

**T³-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, hyeuju, 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 effectively model 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 (Kovalева 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 T³-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-
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.

Table 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 T³-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 T³-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...
### Table 1: Comparison with other visual frameworks from recent work.

<table>
<thead>
<tr>
<th>Frameworks</th>
<th>Components</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dataset</td>
<td>Embeddings</td>
</tr>
<tr>
<td>BertViz (Vig, 2019)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>AllenNLP Interpret (Wallace et al., 2019)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>exBERT (Hoover et al., 2020)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LIT (Tenney et al., 2020)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>InterpreT (Lal et al., 2021)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>T³-Vis</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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

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)

![Interactive scatterplots](image1)

![t-SNE projections](image2)

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

![Head Importance](image3a)

![Attention Pattern](image3b)

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 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-

---

1Details 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).

### 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
by selecting its most informative sentences. BERT-Sum, 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, misclassified data points within such clusters were of greater interest to them since such points tends to indicate model failures rather 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 T³-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


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


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 backpropagation. The saliency of the token $x_i$ for class $c$ of output $y$ could therefore be estimated using the first-order Taylor expansion $\frac{∂y}{∂x_i} x_i$.

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),

$$R_i = \frac{a_i w_{ij}}{\sum_i a_i w_{ij}} R_j$$

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 $j$. 230