T^3-Vis: a visual analytic framework for Training and fine-Tuning Transformers in NLP

Raymond Li[†], Wen Xiao[†], Lanjun Wang[‡*], Hyeju Jang[†], Giuseppe Carenini[†]
[†]University of British Columbia, Vancouver, Canada
{raymondl, xiaowen3, hyejuj, carenini}@cs.ubc.ca
[‡]Huawei Cananda Technologies Co. Ltd., Burnaby, Canada
lanjun.wang@huawei.com

Abstract

Transformers are the dominant architecture in NLP, but their training and fine-tuning is still very challenging. In this paper, we present the design and implementation of a visual analytic framework for assisting researchers in such process, by providing them with valuable insights about the model's intrinsic properties and behaviours. Our framework offers an intuitive overview that allows the user to explore different facets of the model (e.g., hidden states, attention) through interactive visualization, and allows a suite of built-in algorithms that compute the importance of model components and different parts of the input sequence. Case studies and feedback from a user focus group indicate that the framework is useful, and suggest several improvements. Our framework is available at: `https://github.com/raymondzmc/T3-Vis`.

1 Introduction

Approaches through neural networks have made significant progress in the field of NLP, with Transformer models (Vaswani et al., 2017) rapidly becoming the dominant architecture due to their efficient parallel training and ability to effectivelymodel long sequences. Following the release of BERT (Devlin et al., 2019) along with other Transformer-based models pretrained on large corpora (Liu et al., 2019; Lewis et al., 2020; Joshi et al., 2020; Lee et al., 2020), the most successful strategy on many NLP leaderboards has been to directly fine-tune such models on the downstream tasks (e.g., summarization, classification). However, despite the strong empirical performance of this strategy, understanding and interpreting the training and fine-tuning processes remains a critical and challenging step for researchers due to the inherent black-box nature of neural models (Kovaleva et al., 2019; Hao et al., 2019; Merchant et al., 2020; Hao et al., 2020).

Generally speaking, a large number of visual analytics tools have been shown to effectively support the analysis and interpretation of deep learning models (Hohman et al., 2018). For instance, to remedy the black-box nature of neural network hidden states, previous work has used scatterplots to visualize high dimensional vectors through projection techniques (Smilkov et al., 2016; Kahng et al., 2017), with Aken et al. (2020) visualizing the differences of token representations from different layers of BERT (Devlin et al., 2019). Similarly, despite some limitations regarding the explanatory capabilities of the attention mechanism (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019), the visualization of its weights has also been shown to be beneficial in discovering learnt features (Clark et al., 2019; Voita et al., 2019), with promising recent work focusing on Transformers (Vig, 2019; Hoover et al., 2020).

Besides the works on exploring what has been learnt in the pretrained models, there are also several visualization tools developed to show saliency scores generated by gradient-based (Simonyan et al., 2013; Bach et al., 2015; Shrikumar et al., 2017) or perturbation-based interpretation methods (Ribeiro et al., 2016; Li et al., 2016), which can help with visualizing the relative importance of individual tokens in the input with respect to a target prediction (Wallace et al., 2019; Johnson et al., 2020; Tenney et al., 2020). However, only a few studies have instead focused on visualizing the overall training dynamics, where support is critical for identifying mislabeled examples or failure cases (Liu et al., 2018; Xiang et al., 2019; Swayamdipta et al., 2020)

In essence, the framework we propose in this paper, namely T^3-Vis, synergistically integrates some of the interactive visualizations mentioned above to support developers in the challenging task of training and fine-tuning Transformers. This is in contrast with other similar recent visual tools (Ta-

*Corresponding author.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 220–230
November 7–11, 2021. ©2021 Association for Computational Linguistics

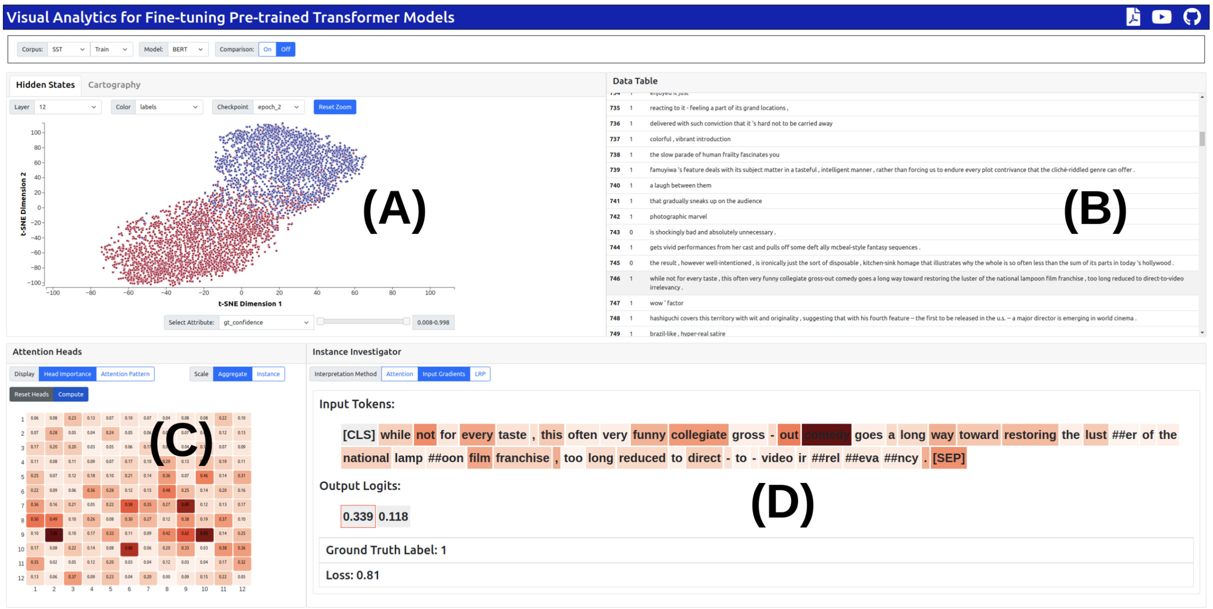


Figure 1: Overview of the interface: **(A)** Projection View provides a 2D visualization of the dataset by encoding each example as a point on the scatterplot; **(B)** Data Table allows the user to view the content and metadata (e.g. label, loss) of the data examples (e.g. document); **(C)** Attention Head View visualizes the head importance and weight matrices of each attention head; **(D)** Instance Investigation View allows the user to perform detailed analysis (e.g. interpretation, attention) on a data example's input sequence.

ble 1), which either only focus on single data point explanations for uncovering model bias (e.g., AllenNLP Interpret (Wallace et al., 2019)), or rely on failed examples to understand the model's behaviour (e.g., Language Interpretability Tool (LIT) (Tenney et al., 2020)).

Following the well-established Nested Model for visualization design (Munzner, 2009), we first perform an extensive requirement analysis, from which we derive user tasks and data abstractions to guide the design of visual encoding and interaction techniques. More specifically, the resulting T^3-Vis framework provides an intuitive overview that allows users to explore different facets of the model (e.g., hidden states, attention, training dynamics) through interactive visualization.

Our contributions are as follows: **(1)** An extensive user requirement analysis on supporting the training and fine-tuning of Transformer models, based on extensive literature review and interviews with NLP researchers, **(2)** the design and implementation of an open-sourced visual analytic framework for assisting researchers in the fine-tuning process with a suite of built-in interpretation methods that analyze the importance of model components and different parts of the input sequence, and **(3)** the evaluation of the current design from case studies with NLP researchers and feedback from a user focus group.

2 Visualization Design

The design of our T^3-Vis is based on the nested model for InfoVis design (Munzner, 2009).

2.1 User Requirements

To derive useful analytical user tasks, we first identify a set of high-level user requirements (UR) through interviews with five NLP researchers as well as surveying recent literature related to the interpretability and the fine-tuning procedures of pretrained Transformers. In the interviews, we prompt participants with the open-ended question of "*If a visualization tool is provided to speed up your development (fine-tuning pretrained Transformers), what information would you like to see and explore?*". Combining the interview results and insights from the literature review, we organize these findings into five high-level requirements, each highlighting a different facet of the model for visualization.

Hidden state visualization (UR-1): Support the exploration for hidden state representations

Frameworks	Components					Functions		
	Dataset	Embeddings	Head Importance	Attention	Training Dynamics	Interpretations	Pruning	Comparison
BertViz (Vig, 2019)				✓				
AllenNLP Interpret (Wallace et al., 2019)						✓		
exBERT (Hoover et al., 2020)	✓	✓		✓				
LIT (Tenney et al., 2020)	✓	✓		✓		✓		✓
InterperT (Lal et al., 2021)	✓	✓	✓	✓				
T^3-Vis	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison with other visual frameworks from recent work.

from the model.

Attention visualization (UR-2): Allow users to examine and explore the linguistic or positional patterns exhibited in the self-attention distribution for different attention heads in the model.

Attention head importance (UR-3): Enable users to investigate and understand the importance of the attention heads for the downstream task and the effects of pruning them on the model's behaviour.

Interpretability of models (UR-4): In addition to attention maps, support a suite of alternative explanation methods based on token input importance, thus allowing users to better understand the model behaviours during inference.

Training dynamics (UR-5): Assist users in identifying relevant data examples based on their roles in the training process.

2.2 Supported Tasks and Data Model

Based on these user requirements, we derive nine analytical tasks framed as information seeking questions . In Table 2, we list the tasks along with important attributes including: *When* they are relevant during the fine-tuning process, the *Granularity* of the data that it operates on, corresponding *User Requirements*, and the framework *Components* that it pertains to. We then look at the specific data to which the tasks are applied to. We characterize our data model (i.e. data types visualized by the interface) as comprising the model hidden states, the dataset examples along with their label/training features, the attention values, head importance scores, and input saliency map. Although our task and data models are derived for the fine-tuning of pretrained models, they can naturally be extended to training any Transformer models from scratch. Importantly, all the questions are invariant to any Transformer-based models for any downstream tasks (e.g. classification, sequence-generation or labeling).

2.3 T^3-Vis Components: Visual Encoding and Interactive Techniques

Projection View: To assist users in visualizing the model's hidden state representation (**UR-1**) and to identify the training role of the data examples (**UR-5**), we design the *Projection View* (Figure 1-(A)) as the main overview of our interface, and visualize the entire (or a subset of the) dataset on a 2D scatterplot, where each data point on the plot encodes a single data example (e.g. document) in the dataset. While the scatterplots can be generated in a variety of ways based on the user's needs, including dimension reduction methods (Wold et al., 1987; McInnes et al., 2018) and plotting based on training dynamics (Li et al., 2018; Toneva et al., 2019). Detailed studies examining the effectiveness of these methods in the context of visual analytics are out of the scope of this paper, but provide a promising direction for future work. In T^3-Vis, we provide two implementations (See Figure 2): (1) t-SNE projection (Van der Maaten and Hinton, 2008) of the model's hidden states, and (2) plotting the examples by their confidence and variability across epochs based on the Data Map technique (Swayamdipta et al., 2020). The color of the data points can be selected by the user via a dropdown menu to encode attributes of the data examples, where color saturation is used for continuous attributes (e.g. loss, prediction confidence), while hue is used for categorical attributes (e.g. labels, prediction). The user can also filter the data points by attributes, where a range slider is used for filtering the data points by continuous attributes, while a selectable dropdown menu is used to filter by categorical attributes. Furthermore, we also introduce a comparison mode by displaying the two scatterplots side-by-side, which allows for the flexibility of comparing across different checkpoints and the projection of different hidden state layers.

Data Table: The *Data Table* (Figure 1-(B)) lists

#	Question	When	Granularity	User Requirements	Components
T1	How to determine the model representation for a given NLP task?	Before	Dataset	1	Projection
T2	What are the outliers of the dataset?	Before, During, After	Dataset	1, 5	Projection
T3	What types of linguistic or positional attributes do the attention patterns exhibit for each attention heads?	Before, During, After	Instance	2	Projection
T4	Which attention heads are considered important for the task, and what are its functions?	After	Both	2, 3	Attention Head Instance Investigator
T5	How does pruning attention heads affects the model?	After	Instance	3	Attention Head
T6	How does the model changes at different stages of fine-tuning?	During, After	Both	1, 2, 3	All
T7	Does the model rely on specific parts of the input sequence when making predictions?	After	Instance	4	Instance Investigator
T8	Are there mislabeled examples in the dataset?	During, After	Both	1, 5	Projection
T9	How can the dataset be augmented to improve the performance and robustness of the model?	During, After	Both	5	Projection

Table 2: Supported analytical tasks: questions that our interface helps to answer.

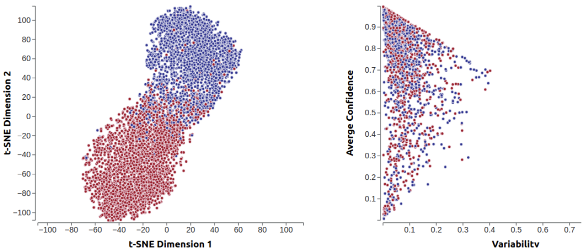
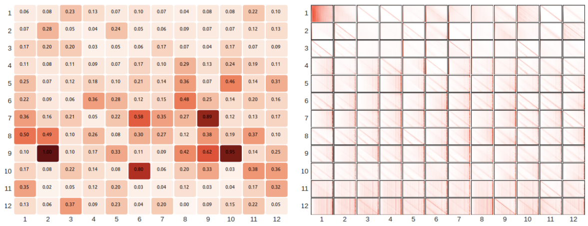


Figure 2: Interactive scatterplots based on the data examples' training dynamics (left), and the t-SNE projections of hidden states (right)



(a) Head Importance (b) Attention Pattern

Figure 3: The two visualization techniques in the Attention Head View.

all examples of the dataset in a single scrollable list, where each entry displays the input text of a data example along with its ground truth label. When the user filters the dataset in the Projection View, the Data Table is also filtered simultaneously.

Attention Head View: In order to visualize the importance of the model's attention heads (**UR-3**), as well as the patterns in the attention weight matrices (**UR-2**), we design the *Attention Head View* (Figure 1-(C)), where each block in the $l \times h$ matrix (l layers and h heads) represents a single attention head at the respective index for layer and head. In this view, we provide two separate visualization techniques: namely (1) Head Importance and (2) Attention Pattern, that can be switched using a toggle button. The Head Importance technique visualizes the normalized task-specific head importance score [1] through the background color saturation and displayed value of the corresponding matrix block (See Figure 3a). On the other hand, the Attention Pattern technique uses heatmaps to visualize the magnitude of the associated self-attention weight matrices (See Figure 3b). We also provide a toggle button for the user to visualize the importance score and attention patterns on two scales, where the **aggregate**-scale visualizes the score and patterns averaged over the entire dataset, while the **instance**-scale visualizes the score and patterns for a selected data example. Lastly, we also offer an interactive technique for the user to dynamically prune attention heads and visualize the effects on a selected example. By hovering over each attention head block in the view, the user can click on the close icon to prune the respective attention head from the model.

Instance Investigation View: After the user selects a data example from the Projection View or Data Table, the *Instance Investigation View* (Figure 1-(D)) renders the corresponding input text sequence along with the model predictions and labels to allow the user to perform detailed analysis on the data example. In this view, each token of the input sequence is displayed in a separate text block, where the background color saturation of each text block encodes the relative saliency or importance of the token based on the interpretation methods. Our interface provides two analysis techniques: (1) By selecting a head in the Attention Head View (Figure 3), the user can click on the text block of any input token to visualize the self-attention dis-

[1]Details are in A.1 of the Appendix

tribution of the selected token over the input text sequence **(UR-3)**. (2) Similarly, the user can visualize the input saliency map with respect to a model output, by clicking the corresponding output token **(UR-4)**. Since our framework allows the user to plug in different interpretation techniques based on their preference, details regarding the meaningfulness of such techniques are out of the scope of this paper. Our interface provides the implementation of two input interpretation methods[2] : Layer-wise relevance propagation (Bach et al., 2015), and input gradient (Simonyan et al., 2013).

2.4 Implementation

Data Processing For each model checkpoint, data pertaining to dataset-level visualizations including hidden state projections, prediction confidence/variability, head importance score, and other attributes (e.g. loss, prediction) are first processed and saved in a back-end directory. The only added computational overhead to the user's training process is the dimension reduction algorithm for projecting hidden state representation, as other visualized values can all be extracted from the forward (e.g. confidence, variability, loss) and backward pass (e.g. head importance, input saliency) of model training.

Back-end Our back-end Python server provides built-in support for the PyTorch HuggingFace library (Wolf et al., 2020), including methods for extracting attention values, head pruning, computing importance scores, and interpreting the model predictions. In order to avoid saving instance-level data (e.g., attention weights, input heatmap, etc.) for all examples in the dataset, the server dynamically computes these values for a selected data example by performing a single forward and backward pass on the model. This requires the server to keep track of the model's current state, as well as a dataloader for indexing the selected data example.

Front-end Our front-end implementation keeps track of the current visual state of the interface including the selections, filters, and checkpoint. The interface can be accessed through any web browser, where data is retrieved from the back-end server via the RESTful API. The interactive visual components of the interface are implemented using the D3.js framework (Bostock et al., 2011), and other UI components (e.g. buttons, sliders) are implemented with popular front-end libraries (e.g. jQuery, Bootstrap).

[2]Details are in A.2 of the Appendix

3 Iterative Design

3.1 Focus Group Study

In order to collect suggestions and initial feedback on T^3-Vis, we conducted a focus group study with 20 NLP researchers that work regularly with pre-trained Transformer models. In this study, we first presented the design of the interface, then gave a demo showing its usage on an example. Throughout the process, we gathered responses from the participants via open discussions.

Most positive feedback focused on the effectiveness of our techniques for visualizing self-attention especially on longer documents (in contrast to showing links between tokens (Vig, 2019)). There were also comments on the usefulness of the input saliency map in providing insightful clues on the model's decision process.

Some participants also suggested that the interface would be more useful for classification problems with well-defined evaluation metrics since data examples tended to be better clustered in the Projection View so that they could be easily filtered for error analysis. The need of optimizing the front-end to support the visualization of large-scale datasets was also mentioned.

On the negative side, some participants were concerned by the information loss intrinsic in the dimension reduction methods, whose possible negative effects on the user analysis tasks definitely requires further study. Encouragingly, at the end, a few participants expressed interest in applying and evaluating T^3-Vis on their datasets and NLP tasks.

3.2 Case Studies

This section describes two case studies of how T^3-Vis facilitates the understanding and exploration of the fine-tuning process through applications with real-world corpora. These studies provide initial evidence on the effectiveness of different visualization components, and serve as examples for how our framework can be used.

3.2.1 Pattern Exploration for an Extractive Summarizer

NLP researchers in our group, who work on summarization, applied T^3-Vis to the extractive summarization task, which aims to compress a document

Figure 4: The self-attention distribution of token "photo" in the *Instance Analysis View*.

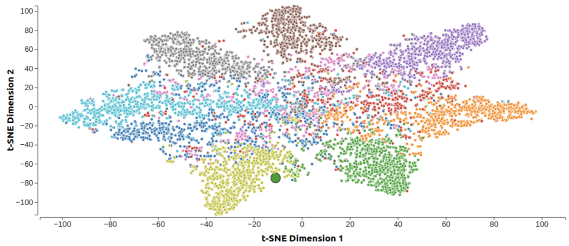


Figure 5: A misclassified example within a cluster of well-classified example.

by selecting its most informative sentences. BERTSum, which is fine-tuned from a BERT model (Liu and Lapata, 2019), is one of the top-performing models for extractive summarization, but why and how it works remains a mystery. With our interface, the researchers explored patterns captured by the model that played important roles in model predictions. They performed an analysis on the CNN/Daily Mail dataset (Hermann et al., 2015), which is arguably the most popular benchmark for summarization tasks.

The first step was to find the important heads among all the heads across all the layers. From the Head Importance View (Figure 1-(C)), the researchers selected the attention heads with high head importance scores, so that the corresponding attention distribution was available to interact with. Then they selected some tokens in the Attention View to see which tokens they mostly attended to, and repeated this process for multiple other data examples, in order to explore whether there was a general pattern across different data examples.

While examining attention heads based on their importance in descending order, the researchers observed that tokens tended to have high attention on other tokens of the same word on the important attention heads. For example, the token "photo" attributed almost all of its attention score to other instances of the token "photo" in the source document (Figure 4). They further found two more patterns in other important heads, in which the tokens tended to have more attention on the tokens within the same sentence, as well as the adjacent tokens. These behaviours were consistent across different pretrained models, such as RoBERTa (Liu et al., 2019).

These findings provided useful insights to assist the researchers in designing more efficient and accurate summarization models in the future, and served as a motivation for the researchers to perform similar analysis for other NLP tasks.

3.2.2 Error Analysis for Topic Classification

Other researchers in our group explored the interface for error analysis to identify possible improvements of a BERT-based model for topic classification. The Yahoo Answers dataset (Zhang et al., 2015) was used, which contains 10 topic classes.

Researchers first used the Projection View (Figure 1-(A)) to find misclassified data examples as applying filters to select label and prediction classes. For a selected topic class in the t-SNE projection of the model's hidden states, they found out that the misclassified data points far away from clusters of correctly predicted examples were often mislabeled during annotation. Therefore, misclassfied data points within such clusters were of greater interest to them since such points tends to indicate model failuresrather than mistakes in annotation (Figure 5). Furthermore, data points in the area with low variability and low confidence on the Data Map plot were also selected for investigation since they are interpreted as consistently misclassified across epochs. After selecting the examples, the researchers inspected each instance by using the Instance Investigation View (Figure 1-(D)) with the Input Gradient method to visualize the input saliency map for the prediction of each class.

From this analysis, they discovered two scenarios that led to misclassification. First, the model focused on unimportant and possibly misleading details that are not representative of the document's overall topic. For instance, a document about *Business & Finance* was classified into the *Sport* category because the model attended to "hockey player", "football player", and "baseball player", which were listed as job titles while discussing available jobs in Michigan. Second, the model failed in cases where background knowledge is required. For example, a document under the *Entertainment & Music* category mentioned names of two actors which were key clues for the topic, but the model only attended to other words, and made a wrong prediction.

These findings helped researchers to gain insights for future model design where additional information such as discourse structure (which can better reveal importance) and encyclopedic knowledge could be injected into the model's architecture to improve the task performance.

4 Conclusion

In this paper, we presented T^3-Vis, a visual analytic framework designed to help researchers better understand training and fine-tuning processes of Transformer-based models. Our visual interface provides faceted visualization of a Transformer model and allows exploring data across multiple granularities, while enabling users to dynamically interact with the model. Additionally, our implementation and design allows flexible customization to support a diverse range of tasks and workflows. Our focus group and case studies demonstrated the effectiveness of our interface by assisting the researchers in interpreting the models' behaviour and identifying potential directions to improve task performances.

For future work, we will continue to improve our framework through the iterative process of exploring further usage scenarios and collecting feedback from users. We will extend our framework to provide a more advanced visualization for custom Transformers. For example, we may want to support the visualization of models with more complex connections (e.g. parallel attention layers) or an advanced attention mechanism (e.g. sparse attention).

References

Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A Gers. 2020. Visbert: Hidden-state visualizations for transformers. In *Companion Proceedings of the Web Conference 2020*, pages 207–211.

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).

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. 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 4143–4152, Hong Kong, China. Association for Computational Linguistics.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2020. Investigating learning dynamics of BERT fine-tuning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 87–92, Suzhou, China. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28, pages 1693–1701. Curran Associates, Inc.

Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2018. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 25(8):2674–2693.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

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

David Johnson, Giuseppe Carenini, and Gabriel Murray. 2020. Njm-vis: interpreting neural joint models in nlp. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 286–296.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Polo Chau. 2017. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. 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 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Vasudev Lal, Arden Ma, Estelle Aflalo, Phillip Howard, Ana Simoes, Daniel Korat, Oren Pereg, Gadi Singer, and Moshe Wasserblat. 2021. InterpreT: An interactive visualization tool for interpreting transformers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 135–142, Online. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*.

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

Dongyu Liu, Weiwei Cui, Kai Jin, Yuxiao Guo, and Huamin Qu. 2018. Deeptracker: Visualizing the training process of convolutional neural networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(1):1–25.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. 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 3730–3740, Hong Kong, China. Association for Computational Linguistics.

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 preprint arXiv:1907.11692*.

L. McInnes, J. Healy, and J. Melville. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online. Association for Computational Linguistics.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

Tamara Munzner. 2009. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6):921–928.

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.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. 2016. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 107–118, Online. Association for Computational Linguistics.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

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.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. AllenNLP interpret: A framework for explaining predictions of NLP models. 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): System Demonstrations*, pages 7–12, Hong Kong, China. Association for Computational Linguistics.

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

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shouxing Xiang, Xi Ye, Jiazhi Xia, Jing Wu, Yang Chen, and Shixia Liu. 2019. Interactive correction of mislabeled training data. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 57–68. IEEE.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

A Appendix

A.1 Head Importance Score

Although the multi-head self attention mechanism in Transformers allows the model to learn multiple types of relationships between input representations across a single hidden layer, the importance of the individual attention heads can vary depending on the downstream tasks. Following previous work, we adapt the Taylor expansion method (Molchanov et al., 2019) to estimate the error induced from removing a group of parameters from the model. In our implementation, we use the first-order expansion to avoid the overhead from computing the Hessian, where the gradient with respect to validation loss is summed over all parameters of an attention head to estimate its importance.

A.2 Input Interpretation

Input Gradients The input gradient method (Simonyan et al., 2013) computes the gradient with respect to each token. During inference, the class-score derivative can be computed through back-propagation. The saliency of the token x_i for class c of output y could therefore be estimated using the first-order Taylor expansion $\frac{\partial y_c}{\partial x_i} x_i$.

Layer-wise Relevance Propagation Layer-wise Relevance Propagation (LRP) (Bach et al., 2015) was originally proposed to visualize the contributions of single pixels to predictions for an image classifier. By recursively computing relevance from the output layer to the input layer, LRP is demonstrated to be useful in unravelling the inference process of neural networks and has been adopted in recent work to analyze Transformer models (Voita et al., 2019). The intuition behind LRP is that, each neuron of the network is contributed by neurons in the previous layer, and the total amount of contributions for each layer should be a constant during back-propagating, which is called the *conservation principle*. LRP offers flexibility to design propagation rules to explain various deep neural networks, one example propagation rule is shown as follows (Montavon et al., 2018),

$$R_i = \Sigma_j \frac{a_i w_{ij}}{\Sigma_i a_i w_{ij}} R_j \tag{1}$$

where R_i and R_j are relevance scores of two neurons in consecutive layers, a_i is the respective activation for neuron i, and w_{ij} is the weight between neuron i and j.