | - | - | - |
| hypothesis | A kid ate fruit. | ✓ | | |
| hypothesis | A boy ate a banana. | | ✓ | |
| hypothesis | Tom ate an apple. | | | ✓ |
| hypothesis | A boy ate a Fuji apple. | | | ✓ |

At first glance, the task of natural language inference may seem less practical compared to other well-known NLP challenges such as machine translation; however, the ability to distinguish the entailment and contradiction relationship is fundamental to understanding natural language at large. Considering the ambiguousness of natural language and the polysemy of words, the inference task can become quite challenging (especially from the learning algorithm's point of view). Take the following sentences as an example (here **P** refers to premise, **H** refers to hypothesis): (**P**) Facebook's IPO electrified the general public; (**H1**) Facebook went public; (**H2**) General Electric went public; (**H3**) People ignored Facebook's IPO. The literal similarity between "Electric" and "electrified" may trick a model to predict **H2** as *entailment*. A model likely will also fail to understand the link between "went public" and "IPO", and therefore, mistake **H1** as *neutral*. Recently, Bowman et al. introduced a large corpus [4] for NLI tasks, which has helped spawn a new wave of effective neural network models for those tasks. Here, we focus on the analysis of the decomposable attention model [23] on this dataset (see **Appendix A** for a detailed description of the model).

### 2.2 Neural Network Models in NLP

Neural network models employed for computer vision and NLP tasks share a key common characteristic: a majority of recent approaches use end-to-end models. That is, the entire model operates as a black-box that takes vectorized inputs and yields a final prediction for a task of interest. However, as discussed in the introduction, the discrete nature of words and sentences presents additional challenges for interpreting NLP models.

Many recent end-to-end neural network models in NLP, despite having diverse network architectures, share a similar high-level design that consists of three distinct stages (encode, align, classify, see Fig. 2). The existence of shared conceptual structures means the proposed tool, although designed for NLI tasks, can be readily generalized for a much broader set of applications.

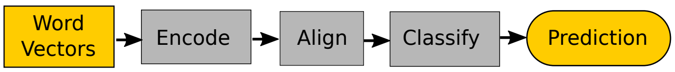


Fig. 2. The shared structure of end-to-end NLP neural network models.

These end-to-end models usually take pre-trained word vectors (numerical vector representation of individual words, in which the semantic similarities are expressed by distances in the high-dimensional vector space [17, 26]) as input, and then in the encoding stage, the pre-trained vectors are adjusted to the specialized task at hand. Subsequently, the next stage aims at finding alignment between words in the input sequences. For the NLI task, this means finding the correspondence between words from the premise sentence to the hypothesis sentence (see details in Fig. 3 and Section 2.3). Finally, in the last stage, the alignment information and the encoded vector representations are aggregated and then used as features for a classification network (last part of the end-to-end neural network). Since all three stages of the model are trained jointly in an end-to-end fashion, it is important to explore the interaction between intermediate representations and predictions to make sense of how predictions are made inside such a model.

### 2.3 Attention Mechanism

Among the three stages, the second stage often constitutes a crucial part of the model as it determines where the classifier will focus for generating a prediction. The operation to compute the alignment between words in the input is referred to as the attention mechanism [2]. The introduction of the attention mechanism allows pairwise interaction between internal representations of words. This interaction can be naturally explained as a form of alignment that exposes an interpretable layer in end-to-end neural networks. Recently, the attention mechanism has contributed to the strong performance of many NLP models [23, 29, 31, 32, 38].

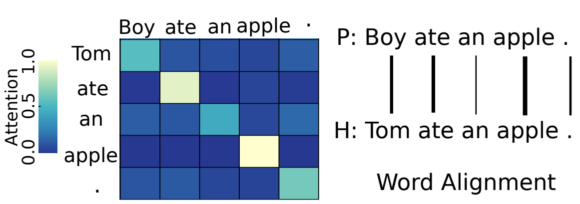


Fig. 3. Attention can be naturally explained as a form of alignment that exposes an interpretable layer in end-to-end neural networks. On the left, the matrix shows the attention between the premise and hypothesis pair. On the right, we illustrate that the attention can also be interpreted as alignment.

For natural language inference task, as illustrated in Fig. 3, the attention information can be represented as a matrix describing the soft alignment between words in the premise and words in the hypothesis. As we can see in this example, the higher values indicate significant alignments between words. The *subject*, *verb*, and *object* between these two sentences are aligned correctly (words, such as "an", which has less importance in determining sentence relationship, was assigned a lower score). Interestingly, the difference between the subject words does not affect their alignment. The classification stage can then utilize this information to determine if the subject of the sentence is different, and therefore, despite the other part of the sentences being identical, the relationship between the premise and the hypothesis is neutral.

## 3 RELATED WORK

With the increasing demands for model accountability (e.g., what is the evidence for making the prediction?) and model fairness (e.g., is the prediction affected by the bias in the training data), the interpretability of the machine learning model has significant implications. As a result, a wealth of research has been developed that focuses on the explanation and interpretation of neural network models by both the machine learning and visualization communities.

**Model Interpretability.** In the machine learning community, one of the most studied areas in model interpretability is the interpretation of the convolution neural network (CNN) [21, 22, 33, 39, 40]. These methods either identify the pattern that maximally activates a part of the network (referred to as *feature visualization*, i.e., what feature does the given neuron, channel, or layer capture?), or reveals what areas in the input contribute most to the predicted label (often referred to as *attribution*, i.e., attribute the prediction label in the input domain). Besides the techniques designed for specific types of models, a few study [10, 15, 28] approach the interpretability challenge from a model agnostic perspective, which makes them applicable to different applications. In the LIME [28] work, a simpler linear model is fitted near the prediction of interests to provide a simpler but easier to understand snapshot of the classifier near a specific input. Despite being invaluable for generating a human understandable explanation, however, the lack of ability to access and explore the internal states of the system limits their application. To avoid similar limitations, the proposed system enables direct access and manipulation of the internals of the model.

**Visual Exploration of Neural Network Models.** Due to the effectiveness of exploring complex relationships, visualization has long been adopted for interpreting neural network models [35]. Recently, the increasing demand for interpretability for machine learning models has motivated the introduction of many visualization studies that are dedicated to neural network models. In [3], Bilal et al. analyzed classification errors in convolutional neural networks (CNN) and answered the question of whether class hierarchy is learned in the network. DeepEyes [27] focused on analyzing the training process of CNN, which provided domain experts design guidelines for deep neutral network architecture. ActiVis [8] introduced an interactive system tailored for tuning industry-scale neural networks in Facebook. The TensorFlow Graph Visualizer [37] work designed a hierarchical visual representation to encode complex computation graphs of deep neural networks for the TensorFlow framework [1]. Finally, Liu et al. [12] presented a visualization tool for understanding the often challenging training process of deep generative models.

**Visualization of NLP Models.** Compared to many studies for making sense of convolutional neural networks, relatively fewer studies have been dedicated to neural language processing models. The previous works [9, 11, 14, 34] not only demonstrated the benefit of visualization in understanding NLP models but also revealed enormous possibility for future applications. The early work on character-level recurrent networks [9] demonstrates the effectiveness of the hidden state in capturing the unique pattern in the training text. The compositionality (i.e., build the meaning of a sentence from the meaning of words and phrases) exhibited in neural network linguistic models is examined by utilizing techniques inspired by model interpretability techniques in computer vision by Li et al. [11]. In the RNNVis [19] work, Ming et al. visualize the hidden state units in a recurrent neural network based on their expected responses to model input. The RNNbow [5] system helps domain experts understand how the model is trained by visualizing the backpropagation gradient information. The LSTMvis [34] work employs a line-plot style visual encoding to represent the time-varying hidden states of the recurrent neural network. The word embedding visualization work [14] illustrates how semantic relationships can be recovered by linear projections in the high-dimensional word embedding space (e.g., word2vec [17] or Glove [26]). All the previously discussed NLP model visualization works either focus on the training process or examine the trained model as a fixed object. In contrast, the proposed method allows direct user interaction with the internals of the trained model in a dynamic environment, which enables unrestrictive exploratory analysis that has not been possible before.

# 4 Task Analysis

The primary driving force for designing the proposed tool is the error analysis challenges faced by our long-term collaborators working on natural language inference research. During the entire design and development process, we worked closely with two NLP experts via weekly meetings over a period of roughly seven monthes. Their constant evaluation and feedback have helped shape the tool we see today. During this period, we have conducted extensive discussions to understand the common approaches employed by researchers for assessing the behavior of a model. Predictive accuracy has its place as an objective evaluation metric to measure the overall effectiveness of the model; however, the accuracy number alone does not provide the full story. For example, the model may produce correct predictions for the "wrong" reasons (e.g., pick up an unintended pattern in the training data that cannot be generalized in real-word scenarios). As a result, the domain experts rely on an exploration-centric approach to conduct error analysis and obtain intuition. The experts often start with simple examples and then make minor perturbations (replace a word or phrase) to the input and observe the change in the prediction (and potential failures). This exercise helps the domain experts reason about the relationship between input elements and the predicted results. For many NLP models, the attention information (see Section 2.3) is essential to infer the mechanism of the model. Experts often print out the attention values or generate plots to visualize sentence alignment or compare attention values. As the experts explore more variation of similar examples, combined with their domain knowledge, they may develop hypotheses about the cause of failures or ask additional questions that lead to further experimentation.

In such an exploration workflow, NLP researchers often need to utilize multiple scripts and manually run all the experiments. The batch process approach is not only time consuming, but it also hinders the flow of reasoning that relies on obtaining instantaneous feedback and making many on-the-fly adjustments. Therefore, the expert-driven exploration process can greatly benefit from the introduction of an interactive visual exploration environment, where the researchers can easily express their hypotheses and obtain quick visual feedback of the results. Moreover, by introducing new visual encodings and summarizations, we can drastically expand the ways domain experts interact with the model, enabling exploration options that previously were not possible either due to either tedious manual operation or the lack of communication channels. To support the exploratory workflow for analyzing NLI models, we design the proposed tool to address the following tasks based on the discussions with NLP experts;

- **T1**: Understand the stability of a prediction, i.e., how do perturbations of the input affect the behavior of the model?
- **T2**: Examine the attention mechanism, i.e., what is the relationship between the input and the attention, and how does the attention affect the prediction?
- **T3**: When the predicted label is wrong, how can we update the model to correct the prediction? What are the effects of updating different parts of the model?

In Section 6, we will illustrate how to utilize the proposed tool for these tasks (the application scenarios 1, 2, 3 correspond to **T1-T3**, respectively).

# 5 NLIZE System

In this section, we discuss the design and implementation of the NLIZE (pronounced as "analyze"; see interface overview in Fig. 1) system. As discussed in the previous section, the experts often analyze and obtain intuition about the model by studying how altering one part of the model affects other stages of the pipeline. Such a process can be generalized as the perturbation-driven paradigm, which is used as a guiding principal for designing the proposed tool. As illustrated in Fig. 4, we enable the automated or user-guided *perturbation* (i.e., replace words) of the input sentence, the *perturbation* of attention (i.e., alter the alignment between sentences) inside the model, and the *perturbation* of the prediction (i.e., adjust the prediction by making updates to the model, the optimization is disucssed in Section 5.4). In the following sections, we describe in detail the five major components of the proposed system, namely, the sentence view (Section 5.1), the attention view (Section 2.3), the prediction view (Section 5.3), the pipeline view (Section 5.4), and the pair summary view (Section 5.5).

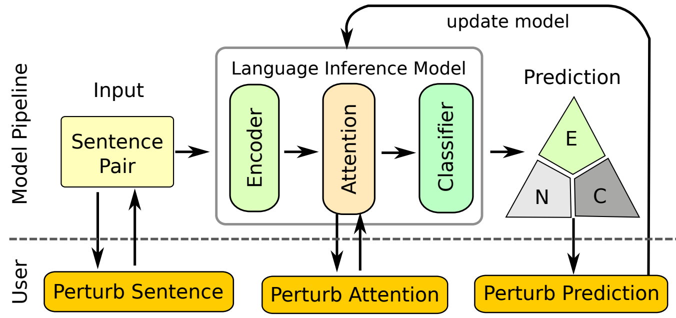


Fig. 4. Perturbation-driven exploration of the natural language inference model. In the proposed tool, we enable the interrogation of the relationship between different components of the model via the perturbation-based analysis. The user can perturb the input sentences (i.e., replace words with synonymous), perturb the attention (i.e., alter the soft alignment between sentences), and perturb the prediction (i.e., adjust the prediction by making updates to the model).

## 5.1 Sentence View

As illustrated in Fig. 1(b), the sentence view shows the premise and hypothesis sentence. To facilitate the analysis task (**T1**), we employ an automate sentence perturbation scheme that replaces *nouns* and *verbs* by their synonyms in the wordNet [18] (the standard lexical database for NLP applications). When the perturbation is applied to either of the sentences, the replaced words are highlighted in blue to signal the modification made to the original sentence.

At the top left of the view, the two controls *Predict* and *Predict All* correspond to predicting the currently displayed sentence pairs and predict all combinations of perturbed premises and hypotheses, respectively. To avoid the situation where both sentences are perturbed, we ensure only one perturbation per sentence pair (i.e., we use the original premise if the hypothesis is perturbed, or use the original hypothesis if the premise is perturbed). The previously explored original sentences are stored in the dropdown list (on the right side of the buttons) that allows the user to revisit examined examples. Also, the user can type any sentences or modify existing text in the sentence display areas to accommodate user-defined inputs.

## 5.2 Attention View

As discussed in Section 2.3, the attention is the only intermediate layer in the network that provides interpretable information for domain experts to infer the inner mechanisms of the model. Intuitively, the attention captures the alignment of words between input sentences. For the NLI model examined in this work, the attention is represented as a matrix, in which the entries in the $i^{th}$ row correspond to the probabilities of words in the hypotheses aligned to the $i^{th}$ word in the premise.

As illustrated in Fig. 5, we employ both graph and matrix visual encodings for visualizing attention. In the graph attention view (Fig. 5(a)), a bipartite graph encoding is adopted, in which the edge thickness corresponds to the attention value. The color of the rectangle text block encodes the sum of all edge values connected to it (darker shade of blue corresponds to higher values). The graph view is good for highlighting the most dominant alignments. However, if many attention values are high, the edges may become cluttered, leading to less effective visualization. Also, if the sentence structures between premise and hypothesis are drastically different, we are likely to see the prominent edges cross each other, which can also lead to confusing and misleading visual patterns. The matrix attention view (Fig. 5(b)), despite being more verbose and less efficient in highlighting the dominant alignment, does not have similar shortcomings. However, extra effort may be required for identifying the words that correspond to high-value entries in the matrix. Together, the graph and matrix views complement each other

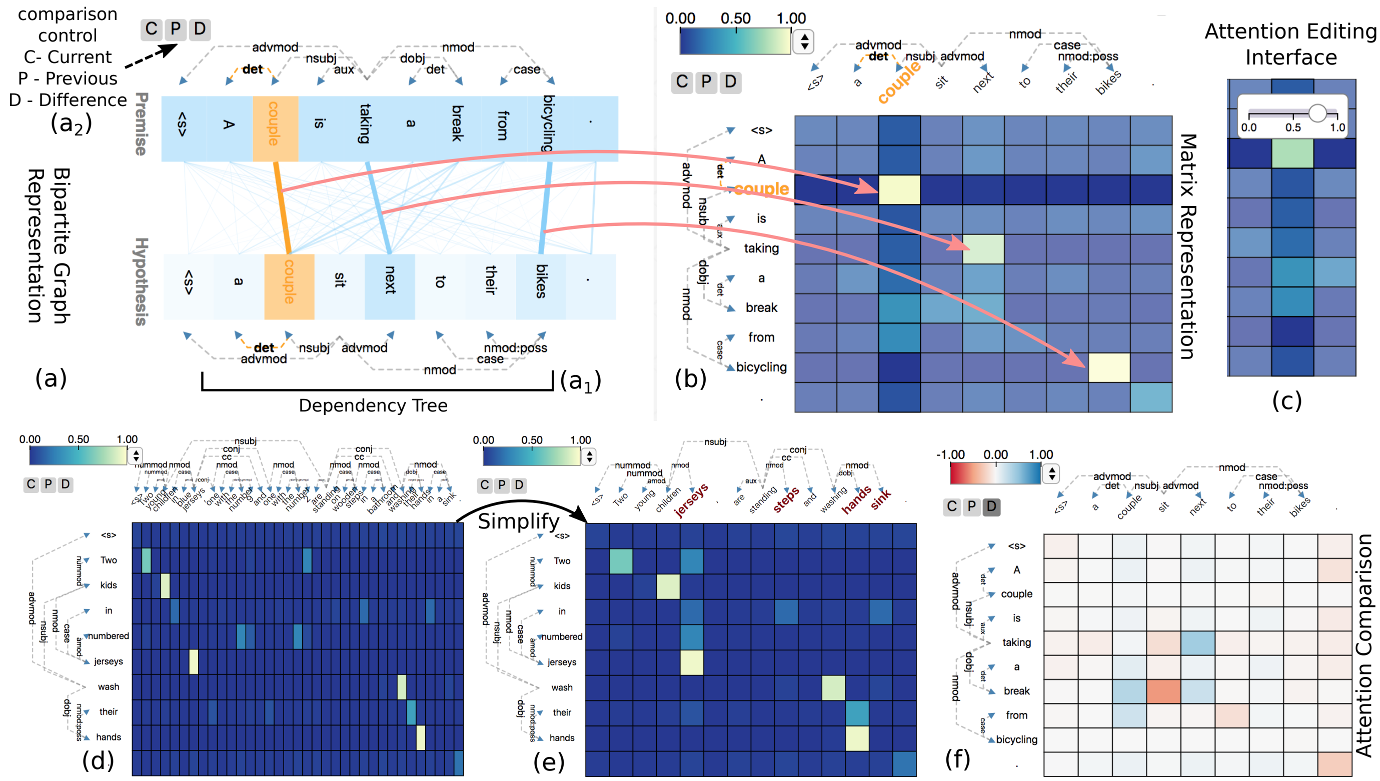


Fig. 5. Attention visualization. In the graph attention view (a), a bipartite graph encoding is adopted, in which the edge thickness corresponds to the attention value. In the matrix attention view (b), the entries of $i^{th}$ row represent the probabilities of words in hypotheses align to the $i^{th}$ word in the premise. The user can alter the attention values via the pop-up interface illustrated in (c). We overlay the dependency tree $(a_1)$ grammar structure to highlight important words and simplify complex sentence to reduce clutter (d-e). In (f), we show the difference between two attention matrix (the comparison feature is controlled by the buttons in the top left of all attention plots $(a_2)$ ).

and provide the same information from different perspectives (a related matrix/graph visualization scheme is explored in [16] for studying the brain network). To help the user recognize the correspondence during the exploration, we enable the linkage between highlighted actions in both views (see Fig. 5(a)(b), the attention of the two "couple" is highlighted).

To support the ability to perturb the attention values (**T2**), we include the attention editing functionality. The attention matrix view is the most suitable place to conduct the editing operation since it provides a direct mapping of the attention value. As we can see in Fig. 5(c), when a user clicks the cell of the matrix, a sider will pop up for customizing the attention value (as the user edits the value, each row is automatically renormalized). As illustrated in Fig. 5($a_2$)(f), we also allow the user to compare currently and previously displayed attentions by computing and visualizing their cell-wise differences (the user can toggle between different display modes using the C (current), P (previous), D (difference) buttons bellow the colormap).

Even though the attention does not explicitly encode any grammar, it often highlights essential words in the sentence structure. To help the researcher better understand the relationship between attention and sentence structure, as illustrated in Fig. 5($a_1$), we overlay the grammar dependency tree [20] structure next to the sentence. Since the dependency tree encodes the word importance information in a hierarchical manner, it is very suitable for sentence simplification tasks. Here, we utilize the grammar dependency tree to trim the decorative structure to shorten the sentence to combat the visual clutter when examining long sentences (see Fig. 5(d)). A simplification example is shown in Fig. 5(d)(e)(f).

### 5.3 Prediction View

For a given sentence pair, the model predicts a discrete probability distribution of the three labels (neutral, contradiction, and entailment). In the prediction view, as illustrated in Fig. 6(a), a prediction probability is encoded as a point in the barycentric coordinate system of the triangle. Let $\mathbf{C_1}$,$\mathbf{C_2}$,$\mathbf{C_3}$ be the three points defining a triangle, and let $p_1$, $p_2$, $p_3$ be the probabilities the label is neutral, contradiction, or entailment. The coordinate of the prediction $\mathbf{C_p}$ in the triangle is computed as $\mathbf{C_p} = p_1\mathbf{C_1} + p_2\mathbf{C_2} + p_3\mathbf{C_3}$. In the triangle, the distinctly colored background (gray, light gray, and light green) indicates the regions that correspond to different labels. The prediction result for the original sentence pair is represented by the larger yellow circle, whereas the smaller gray circles illustrate the perturbed sentence pairs. A density contour of the prediction is computed via kernel density estimation to emphasize the highly cluttered areas and distinguish the outliers. The pattern of the perturbed pairs' prediction directly conveys the stability of the model for the given sentence pair. Here, we should also take the length of the sentence into consideration as greater numbers of nouns and verbs will likely lead to more varied perturbations.

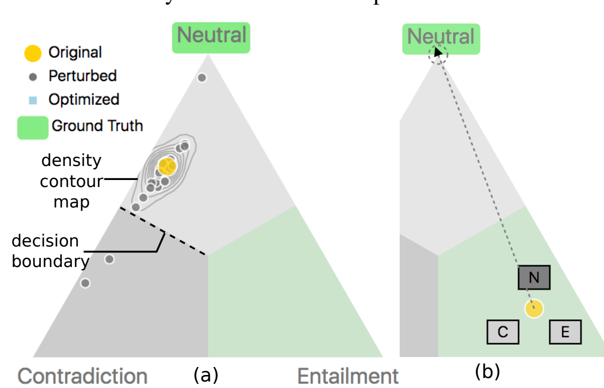


Fig. 6. In the prediction view, the prediction is encoded as a point in the barycentric coordinate system of the triangle shown in (a). A density contour of the prediction is computed via kernel density estimation to emphasize the highly cluttered areas and distinguish the outliers. As illustrated in (b), the predicted label does not match the ground truth. Therefore, we apply a label reassignment operation, which triggers a model update.

To perturb the prediction (**T3**) (the optimization for solving the prediction perturbation is discussed in Section 5.4), we need a way to communicate the reassignment of the predicted label. As illustrated in Fig. 6(b), we integrate such an operation in the prediction view. When pressing and dragging the prediction (represented as a circle), the user is presented with the three options (E, N, C) corresponding to the labels. When the user hovers on one of the options, a dotted line is shown to indicate the newly assigned label. The reassignment is applied when the user releases the mouse while hovering on the label of choice.

### 5.4 Pipeline View

The pipeline view provides a direct visual representation of the three stages (encoder, attention, classifier) of the model. In the proposed tool, we allow model parameters to be updated (via an optimization) to correct a prediction error (**T3**). The pipeline view, by visualizing the distribution of the parameter changes, informs the user about how each stage responds to the optimization.

There are many ways to update the model to correct a prediction. The simplest approach is applying standard backpropagation and overfits to the example. However, without any constraint, the update step may alter the model in unexpected ways. Instead, we adopt the idea in the *margin-infused relaxed algorithm (MIRA)* [6], where we regulate the optimization with the L2-norm of the parameter change. In the proposed tool, we obtain the target parameters by the following optimization:

$$\underset{\mathbf{W}'}{\operatorname{argmin}}(C\mathbb{J}(\mathbf{W}') + ||\mathbf{W}' - \mathbf{W_o}||^2) \tag{1}$$

where $\mathbb{J}(\mathbf{W})$ is the loss function of the neural network model, $\mathbf{W_o}$ is the original model's parameters taken as constant, $\mathbf{W}'$ is the updated parameters, and $C$ is the weighting term, which determines whether we intended to emphasize more on obtaining better fitting or deviating less from the original model. Due to the nonconvex nature of the neural networks, we use SGD to optimize the above combined loss function. With this formulation, we try to find a good approximation to the newly assigned label, while still maintaining relatively small changes with respect to the original model.

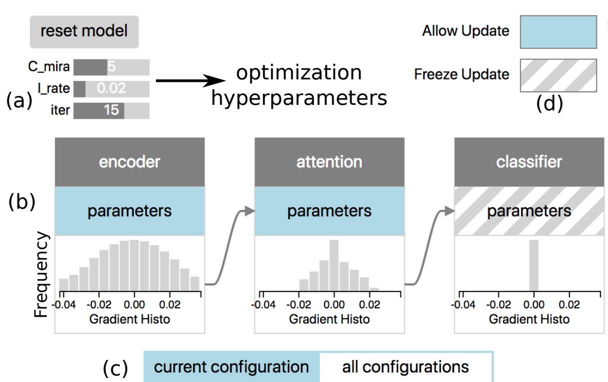


Fig. 7. The pipeline view provides the visual representation of the three stages (encoder, attention, classifier) of the NLI model. In the proposed tool, we allow model parameters to be updated to correct a prediction error via a constrained optimization. The hyperparameters for this optimization are shown in (a). In (b), we utilize a graphical representation for each stage of the pipeline. In (c), the user can select whether to use the current pipeline configuration as displayed or try all the pipeline configuration combinations for the optimization.

The optimization hyperparameters are shown in Fig. 7(a). Each stage is illustrated by a glyph (Fig. 7(b)), in which the user can enable or disable its parameter update by clicking on the blue rectangle marked with the word "parameter" (the legend about its state is shown in Fig. 7(d)). In Fig. 7(c), we select whether we want to use the current pipeline update setting as displayed or try all the pipeline configuration combinations (i.e., each stage can be either enabled or disabled; therefore, there are 8 combinations in total, or 7 if we discard the case where all stages are not enabled).

### 5.5 Summary View

We can focus on only one example at a time for a detailed analysis using the combination of all previously discussed views. Therefore, how to select a pair of sentences of interest from the development dataset, which consists of close to $10k$ examples, is an obvious challenge. In addition, the experts are also interested in obtaining a high-level understanding beyond the information prediction accuracy provides.These two goals are the two sides of the same coin. The selection task will become easier if we can generate a good visual summary of the $10k$ examples.

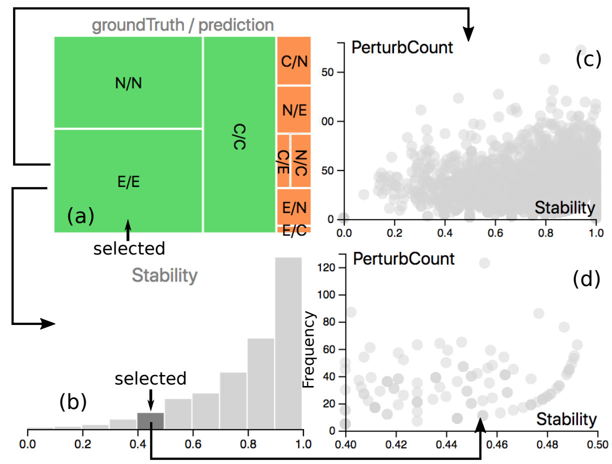


Fig. 8. We summarized all prediction results of $10k$ sentence pairs in (a). The green block indicates correct predictions, the orange block indicates wrong predictions. The user can click on a treemap node to focus on the specific type of scenarios (e.g., E/E, indicating both the ground truth and the predicted label are Entailment) to automatically reveal the histogram (b) and scatterplot (c) for displaying the selected subset. The selection can be further narrowed down by selecting the bin in the histogram. In (c) and (d), each point corresponds to one sentence pair.

To address these challenges, we introduce the summary view (see Fig. 1(a)), which consists of a treemap, a histogram, and a scatterplot, to summarize the prediction results of the $10k$ examples and provide the ability to drill down to individual examples for detailed analysis. As illustrated in Fig. 8, we utilized a treemap (a) to encode the different combinations of the ground truth label and the predicted label. The green treemap blocks correspond to examples with correct predictions, whereas the orange blocks indicate failures. The size of the block encodes the number of examples belonging to each category.

By clicking on the treemap node, we can narrow down the selection by focusing on a specific scenario. As we select the "E/E" (ground truth: E-Entailment / predicted label: E-Entailment) category in the treemap (see Fig. 8(a)), the histogram (Fig. 8(b)) and scatterplot (Fig. 8(c)) are shown. The histogram shows the distribution of the prediction stability in the selected category. For each example, the stability is defined by the ratio of the number of perturbed pairs that maintain the same predicted label and all the perturbed pairs. Assuming we have generated 100 pairs via the automated sentence perturbation operation (i.e., replace nouns and verbs with synonyms), the stability is 0.8 if 80 of the 100 maintain the original label. We can further narrow down the focused set by selecting the bins in the histogram (see Fig. 8(b)). In the scatterplot (Fig. 8(c)(d)), each point corresponds to a sentence pair (the user can focus the rest of the visualization on one particular instance by selection). To help users better assess the *stability* number, we also include the number of perturbed pairs (labeled as *perturbCount*). If the *perturbCount* is rather small ($< 10$), then the *stability* value is likely very noisy and unreliable.

### 5.6 Implementation

The initial learning curve and workflow setup cost of the tool are often the most significant barriers for user adaptation. In the proposed system, we approach these challenges by designing the system as a Python library rather than as a monolithic standalone application. Just like a

Python plotting library, the different pieces of the visualization can be accessed individually, which helps ease the initial learning curve. The individual components can also be combined in any configuration desired by users via a simple Python API to better fit into one's workflow. More importantly, the library-based design allows easy integration with the existing model implemented in Python. To create a visualization, users only need to import the library, create an instance of the visualization object, and specify a set of callback functions, such as generating a prediction and accessing attention, to link the visualization to their NLP models (see the code example in **Appendix B**).

In the proposed work, the NLI model is implemented in Python using *pytorch* [25]. The visual interface is implemented in Javascript using *D3.js* library, and a Python server acts as the glue between the Javascript visualization and the *pytorch* model.

## 6 Application Scenarios

To illustrate how the proposed perturbation-driven exploration tool helps researchers interpret the neural network model, we present five interconnected application scenarios domain experts may employ in their analysis workflow. Users can start the exploration by examining the stability of prediction (Scenario 1), from which they may identify the individual instances worth further investigation (Scenario 2, 3, 4). Alternatively, users can begin with handcrafted common/extreme cases (Scenario 5) and continue from there.

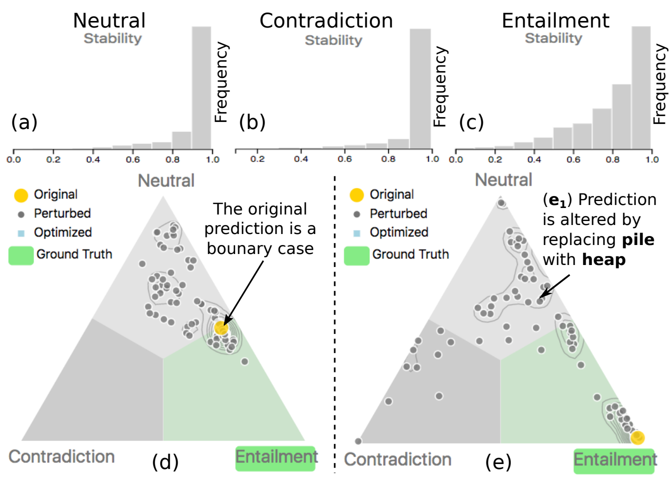


Fig. 9. Prediction stability assessment. In (a)(b)(c), we estimate the overall prediction stability (regarding synonymous perturbation) for each type of prediction over the entire development set (10k examples). The user can drill down to individual examples by using the interface described in Fig. 8. In (d), we illustrate a highly unstable prediction, where the original pair's prediction is near a decision boundary (i.e., the yellow circle is between *entailment* and *neutral*). In (e), we show another example of unstable predictions, in which the prediction changes drastically with a minor perturbation (see ($\mathbf{e_1}$)).

### 6.1 Scenario 1: Assess the Model Prediction Stability

The robustness of the prediction is often hard to evaluate. However, the prediction stability provides valuable information for researchers to better understand the model. In the proposed work, we approach the prediction robustness from a sensitivity analysis point of view. The stability of the prediction is measured by how often the predicted labels are altered after small perturbations are applied to the input. Compared to other types of input (e.g., image), the perturbation of the natural language can be particularly tricky, as small alterations of words can drastically change the meaning of the sentence. As discussed in Section 5.1, we try to maintain the sentence semantic by replacing only words with their synonyms and only one word for each pair. As illustrated in Fig. 9, by utilizing the proposed tool, the domain expert can not only examine a visual summary of the stability but also quickly dive into individual examples for a case-by-case analysis.

In Fig. 9(a)(b)(c), we compare the overall prediction stability (regarding synonymous perturbation) for all correct predictions in the development set (10k examples in total). We observe a drastic difference for the stability for *entailment* predictions compared to the *contradiction* and *neutral* ones. Such a distinction can be partially explained by how the *entailment* relationship is defined. The relationship is valid only if the concept in the premise is more specific than the concept in the hypothesis. Therefore, the synonymous perturbation may change the *entailment* relationship, as the replaced *noun* or *verb* can be more or less restrictive compared to the original. This inherent disparity of sensitivity may warrant extra consideration when designing future NLI models.

Besides presenting the summary view, the tool also allows the user to quickly narrow down the selection to a single example by filtering via the histogram and scatterplot (see details in Fig. 8). Through the exploration of many samples with low stabilities, domain experts notice that many highly unstable outliers are from sentence pairs where the predictions are near the decision boundary (see Fig. 9(d): the yellow circle corresponds to a *entailment* prediction that is very close to *neutral*). However, we can also find sentence pairs, such as the one illustrated in Fig. 9(e), in which the prediction is altered drastically with minor perturbations (e.g., replace the word **pile** with **heap** in "pile of snow"). In the following section, we examine what happened inside the model and hypothesize the cause of the failure (see Fig. 10).

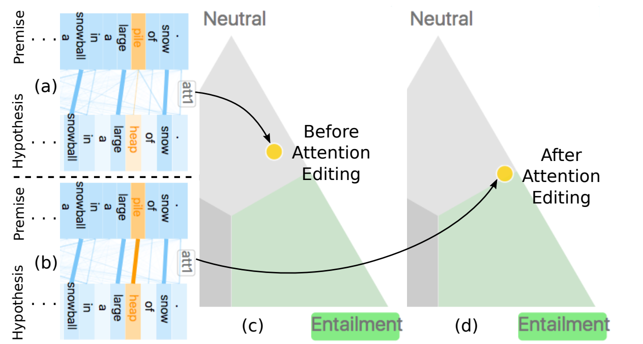


Fig. 10. Editing the original attention (a) to correctly align the word "heap" with "pile" as shown in (b) (these two words are highlighted in orange). The change of attention leads to the change of prediction from neutral to the class boundary between *neutral* and *entitlement*.

### 6.2 Scenario 2: Examine the Decision-Making Process

The predicted label alone provides limited information. Often, domain experts want to know how the model arrives at a conclusion, and if the prediction is incorrect, where in the model the error occurs. Examining the decision-making process is not only instrumental in evaluating the model performance but also essential for hypothesizing improvement strategies for future models. In the NLI model, the three stages (encoder, attention, classifier) work in synergy to produce the prediction. Therefore, making sense of the prediction involves understanding how different parts of the model affect the final prediction.

In the previous section, we have noticed that a minor perturbation of the sentence may result a change in the final prediction (Fig. 9(e)). Here, we want to make sense of what leads to the failed prediction. In this example, the premise **P** is "A very young child in a red plaid coat and pink winter hat makes a snowball in a large **pile** of snow", and the original hypothesis **H1** is "A child in a red plaid coat and pink winter hat makes a snowball in a large **pile** of snow". The perturbed hypothesis **H2** replaces the word **pile** with **heap** in **H1**. This example should be rather straightforward for the model since there are only minor differences between **P** and **H1/H2**.

As illustrated in Fig. 10(a), based on the graph attention visualization, we can see in the attention for (**P**, **H2**) pair that the words **pile** and **heap** are not well aligned. To test whether the alignment is what contributes to the misclassification, the domain expert utilizes the attention editing functionality in the matrix attention view (Fig. 5(c)) to make the word **pile** align with **heap** (shown in Fig. 10(b)). After the edit, the original prediction (Fig. 10(c)) has been moved from neutral to the classification boundary (Fig. 10(d)), but the corrected attention fails to produce a conclusive *entailment* prediction (another example, in which the

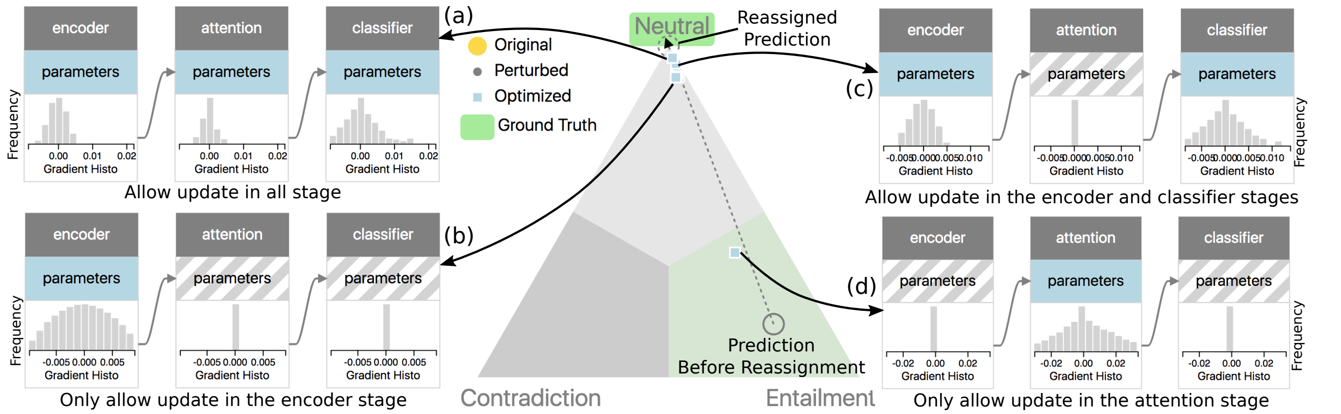


Fig. 11. Experiment with all configurations for the label reassignment optimization. As shown in (d), the update to the attention stage seems to have significantly less impact on the prediction result compared to the classifier or encoder stage of the model.

attention edit corrects the final prediction, is shown in Fig. 12). Such an observation implies that the model does not firmly believe "**heap** of snow" and "**pile** of snow" have the same meaning, which may indicate a potential issue with the encoder or word embedding.

We have shown that the perfect alignment does not necessarily guarantee a correct prediction. On the flip side, we may produce the right predictions for the "wrong" reason (i.e., incorrect attention). For example, in the sentence pair (**P**: "A couple is taking a break from bicycling", **H**: "sisters sit next to their bikes"), the words **take** and **next** should not align with each other, yet the model still predict the correct label (*neutral*).

### 6.3 Scenario 3: Update the Model to Correct a Prediction

Up to here, we have employed perturbation for input to understand prediction robustness and utilized the perturbation of attention (and input) to infer the model decision-making process. Both these perturbation operations rely on forward propagation in the pipeline and assume the model parameter remains unchanged. However, once we get a sense of how predictions are made and hypothesize about the cause of the failure in the case of a prediction error, it is natural to ask follow-up questions: What does it take to fix an incorrect prediction? And more importantly, what role does each of the three stages play in such a process? And, are they affecting the prediction differently?

Domain experts can obtain answers to these questions by utilizing the prediction and pipeline view in the proposed tool. As discussed in detail in Section 5.4, we employed a margin-infused relaxed algorithm (MIRA) based optimization with two objectives (apply the least amount of change to the parameter, and make the new prediction as close to the reassigned prediction as possible) to update the network parameters. We then visualize how much each stage of the model is changed through the distribution of differences between the two sets of parameters (see Fig. 7).

To infer the role each stage of the pipeline plays, the proposed tool allows the parameter update to be enabled or disabled for each pipeline stage. The system also includes an automatic option to test all the possible configuration combinations. As illustrated in Fig. 11, the ground truth for this sentence pair is *neutral*. However, the model produces an incorrect label *entailment*. The domain expert reassigns the prediction to *neutral*, which triggers the prediction update optimization for seven different pipeline configurations. Four configurations are shown in Fig. 11(a)(b)(c)(d). The updated predictions are illustrated as blue squares, and the arrowed lines highlight the corresponding pipeline configurations.

Interestingly, all configurations except one are concentrated around the full *neutral* prediction. Referring back to the pipeline visualization, we observe that the only configuration that failed to produce the correct label is the one for which we allow only updating of the attention stage of the model. The domain experts find this observation very interesting and suggest a preliminary interpretation. They believe the attention provides a way to compose word semantics. Individual word semantics are yielded from the encoder and input embedding, whereas the composed semantics participate in the classification layer. From this analysis, we can extrapolate that the ultimate decision relies on the semantics (i.e., encodings and composed encodings in the classifier). The encoder and classifier layer can swiftly adjust weights for word/composed semantics to correct prediction. However, the attention layer, which only works on how individual words are aligned, affects the prediction less significantly. Recent word embeddings works (e.g., ELMo [30]) also support such an observation that better word embeddings can substantially benefit a model. However, the result by no means implies attention is not useful, because it serves as a way to compose word semantics.

### 6.4 Scenario 4: Explore the Relationship Between Grammar and Attention

The attention computation in the NLI model does not take the grammar structure of the sentences into consideration, yet the attention often highlights key elements of the sentence. Therefore, domain experts wish to understand whether attention alone is sufficient to capture sentence structure; and, more importantly, what kind of additional information from grammar parsing can help address the NLI challenge.

In the proposed system, we overlay the sentence dependency tree with the attention, which enables researchers to conduct comparisons between attention and grammar structure. As illustrated in Fig. 12(a), the prediction of the sentence pair (**P**: "A woman in a green jacket is drinking tea." **H**: "A woman is drinking green tea.") is wrong. We can infer the cause by examining the attention, in which the word **green** in "**green** jacket" is aligned to the **green** in "**green** tea". Due to such an alignment, the model mistakenly believes the two **greens** are used to describe the same thing (therefore, predict *entailment*). However, as we examine the dependency tree, the two **greens** are attached to different words, i.e., **green** in **P** is attached to "jacket", whereas **green** in **H** is attached to "tea". Therefore, they should not be aligned to allow the classifier to make the right decision. This experiment demonstrates the potential benefit of including grammar structure in the alignment computation. Interestingly, as illustrated in Fig. 12(b), by editing the attention and forcing the alignment of the two **greens** to be zero, the prediction label is corrected (*neutral*).

### 6.5 Scenario 5: Handcrafted Example Exploration

To test the limits of the model, domain experts often handcraft "extreme" examples (such as the Facebook IPO example discussed in Section 2.1) for which they know most models will have difficulty making a correct inference. The researchers start with a set of experiments they plan to run, from which they will develop new hypotheses for further analysis. We can think of such a process as a natural blend of all previously discussed scenarios. However, instead of having a specific goal in mind, the domain experts focus on probing around to uncover any interesting or out-of-the-ordinary behaviors in the model.

## 7 Evaluation and Feedback

As discussed previously, we have worked closely with NLP experts during the development of the tool. However, since these two NLP

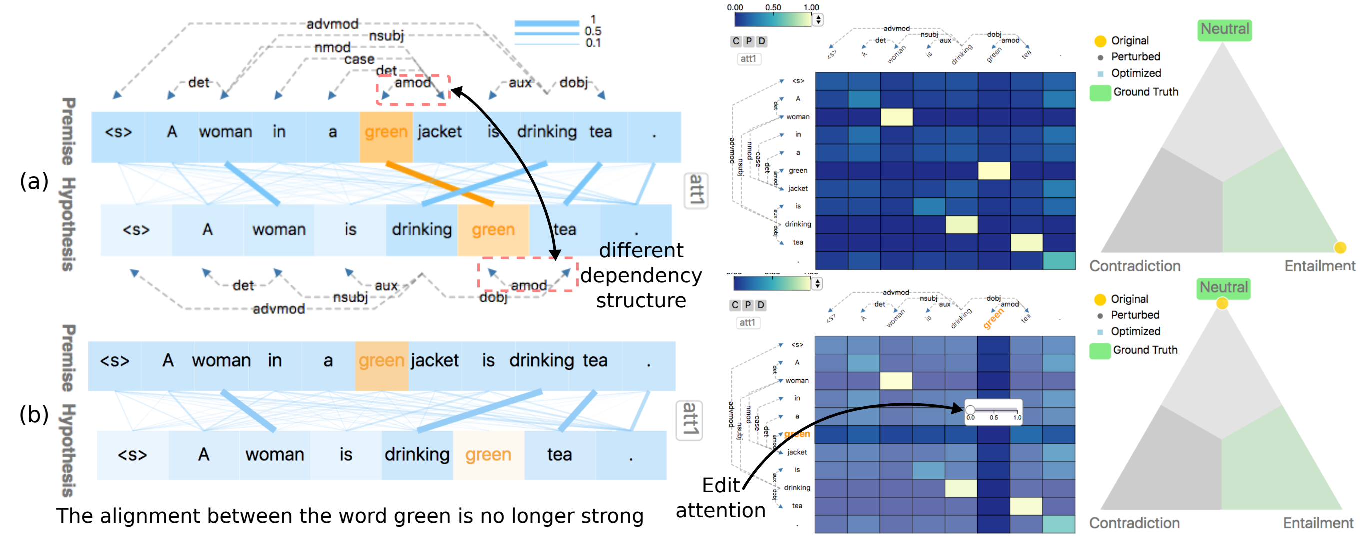


Fig. 12. The dependency tree provides valuable information that can help fix the prediction error. In (a), the model mistakenly aligns the word green, which leads to an incorrect prediction. After examining the dependency tree (highlighted by pink squares), we can see the two **greens** are attached to different words. In (b), by editing the attention and forcing the alignment of the two **greens** to be zero, the prediction label is corrected to *neutral*.

experts have been heavily involved in the design process, they may not be the best candidates for identifying potential issues of the tool due to familiarity. To uncover the limitation and identify areas for improvement, we gathered a wider audience from either visualization (four researchers, including three Ph.D.-level students and one postdoc researcher) or NLP backgrounds (five Ph.D.-level students who are familiar with the concept of natural language inference and attention) for obtaining feedback for the tool.

With the goal to identify the potential interface design issues, we first conducted an informal demonstration-and-feedback session with the visualization group. We started by explaining the basic natural language inference concept. We then demonstrated the features of the tool in detail. After that, we answered questions and sought feedback. From this session, we have gathered complaints and suggestions on various aspects of the visual interface. Some of the identified issues include: (1) difficult to distinguish different types of predictions (distinct colors are added to the final version); (2) hard to recognize the ground truth label (we now use a green rectangle to indicate the ground truth); (3) lack of legends to understand the key elements in plots (legends are added). We address these interface issues before presenting the final version to the NLP group.

We conducted an individual revaluation session with each participant from the NLP group, in which the participant was given 30 minutes to experiment with the tool after an overall feature demonstration. Since both the matrix and graph-based visual encodings were the most common attention representations used in the NLP literature, most participants can utilize them immediately and find the linked highlighting feature of the two views quite useful. Once the participants become familiar with the tool, they often try to type two similar sentences and examine the attention and prediction. After that, they will modify some words, or negate the hypothesis sentence, and then check the attention and prediction again. Such a "perturb and observe" operation demonstrates the most fundamental exploration strategy and matches well to the perturbation-driven paradigm the proposed tool aims to support.

One participant shows us two examples after a quick exploration session. In the first example, he identifies a case where the wrong attention alignment produces a correct final prediction. In the other example, he finds a sentence pair with the incorrect attention that produces a wrong prediction. However, even after he forces the correct alignment, the prediction result remains incorrect. He believes observations such as these will provide valuable insights for him to interpret the model. Another participant comments that the ability to enable or disable the model parameter update in the pipeline view is beneficial.

The participants also identify potential issues and places for improvement. One participant wishes the pipeline view could provide more detailed information compared to the current aggregated histogram visualization. Another participant suggests the possibility to examine multiple similar examples at a time (instead of one by one). Also, for the participants who do not focus on NLI research, understanding all the views at first can be a bit challenging. However, this issue can be addressed by the modular design, as the user can simply enable only the sentence, attention, and prediction views. Overall, all the participants believe the proposed tool is very convenient for conducting experiment and exploring various hypotheses, which is essential for building intuitions about the model.

## 8 Discussion

The current setup for the proposed tool is suitable only for the natural language inference task. However, due to the modular design and the many shared attributions among the end-to-end NLP models, we can readily extend the system to handle other tasks. In the future, we plan to open source our projection and add support for more NLP tasks, such as neural machine translation and question-and-answer. From our evaluation process, we found that the quality of automatic perturbed sentences can be a potential limitation of the tool, since we rely on WordNet to generate the perturbation, which has a rather inclusive definition for synonymous. Often, we can identify perturbed sentences that are not particularly meaningful (e.g., sentences with very obscure words or usages). Even if all the words are meaningful, currently, there is no way we can verify whether the perturbed sentence is valid natural language composition or not. However, from the natural language process point of view, the perturbation of a sentence while maintaining semantic and correct grammar is an open research area on its own.

To conclude, this work introduces a perturbation-driven visual interrogation system that provides experts with a streamlined exploration environment for testing hypotheses and obtaining intuition about the model. The proposed system frees the researchers from interruptions and tedious operations, thereby allowing them to focus on more productive activities.


## Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work is also supported in part by NSF: CGV: Award:1314896, NSF:IIP Award :1602127 NSF:ACI:award 1649923, DOE/SciDAC DESC0007446, CCMSC DE-NA0002375, PIPER: ER26142 DE-SC0010498, and NVIDIA Corporation. This material is based upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number(s) DE-NA0002375.

## REFERENCES


[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, vol. 16, pp. 265–283, 2016.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *IEEE transactions on visualization and computer graphics*, 24(1):152–162, 2018.

[4] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[5] D. Cashman, G. Patterson, A. Mosca, and R. Chang. Rnnbow: Visualizing learning via backpropagation gradients in recurrent neural networks. In *Workshop on Visual Analytics for Deep Learning (VADL)*, 2017.

[6] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3(Jan):951–991, 2003.

[7] I. Dagan, D. Roth, M. Sammons, and F. M. Zanzotto. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220, 2013.

[8] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97, 2018.

[9] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[10] J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 5686–5697. ACM, 2016.

[11] J. Li, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[12] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu. Analyzing the training processes of deep generative models. *IEEE transactions on visualization and computer graphics*, 24(1):77–87, 2018.

[13] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2017.

[14] S. Liu, P.-T. Bremer, J. J. Thiagarajan, V. Srikumar, B. Wang, Y. Livnat, and V. Pascucci. Visual exploration of semantic relationships in neural word embeddings. *IEEE transactions on visualization and computer graphics*, 24(1):553–562, 2018.

[15] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4768–4777, 2017.

[16] C. Ma, R. V. Kenyon, A. G. Forbes, T. Berger-Wolf, B. J. Slater, and D. A. Llano. Visualizing dynamic brain networks using an animated dual-representation. In *Proceedings of the Eurographics Conference on Visualization (EuroVis)*, pp. 73–77, 2015.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[18] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[19] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. *arXiv preprint arXiv:1710.10777*, 2017.

[20] J. Nivre. Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32, 2005.

[21] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. doi: 10.23915/distill.00007

[22] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks. doi: 10.23915/distill.00010

[23] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255. Association for Computational Linguistics, Austin, Texas, November 2016.

[24] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[26] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[27] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):98–108, 2018.

[28] M. T. Ribeiro, S. Singh, and C. Guestrin. 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*, pp. 1135–1144. ACM, 2016.

[29] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[30] S. Salant and J. Berant. Contextualized word representations for reading comprehension. *arXiv preprint arXiv:1712.03609*, 2017.

[31] I. Schwartz, A. Schwing, and T. Hazan. High-order attention models for visual question answering. In *Advances in Neural Information Processing Systems*, pp. 3667–3677, 2017.

[32] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *ICLR 2017*, 2016.

[33] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[34] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676, 2018.

[35] F.-Y. Tzeng and K.-L. Ma. Opening the black box-data driven visualization of neural networks. In *Visualization, 2005. VIS 05. IEEE*, pp. 383–390. IEEE, 2005.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 6000–6010, 2017.

[37] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mané, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12, 2018.

[38] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

[39] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[40] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.