AllenNLP Interpret:
A Framework for Explaining Predictions of NLP Models

-Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, Sameer Singh

January 24, 2020

Presenter: Rishab Bamrara

https://qdata.github.io/deep2Read/

Motivation

- State-of-the-art models for NLP are imperfect and significantly underperform humans on a myriad of tasks.
- These imperfections leave us with a question, “why did model made this prediction?”
- Interpretations are useful to illuminate strengths and weaknesses of a model, increase user trust, and evaluate hard-to-define criteria such as safety and fairness.
- However, most codebases on computer vision are model or task-specific (sentiment analysis), or contain implementations for a small number of interpretation methods.
- As a result, existing interpretation codebases are hard to adopt for practitioners and burdens interpretability researchers.

Background

- **Saliency maps** explain a model's prediction by identifying the importance of the input tokens.

- **Gradient-based methods** determine the importance using the gradient of the loss with respect to the input tokens.
  - **Vanilla Gradient** visualizes the gradient of the loss with respect to each token.
  - **Integrated Gradients** define a baseline x' and determine the word importance by integrating the gradient along the path from the baseline to the input.
  - **SmoothGrad** averages the gradient over many noisy versions of the input.

- Adversarial attacks change the input itself.
  - **HotFlip** replaces the words to change the model's prediction.
  - **Input Reduction** means removing words to maintain the model's prediction.

Background (Contd.)

- **Targeted HotFlip (Extension):** What words should be swapped in order to cause a specific prediction.
- **Untargeted HotFlip:** How would the prediction change if certain words are replaced.
- **Reading Comprehension:** Given a passage, answers questions about it. SQuAD and DROP are generally used.
- **Masked Language Modelling:** Mask one or more words in a sentence and make the model predict those words. BERT is commonly used.
- **Text Classification:** Classify a given piece of text into some predefined classes. BiLSTM is used.
- **Textual Entailment:** Directional relation between text fragments. T entails H (T=>H) means on the basis of T, H is likely to be true. Self-attention models are generally used.

Background (Contd.)

- **Named Entity Recognition (NER):** Locate and classify a named entity in the text. Done using Input Reduction.
- **Coreference Resolution:** Finding all expressions that refer to the same entity in the text. E.g. I, me and Alfred, he, his, him etc.

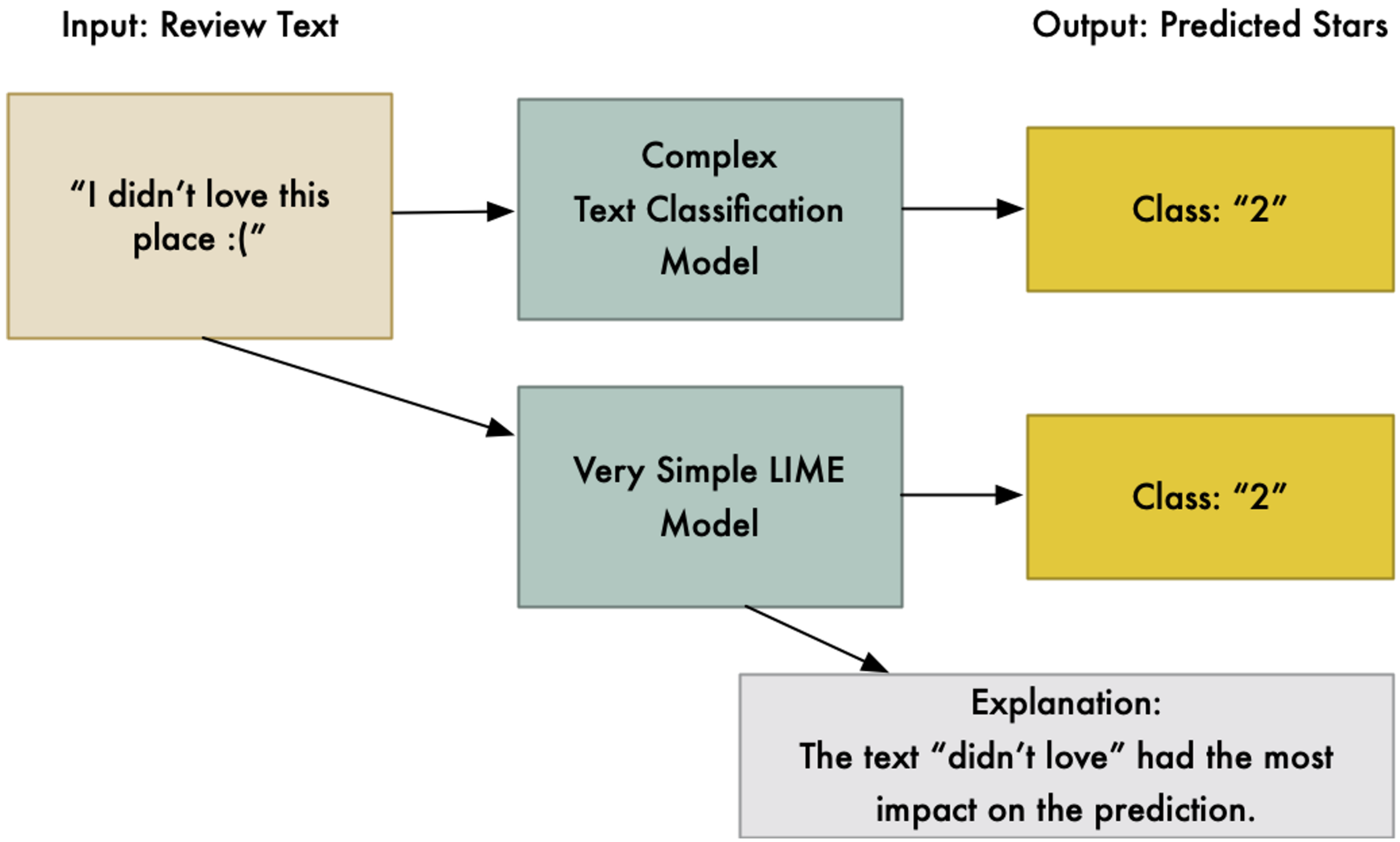
Related Work

- **Alternative Interpretation Methods:**
  - Bahdanau et al., 2015 : Visualization of attention weights
  - Karpathy et al., 2016: Isolate the effect of individual neurons
- **Existing Interpretation Toolkits in Computer Vision:**
  - Papernot et al., 2016
  - Norton and Qi: Interactive demos
  - Liu et al., 2018, Strobelt et al. 2019 and Vig 2019: Visualization of attention weights
  - Lee et al., 2019: Adversarial attacks to reading comprehension systems

Claim / Target Task

Create an extensible toolkit for interpreting NLP models. Make an easy to apply existing interpretation methods to new models as well as develop new interpretation methods.

An Intuitive Figure Showing WHY Claim


Input: Review Text
Output: Predicted Stars
"I didn't love this place :("
Complex Text Classification Model
Class: "2"
Very Simple LIME Model
Class: "2"
Explanation: The text "didn't love" had the most impact on the prediction.

https://medium.com/@ageitgey/natural-language-processing-is-fun-part-3-explaining-model-predictions-486d8616813c

Proposed Solution

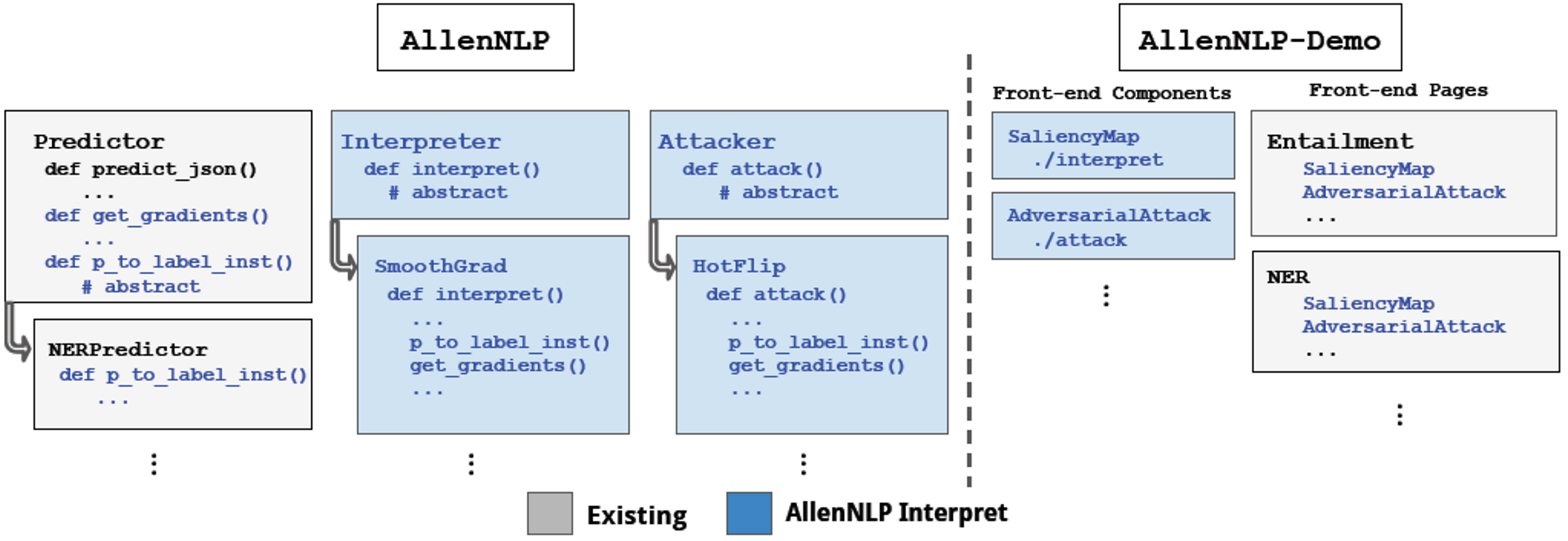


Figure 4: **System Overview:** Our toolkit (in blue) and the surrounding AllenNLP ecosystem. The only model-specific code is a simple function called predictions_to_labeled_instances() (abbreviated as p_to_label_inst()), which is added to the model's Predictor class (e.g., for an NER model's predictor; left of figure). This function allows input gradients to be calculated using get_gradients() in a model-agnostic manner (e.g., for use in SmoothGrad or HotFlip; middle left of Figure). On the front-end (right of Figure), we create reusable visualization components, e.g., for visualizing saliency maps or adversarial attacks.

Implementation (Existing NLP Models)

- Models in AllenNLP include a forward() function to run the model as well as compute the loss if a label is provided (supervised).

- For obtaining results there is a Predictor class. AllenNLP calls predict_json() with a JSON containing raw strings (input) to get the model’s prediction.

- Predictor class also computes input gradients in a model-agnostic way. In case when there are widely varying outputs, AllenNLP leverages the fact that forward() gives the loss if given a label.

- Firstly, it gets the prediction and then converts the prediction into a set of labelled examples using predictions_to_labelled_instances(). Each instance is used to compute loss for a different part of output.

- In case of multiple embeddings, AllenNLP computes gradient by registering a PyTorch backward gradient hook on the model's TokenEmbedder function.

API => call predictions_to_labelled_instances() and then get_gradients()

Implementation (Existing NLP Models)

- Context-Dependent Embedding Matrices such as the ones used in ELMo and BERT, creates problem for HotFlip operation which requires searching over a discrete embedding matrix.

- Create context-independent matrix containing features from model's last context-independent layer. E.g. for ELMo save the features from its context independent Char-CNN into a matrix.

- For Visualization, AllenNLP Demo has HTML and JavaScript components for visualizing saliency maps and adversarial attacks.

Implementation (Adding Interpretation)

1. Implement the new Interpretation into AllenNLP using “predictions_to_labelled_instances()” and “get_gradients()”.

1. Add the new Interpreter to the demo back-end.

1. Add the frontend HTML/JavaScript for saliency visualization. Can make a one-line call for reusable front-end components.

Implementation (Adding New Model)

1. Implement “predictions_to_labelled_instances()” for the new model.

1. Add the new model’s path to the demo back-end.

1. Add the frontend HTML/JavaScript for saliency visualization. Can make a one-line call for reusable front-end components.

Data Summary

- **General English Sentences.**

Experimental Results and Analysis

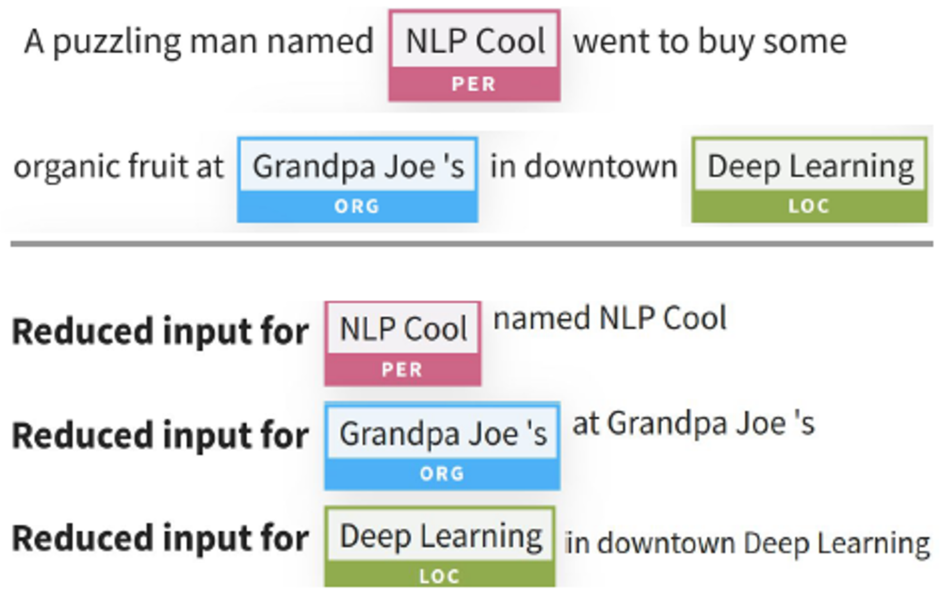
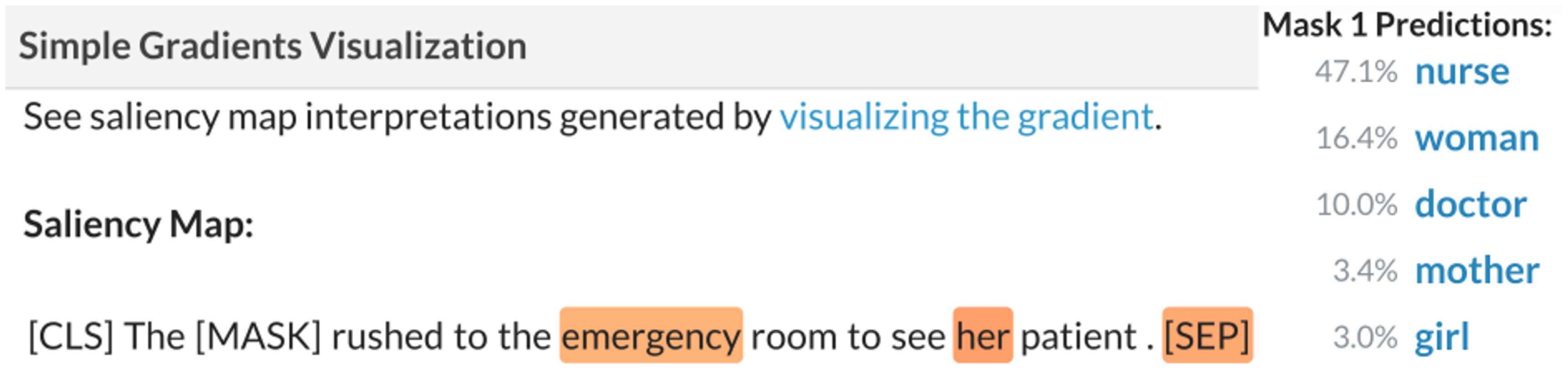


Figure 1: An interpretation generated using AllenNLP Interpret for NER. The model predicts three tags for an input (top). We interpret each tag separately, e.g., input reduction (Feng et al., 2018) (bottom) removes as many words as possible without changing a tag's prediction. Input reduction shows that the words "named", "at", and "in downtown" are sufficient to predict the People, Organization, and Location tags, respectively.

Experimental Results and Analysis


Simple Gradients Visualization

See saliency map interpretations generated by visualizing the gradient.

Saliency Map:

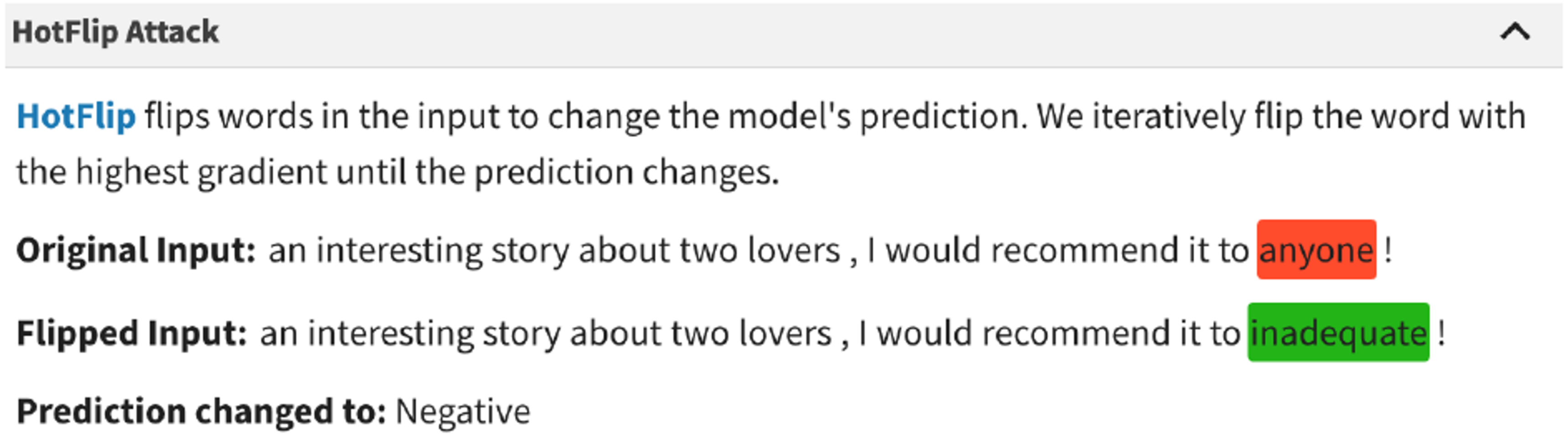
[CLS] The [MASK] rushed to the emergency room to see her patient . [SEP]

Mask 1 Predictions:

47.1%	nurse
16.4%	woman
10.0%	doctor
3.4%	mother
3.0%	girl

Figure 2: A saliency map generated using Vanilla Gradient (Simonyan et al., 2014) 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".

Experimental Results and Analysis


HotFlip Attack

HotFlip flips words in the input to change the model's prediction. We iteratively flip the word with the highest gradient until the prediction changes.

Original Input: an interesting story about two lovers , I would recommend it to anyone !

Flipped Input: an interesting story about two lovers , I would recommend it to inadequate !

Prediction changed to: Negative

Figure 3: A word-level HotFlip attack on a sentiment analysis model—replacing "anyone" with "inadequate" causes the model's prediction to change from Positive to Negative.

Conclusion and Future Work

AllenNLP Interpret toolkit facilitates the interpretation of NLP models. The toolkit is flexible i.e. it enables the development and evaluation of interpretation methods across a wide range of NLP models and tasks.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
- Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078, 2015.
- Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, et al. 2016. Technical report on the CleverHans v2.1.0 adversarial examples library. arXiv:1610.00768.
- Andrew P Norton and Yanjun Qi. 2017. Adversarial-Playground: A visualization suite showing how adversarial examples fool deep learning. In 2017 IEEE VizSec Symposium.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized Bert pretraining approach. arXiv:1907.11692.
- Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. 2019. Seq2Seq-Vis: A visual debugging tool for sequence-to-sequence models. IEEE TVCG.
- Jesse Vig. 2019. Visualizing attention in transformer based language models. arXiv:1904.02679.
- Gyeongbok Lee, Sungdong Kim, and Seung-won Hwang. 2019. QADiver: Interactive framework for diagnosing QA models. In AAAI Demonstrations