AllenNLP Interpret: Explaining Predictions of NLP Models

Eric Wallace
Allen Institute for AI

Jens Tuyls
University of California, Irvine

Junlin Wang
University of California, Irvine

Sanjay Subramanian
Allen Institute for AI

Matt Gardner
Allen Institute for AI

Sameer Singh
University of California, Irvine

Abstract

Neural NLP models are increasingly accurate but are imperfect and opaque—they break in counterintuitive ways and leave end users puzzled at their behavior. Model interpretation methods ameliorate this opacity by providing explanations for specific model predictions. Unfortunately, existing interpretation codebases make it difficult to apply these methods to new models and tasks, which hinders adoption for practitioners and burdens interpretability researchers. We introduce AllenNLP Interpret, a flexible framework for interpreting NLP models. The toolkit provides interpretation primitives (e.g., input gradients) for any AllenNLP model and task, a suite of built-in interpretation methods, and a library of front-end visualization components. We demonstrate the toolkit’s flexibility and utility by implementing live demos for five interpretation methods (e.g., saliency maps and adversarial attacks) on a variety of models and tasks (e.g., masked language modeling using BERT and reading comprehension using BiDAF). These demos, alongside our code and tutorials, are available at https://allennlp.org/interpret. A video that walks through various use cases of our toolkit is available as well.

1 Currently Available Models

The toolkit currently interprets six tasks that cover a wide range of input-output formats and models:

- **Masked Language Modeling** using the transformer models available in Pytorch Transformers [2], e.g., BERT [1], RoBERTa [5], XLNet [11], and more.
- **Text Classification** and **Textual Entailment** using BiLSTM and self-attention classifiers.
- **Named Entity Recognition (NER)** and **Coreference Resolution** using LSTM-CRF taggers.

These tasks have complex input-output structure—our toolkit handles these out of the box.

2 Interpreting Model Predictions

We introduce an end user’s view of our toolkit: the interpretations methods, models, and visualizations.

**Saliency Map Visualizations** We consider three saliency methods. Since our goal is to interpret why the model made its prediction (not the ground-truth answer), we use the model’s own output in the loss calculation. For each method, we reduce each token’s gradient (which is the same dimension as the token embedding) to a single value by taking the $L_2$ norm.

Preprint. Under review.
Figure 1: An interpretation generated using AllenNLP Interpret for NER. The model predicts three tags for an input (top). We interpret each tag separately, e.g., input reduction (bottom) removes as many words as possible without changing a tag’s prediction. Input reduction shows that the words “named”, “at”, and “in downtown” are sufficient to predict the People, Organization, and Location tags, respectively.

![Figure 1](image1.png)

Figure 2: A saliency map generated using Vanilla Gradient for BERT’s masked language modeling objective. BERT predicts the [MASK] token given the input sentence; the interpretation shows that BERT uses the gendered pronoun “her” and the hospital-specific “emergency” to predict “nurse”.

- **Vanilla Gradient**: This method visualizes the gradient of the loss with respect to each token [8]. Figure 2 shows an example interpretation of BERT [1].
- **Integrated Gradients**: This method defines a baseline $x'$, which is an input absent of information (we use a sequence of all zero embeddings). Word importance is determined by integrating the gradient along the path from this baseline to the original input [10].
- **SmoothGrad**: This technique averages the gradient over many noisy versions of the input [9]. We add small Gaussian noise to every embedding and take the average gradient value.

Adversarial Attacks We consider two adversarial attacks: replacing words to change the model’s prediction (HotFlip [3]) and removing words to maintain the model’s prediction (Input Reduction).

- **Untargeted & Targeted HotFlip** HotFlip uses the gradient to swap out words from the input in order to change the model’s prediction. It answers a sensitivity question: how would the prediction change if certain words are replaced? We also extend HotFlip to a targeted setting, i.e., we substitute words in order to change the model’s prediction to a specific target prediction. This answers an almost counterfactual question: what words should be swapped in order to cause a specific prediction?
- **Input Reduction** This technique removes as many words as possible from the input without changing a model’s prediction [4]. Input reduction works by iteratively removing the word with the smallest gradient value. Figure 1 shows an example of reducing an NER input.

3 Adding a Model or Interpretation

**New Interpretation** We provide a tutorial for adding a new analysis method to our toolkit. It walks through the three main requirements for adding SmoothGrad.

**New Model** We also provide a tutorial for interpreting a new model. If your task is already available in the demos (e.g., text classification), you need to change a single line of code to replace the demo model with your model. If your task is not present in the demos, you will need to follow three easy steps to add your task to the set of demos.

---

The tutorials can be found at https://allennlp.org/interpret
References


