Online Research @ Cardiff

information services
gwasanaethau gwybodaeth

A Unified Understanding of Deep NLP Models for Text Classification

Zhen Li, Xiting Wang, Weikai Yang, Jing Wu, Zhengyan Zhang,
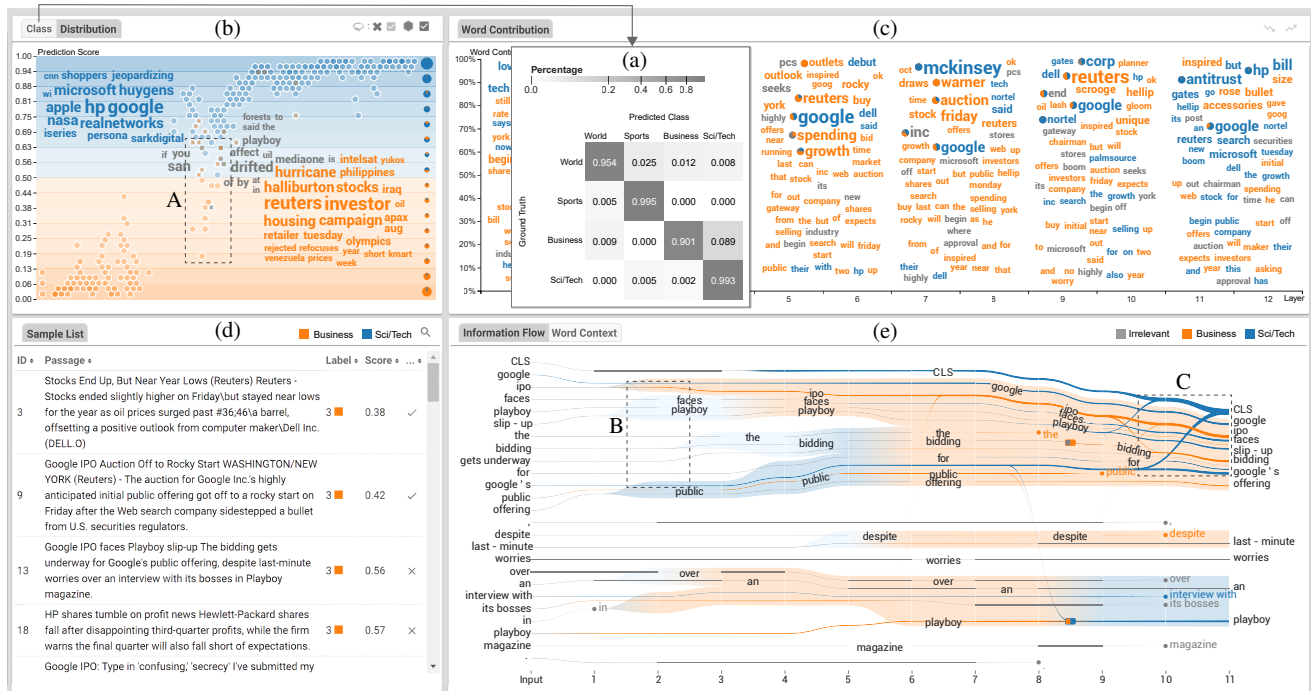Zhiyuan Liu, Maosong Sun, Hui Zhang, Shixia Liu

Fig. 1: DeepNLPVis for analyzing the BERT model on news classification: (a) class view for showing the overall model performance; (b) distribution view for identifying samples and words of interest; (c) word contribution of selected samples; (d) sample list; (e) information flow for analyzing a sample by its intra- and inter-word information;

Abstract—The rapid development of deep natural language processing (NLP) models for text classification has led to an urgent need for a unified understanding of these models proposed individually. Existing methods cannot meet the need for understanding different models in one framework due to the lack of a unified measure for explaining both low-level (e.g., words) and high-level (e.g., phrases) features. We have developed a visual analysis tool, DeepNLPVis, to enable a unified understanding of NLP models for text classification. The key idea is a mutual information-based measure, which provides quantitative explanations on how each layer of a model maintains the information of input words in a sample. We model the intra- and inter-word information at each layer measuring the importance of a word to the final prediction as well as the relationships between words, such as the formation of phrases. A multi-level visualization, which consists of a corpus-level, a sample-level, and a word-level visualization, supports the analysis from the overall training set to individual samples. Two case studies on classification tasks and comparison between models demonstrate that DeepNLPVis can help users effectively identify potential problems caused by samples and model architectures and then make informed improvements.

Index Terms—Explainable AI, visual debugging, visual analytics, deep NLP model, information-based interpretation

◆

1 INTRODUCTION

Text classification is a fundamental task in natural language processing (NLP) and has been under rapid development to assist our everyday communication [1]. In recent years, different deep NLP models from CNN-based [2], LSTM-based [3], and Transformer/attention-based [4] have been consecutively proposed to improve the performance of text classification tasks. However,

along with the improved performance is the increasing complexity of the model architecture, which poses difficulties for model developers not only in training the model, but also in debugging when the performance is not as expected.

For example, the recent BERT model [5] contains hundreds of millions of parameters. Training such a model from scratch requires massive data and computing resources that are unaffordable to most NLP developers. A training schema with pre-training and fine-

tuning is thus getting popular. Starting from a pre-trained model (e.g., a pre-trained BERT), model developers fine-tune the model to the end classification task with specific input-output designs. They usually follow two approaches to improve the model performance. One is to augment the training data and improve the data quality with the model architecture unchanged. In machine learning, it has been proposed that "80% data + 20% model = better machine learning" [6], which demonstrates the importance of data quality. Label errors, missing samples, and sample bias are all factors that affect classification performance [7]. The other approach is to make slight changes to the model architecture. For example, inserting "adapter layers" between specific layers of the model has been shown effective in improving model performance [8].

Effective improvement from either of the above approaches will require the model developers to understand the model's working mechanism, and in turn to identify the deficiencies for informed augmentation of the data and/or adaptation of the model architecture. Several visualization tools have been developed to facilitate the understanding of a specific deep NLP model, such as RNNVis [9] and Attention Flows [10] for RNN-based and attention-based NLP models, respectively. Our work follows this direction to assist in understanding deep NLP models for text classification. However, we argue that a tool for a specific model may restrict the model developer's choice of the most suitable model for a specific task. Currently, there is a trend to revisit deep learning models [11], and it has been found some simple models can actually achieve competing performance as more complex models. Given the high computing demand of pre-trained models, it would be interesting to know whether simpler models, such as LSTM or CNN-based models, are potential alternatives at least for some scenarios in classification tasks. For example, improving the accuracy of these simpler models will make them deployable on portable devices with limited computing resources, such as mobile phones. It thus naturally arouses the interest to revisit different classification models in NLP. However, due to the diverse model architectures, this is non-trivial. A fundamental requirement is to unify the understanding of these models' working mechanisms. With this in mind, although our work follows the direction of developing visualization tools for deep NLP models, it has a unique focus on a unified understanding across different classification models. It aims to help model developers better understand the strengths and weaknesses of different NLP models and make informed improvements.

We thus develop DeepNLPVis, an interactive visual analysis tool to help model developers gain a unified understanding of different NLP models for text classification, quickly identify problems, and make informed improvements. We propose to improve the mutual information-based measure in [12] to explain the information learned by intermediate layers, including both intra-word information (e.g., word contribution to classes) and inter-word information (e.g., relationships between words). With this measure, DeepNLPVis adopts a coordinated multiple-level visualization connecting the analysis from the overall training corpus to individual samples and words (Fig. 1). At the corpus-level, a class view shows the overall model performance on all classes, from which the user can select two classes to explore the corpus against predictions on the two classes in the distribution view. The sample-level visualization supports analyzing a sample in terms of both intra-word and inter-word information. And the word-level visualization supports examining words in terms of word contribution and word meaning. The interactive analysis

enabled by the coordinated visualization helps users explore model deficiencies and identify the root cause of low performance. The demo is available at https://bit.ly/2QUF4Pb.

In summary, the main contributions of this work are:

- A mutual-information-based visual analysis tool for efficient identification and diagnosis of problems in deep NLP models for text classification.
- An information-based sample interpretation method for simultaneously understanding the intra-word and inter-word information in a unified way.
- A three-level visualization consisting of a corpus-level visualization for quickly identifying samples and words of interest, and a sample-level and a word-level visualization to disclose the intra-word and inter-word information and their changes across layers.

2 RELATED WORK

2.1 Machine Learning for Understanding NLP Models

Existing machine learning methods for understanding deep NLP models can be categorized into three classes: built-in interpretability methods, post-hoc model-specific methods, and post-hoc model-agnostic methods [13], [14]. The first category attempts to design self-explanatory models. The second category analyzes some specific architectures in the model, such as the hidden states or attention heads. The third category interprets the model by considering model inputs, intermediate layers, and outputs. These methods are model-agnostic as they do not make assumptions about the specific model architecture. Our work is relevant to the third category. Here we briefly review the works along this line.

Many post-hoc model-agnostic methods utilize an easy-to-interpret model, such as decision trees, to approximate the original model and explain its behavior [15]. Local Interpretable Model-agnostic Explanations [16] is a representative work, which trains a sparse linear model and uses it to explain the black-box model locally. Recently, Guan *et al.* [12] proposed a measure to quantify the information stored in each word. This measure provides quantitative explanations on the contribution of a word to the final prediction layer by layer. Compared with existing methods, the explanations provided by this method are consistent across different NLP models. Our method extends this unified measure to help understand which class the word contributes to and the relationships between words (e.g., phrases). We also leverage interactive visualization to visually explain the aforementioned word-related information across layers. This helps users quickly identify unusual words/phrases that lead to low performance.

2.2 Visualization for Understanding NLP Models

Existing visualization methods for visually understanding machine learning models can be categorized into two classes: domain irrelevant [17], [18] and domain specific [19], [20], [21], [22], [23], [24]. Our work is in the second category with a focus in the NLP domain, which enables an in-depth understanding of the working mechanism of the model training process. Thus, we briefly review the works along this line.

Earlier efforts focus on utilizing simple and static diagrams, such as heatmaps [25], [26], to demonstrate which input words play important roles in model prediction. Later efforts employ interactive visualization to analyze the intermediate layers of the models, such as hidden states and attention mechanisms [7], [27].
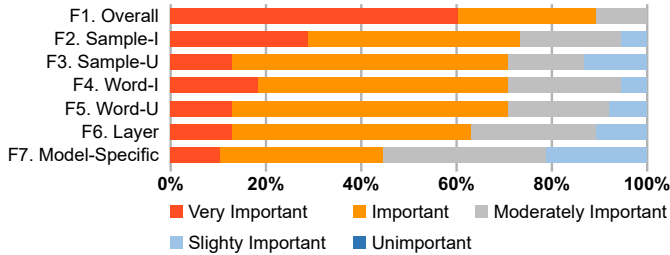
Fig. 2: Importance of different functions. Sample-I (or Word-I) refers to the identification of samples (or words) of interest, and Sample-U (or Word-U) denotes understanding a sample (or a word).

Ming *et al.* [9] introduced a visual analysis method for interpreting hidden states of RNNs based on their expected response to the input. Strobelt *et al.* [3] developed LSTMVis to explain the hidden state changes and identify similar examples. RNNbow [28] visualizes the gradient flow in the training process of RNNs to investigate the learning behavior of a model. Seq2Seq-Vis [29] visually analyzes the five black-box stages of the machine translation process. The attention mechanism is widely used in visual model explanation due to its high interpretability. Accordingly, RetainVis [30] leverages the attention mechanism to explain how predictions are made and supports the modification of the input or the model to conduct what-if analysis. Tenney *et al.* [31] developed a language interpretability tool, which utilizes the attention mechanism and salience maps to provide a comprehensive understanding of model behavior. More recently, Derose *et al.* [10] developed Attention Flows to trace and compare the attention heads in a BERT model. The system supports both single model analysis and comparison between pre-trained and fine-tuned models.

The aforementioned works have achieved considerable success in understanding deep NLP models. However, they mainly focus on a specific type of NLP models (e.g., RNNs or attention-based models). In comparison, we have developed a unified method for understanding different types of NLP models for text classification. Our method enables a deeper understanding of the capabilities of different classification models by revealing the word contribution and word relationships at different layers. A multi-level visualization is also carefully designed to help users quickly identify the samples and words of interest in the context of the overall data distribution and model prediction.

3 REQUIREMENT ANALYSIS

3.1 Survey on Practices of Building Deep NLP Models

We conducted a questionnaire to better understand the current practices of NLP model developers and the key functions they need for efficiently developing deep models for text classification.

Designing the questionnaire. The questionnaire was designed and iteratively refined with five model developers, who have varying experience (from one year to eight years) in NLP. We conducted a 45-60 minute semi-structured interview with each of them. In the interview, we first asked the model developers to introduce his/her current practices and difficulties. Then, we explored the functions that s/he needed for better understanding and debugging a model. Their feedback was summarized to create the questionnaire.

Conducting the questionnaire. The questionnaire was distributed to 1) students in three NLP groups of a top university and 2) NLP model developers in a major technology company. Out of the 46 returned questionnaires, eight (17.4%) were discarded due to incomplete responses. Among the participants, 21.1% had less than 1-year experience in NLP, 44.7% had 1-3-year experience, 28.9% had 3-5-year experience, and 5.3% had 5-10-year experience. The NLP models they used include Pre-trained Model such as BERT (81.6%), Transformer (73.7%), RNN (52.6%), and CNN (36.8%).

Current practices and difficulties. A majority of the participants understood and debugged NLP models by investigating the training loss (89.5%), logging intermediate results (84.2%), and observing the changes in model accuracy (76.3%). Only 34.2% of them leveraged tools that were specifically designed for model understanding, e.g., Tensorboard or BertViz. They commented that it was difficult to use current tools for identifying problems in the training data, e.g., incorrect labels and uneven data distribution (78.9%). It was also difficult to understand why a model could not correctly predict the labels for certain samples (50.0%) and why the model incorrectly understood certain words (28.9%). In addition to the difficulty in understanding and debugging a single model, the participants also expressed the needs to effectively compare different models. On average, they experimented with five models in the most recent project. Many participants have shown interest in a tool that can help investigate NLP models in a unified way and compare the models effectively. We asked them to rate their interest according to a 1-5 Likert scale, and 84.2% of participants returned a rating of 4 (interested) or 5 (extremely interested).

Key functions needed. We then summarized the key functions that help effectively develop NLP models and asked the participants to rate how important each function is based on a 1-5 Likert scale (1: unimportant, 5: very important). Fig. 2 shows the rating distribution of different functions. More than 80% of participants considered examining the overall performance and model behavior on the training data (F1) as (very) important. The participants were also eager to understand how samples and words contribute to the model performance. Specifically, more than 70% of participants agreed that it was (very) important to identify key samples (F2) and words (F4) for understanding and debugging a model. Over 70% of participants also expressed the need to deeply understand how each sample (F3) or word (F5) affects the model prediction. Among the information about model architectures (e.g., layers, neurons, or recurrent cells), analyzing how a model behaves across layers (F6) attracted the most attention. 63.2% of participants considered understanding layer-wise evolution to be (very) important. In comparison, investigating neurons or model-specific architecture (F7) (e.g., specific activation function, recurrent cell, convolution layer) was not frequently cited by the participants.

3.2 Design Requirements

We further conducted interviews with five experts (E_1-E_5) selected from the 38 questionnaire participants. The experts are selected to ensure that they have different levels of expertise and work on various NLP models. In particular, E_1 has 1-year experience in NLP, E_2 and E_3 have 3-year experience, E_4 has 5-year experience, and E_5 has 10-year experience. All experts are familiar with BERT and Transformer. E_4 and E_5 also have experience in training LSTM and CNN. Based on the questionnaire survey and interviews, we distilled three-level requirements: corpus, sample, and word.

The **corpus**-level requirement aims to help users obtain a quick overview of the model behavior (F1).

R1. Exploring how model prediction scores distributed over the dataset. According to the questionnaire, most model developers

considered understanding the overall model behavior and performance an essential step for analyzing NLP models. As the final output of the model, prediction scores are a major signal of its behavior [17]. To obtain an overview of the model performance, E_1 to E_5 agreed that it was essential to show the distribution of prediction scores over the dataset. For example, E_1 said, "I would like to see whether the model makes mistakes on a particular set of similar samples or on diversified samples."

The **sample**-level requirements reflect users' need to identify the samples of interest (F2) and analyze them (F3) in a unified way.

R2. Identifying samples that are essential for understanding and debugging the model. A common need expressed by the questionnaire participants is to find samples that are useful for model understanding and debugging (F2). Such samples of interest can be characterized from multiple aspects. For example, checking the samples for which the model makes a wrong prediction can help quickly debug the model (E_1–E_5). Investigating the samples that are close to the decision boundary can help increase model robustness [32]. Discovering the representative samples that are similar to many samples may shed light on why the model achieves good or bad performance. Moreover, the experts required a way to identify the samples of interest from the word perspective, e.g., finding samples with a word that the model fails to correctly understand.

R3. Revealing how NLP models learn low- and high-level features of a sample across layers in a unified way. After identifying the samples of interest, the experts needed to understand how the model processes the samples across layers and why it makes a certain prediction for the samples (F3, F6). This allows them to figure out the underlying working mechanism of the model, which is important for model understanding and debugging. Most existing tools help reveal important words (low-level features) in a sample. In addition to the low-level features, the experts are also interested in the high-level features learned, e.g., whether the model can correctly understand sentence structures (E_1-E_4). For example, E_3 said, "It is interesting to see whether a model judges the sentiment of a long compound sentence by considering the word relationships (e.g., phrases) or simply by counting positive and negative words." In addition, to facilitate model comparison, the experts noted that it was necessary to provide consistent results for different models.

The **word**-level requirements focus on identifying the words of interest (F4) and analyzing these words (F5) in a unified manner.

R4. Identifying the words that are important for understanding and debugging the model. As an NLP dataset typically contains tens of thousands of words, it is very difficult for a user to check each word manually and decide which word is important for model understanding and debugging. Accordingly, the experts required a method to help them quickly identify the words of interest (F4), e.g., ambiguous words or words that contribute the most to model prediction (E_1-E_5).

R5. Revealing how the model understands the meaning of a word by considering the context. After identifying the important words, the experts wanted to further investigate how the words affect the model prediction (F5), so that they can judge whether the model understands the meaning of the words in a correct way (E_2-E_5). Instead of considering each word independently, most NLP models consider a word by simultaneously modeling its context (the related words in the same sample). To better understand how a word impacts a model, the experts are interested in knowing more about the contextual information of the word. For example, E_4 said, "I would like to see whether the model can correctly distinguish different meanings of *like* based on other words in the sentences."

4 DEEPNLPVIS

4.1 Overview

The large amount of NLP-model-related data, such as samples, words, and information-based measure data, makes it difficult for users to identify the most informative information for model understanding. To tackle this issue, we have developed a multi-level visualization and combined it with a unified information-based measure for understanding a deep NLP model from the perspectives of the corpus, samples, and words.

As shown in Fig. 3, the information-based measure, including intra-word information and inter-word information, is first extracted. Then based on the extracted measure, the three-level visualizations are seamlessly coordinated together and support an iterative **analysis workflow** for a unified understanding of model training behaviors. The **corpus-level visualization** visually illustrates 1) the overall model performance of all classes in a confusion matrix (class view); 2) the training samples of the two selected classes from the class view and the corresponding important keywords in a hexagonal heatmap (distribution view). This visualization enables quick identification of the classes, samples, and words of interest (*R1, R2, R4*). Selected samples are displayed in the **sample-level visualization**, and their associated words are displayed in
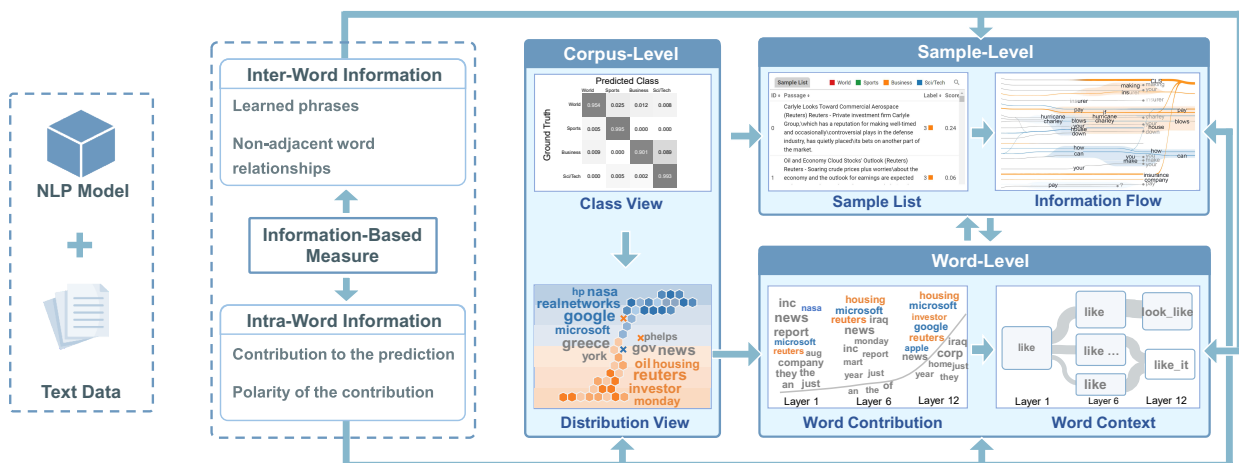

Fig. 3: The analysis workflow supported by three coordinated visualizations at different levels.
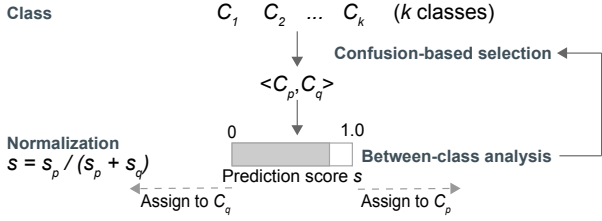
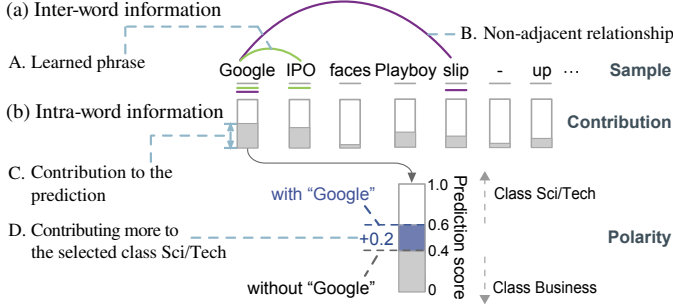Fig. 4: The analysis of multi-class classification is achieved by an iterative between-class analysis.

Fig. 5: The visual explanation of intra- and inter-word information.

the **word-level visualization** for closer examination. For example, users can identify the samples with wrong predictions and words with higher contribution values by using the sample list (*R2*) and word contribution view (*R4*). Then for each sample, users can explore the relationships between words and how they form and change across layers (e.g., formation of phrases) in the information flow (*R3*). They can also analyze the word of interest in the context of relevant words with the word context view (*R5*).

4.2 Information-Based Interpretation

We propose an information-based interpretation method for identifying the key information used by an NLP model for prediction. Without loss of generality, we introduce how to interpret a multi-class classifier. When analyzing the performance of a classification model, the experts usually start with an overall class-level analysis, and then perform the between-class analysis. Inspired by this observation, we simplify the analysis of multi-class classification into an iterative between-class analysis, as illustrated in Fig. 4. In particular, the users first select two classes (e.g., C_p and C_q), based on the confusion matrix for k classes in the class view. We then remove the impact of the unselected classes by computing a normalized prediction score $s = s_p/(s_p + s_q)$, where s_p and s_q are the original prediction scores for C_p and C_q. Normalized prediction score s helps understand whether the model is confident or confused with respect to C_p and C_q: a large s above 0.5 indicates a confident prediction of C_p, a small s below 0.5 indicate a confident prediction of C_q, and a value of s around 0.5 indicates that the model confuses the two classes in terms of the sample. The goal of our interpretation method is to understand what information the model leverages for deciding the prediction s.

The information is divided into two categories: **intra-word** and **inter-word**. Intra-word information helps analyze how each single word contributes to the sample prediction, and inter-word information aims to capture the relationships between words (Fig. 5). Recently, Guan *et al*. [12] proposed a unified information-based measure to estimate the **contribution** of the word to the prediction

(Fig. 5C), which partially addresses the problem of analyzing intra-word information. Given a sample $\mathbf{X} = (w_1, w_2..., w_n)$ and its prediction score s, the contribution of the i-th word w_i at the ℓ-th layer is measured by the amount of information that is passed from layer l to the final prediction. It is computed as mutual information: $\mathrm{MI}(h^{(\ell)}(w_i); s)$, where $h^{(\ell)}(w_i)$ is the latent representation of the i-th word of the sample at the ℓ-th layer. For models where there exist no 1-to-1 association between the latent representations and input words (e.g., CNN [2]), $h^{(\ell)}(w_i)$ is set to the concatenation of all hidden representations affected by w_i. The mutual information can be computed efficiently by using perturbation-based approximation [12]. The basic idea is to perturb $h^{(\ell)}(w_i)$ by adding a Gaussian noise $\boldsymbol{\varepsilon}_i$ and measure the magnitude of change in the prediction score s:

$$\Delta s_i = \mathbb{E}_{\boldsymbol{\varepsilon}_i \sim N(\mathbf{0}, \sigma_i^* \mathbf{I})} \frac{\phi(h^{(\ell)}(w_i) + \boldsymbol{\varepsilon}_i) - s}{\sigma_s}. \quad (1)$$

Here, $\phi(\cdot)$ is the prediction function represented by the layers after l and satisfies $s = \phi(h^{(\ell)}(w_i))$. The perturbation $\boldsymbol{\varepsilon}_i$ is a noise sampled from the Gaussian distribution $N(\mathbf{0}, \sigma_i^* \mathbf{I})$, where σ_i^* is the optimal standard deviation computed by using the maximum likelihood estimation loss [12], $\mathbf{I}$ is an identity matrix, and σ_s is the standard deviation of s. A larger magnitude change $|\Delta s_i|$ indicates a larger contribution of the word (Fig. 5C).

Although this measure can compute the absolute value of word contribution, it fails to provide information for understanding the polarity of the contribution. Moreover, it fails to capture the relationships between words (inter-word information). Here, the **polarity** indicates towards which class the word or word combination (e.g., phrase) contributes to the prediction, given two classes selected from the class view (Fig. 5D). The polarity is important for identifying the root cause for the confusion between the two classes. For example, the sample in Fig. 5 is misclassified to "sci/tech" because the model considers "google" as a word related to "sci/tech" rather than "business", even though "google" is mentioned together with "ipo." In addition to polarity, another important type of information is inter-word information, which is useful for detecting high-level features learned by the model, such as phrases (Fig. 5A) and non-adjacent word relationships (Fig. 5B). Next, we introduce how we extend the information-based measure to learn the polarity of the contribution and inter-word information.

The polarity of the word is measured by the sign of change in the prediction score with the existence of that word (Fig. 5D). Specifically, $\Delta s_i > 0$ means that removing the i-th word increases the prediction score s, which indicates the existence of w_i contributes to assigning a sample to class C_q. Thus, w_i is a C_q-relevant word. $\Delta s_i < 0$ means that w_i is a C_p-relevant word. To increase the robustness of the method, we further use a margin $\xi > 0$ to extract the most relevant words. $\Delta s_i > \xi$, $\Delta s_i < -\xi$, or $-\xi \leq \Delta s_i \leq \xi$ mean that word w_i is a C_q-relevant word, C_p-relevant word, and class-irrelevant word. A good value of ξ should well differentiate class-relevant and -irrelevant words. For example, words like "ipo" and "spending" should be considered relevant with "business," words like "search" and "gates" should be considered relevant with "sci/tech," and words like "time" and "can" should be class-irrelevant. We experiment with seven datasets and find that the best value for ξ usually increases with increasing text length (see supplement for detailed results). We suspect that this is because for longer text, the information is scattered across more words, resulting in a larger variance of mutual information and thus

the requirement for a larger margin ξ. Since the experts usually use datasets with short texts, we set the value of ξ in the system to 0.02, which typically works well for short text whose average number of characters is smaller than 300. We also allow users to interactively change the value of ξ for a given dataset.

The inter-word information reveals how an NLP model models the phrases in a sample and learns the relationships between non-adjacent words (Fig. 5(a)). Although different types of neural networks model word relationships in different ways, (e.g., Transformer uses self-attention and CNN leverages convolutional kernels), they all embed the learned relationships into the contextual word embedding [33]. For word w_i, each model identifies its most relevant words (context) and encodes them into the latent representation $h^{(\ell)}(w_i)$. Based on this, we probe into the learned phrases and non-adjacent relationships by analyzing the word information contained in $h^{(\ell)}(w_i)$. Our method for extracting the inter-word information consists of three steps:

Step 1. Computing context vector. The context vectors of words help identify word clusters (phrases) based on the information each word absorbs. For example, at layer 1, each word only contains information about itself (Fig. 6(a)). Later, the words (e.g., "good" and "movie") absorb information from each other, and their context vectors become more similar (layers 4 and 7 in Fig. 6(a)). The context vector $\mathbf{c}_i^{(\ell)}$ of word w_i is created by decomposing the word information contained in $h^{(\ell)}(w_i)$: $c_{ij}^{(\ell)} = MI(w_j, h^{(\ell)}(w_i))$. Here, $c_{ij}^{(\ell)}$ reveals how much information of the j-th input word is used in the contextual word embedding of the i-th word.

Step 2. Extracting the learned phrases. The learned phrases are extracted by clustering the context vectors (Fig. 6(b)). We employ the agglomerative clustering [34] to cluster adjacent context vectors at each layer. To improve stability, the clustering result of layer ℓ is utilized to initialize the clusters at layer $\ell + 1$. As shown in Fig. 6(b), the clustering allows us to find phrases extracted by the model, e.g., "the original" at layer 4 and "a good movie" at layer 7.

Step 3. Extracting non-adjacent word relationships. In addition to showing how adjacent words form phrases, we also consider the interactions between non-adjacent words. Taking the sample in Fig. 6(c) as an example, "n't" interacts with "good" even though they are non-adjacent. This interaction can be quantified by using mutual information: $e_{ij}^{(\ell)} = MI(h^{(\ell)}(w_i); h^{(\ell+1)}(w_j))$, where i and j are the indices of two non-adjacent words at layer ℓ. A large $e_{ij}^{(\ell)}$ reveals that a great deal of information has been passed from the i-th word to the j-th word at layer $\ell + 1$.

4.3 Three-Level Visualization

The three-level visualization enables users to smoothly navigate from the overall performance at the corpus-level to the detailed information at the sample- and word-level.

4.3.1 Corpus-Level Visualization

The corpus-level visualization contains two parts: a **class view** to reveal the overall model performance on all classes and a **distribution view** to explore the prediction distribution over the selected classes from the class view and identify important samples and words for further analysis.

A confusion matrix is employed in the class view (Fig. 1(a)), where each column represents the samples in a predicted class, each row represents the samples in an actual class, and the percentage of samples displayed in each cell depicts the confusion between two classes.

For the distribution view, the Squares visualization [17] is a straightforward solution to visually convey the desired information. Although Squares can well show the model performance in the context of samples, it fails to disclose the similarity relationships between samples. Understanding such relationships are critical for identifying important samples, such as representative samples and outliers in each class, for further investigation.

To tackle this issue, we have developed a hexagonal heatmap (Fig. 1(b)), which integrates the prediction score (y-axis) with the one-dimensional t-SNE projection [35]. Since the users require to examine the similarity relationships between samples, we employ the t-SNE projection. We choose this technique because of its effectiveness in preserving the neighborhoods and clusters of samples [36]. The one-dimensional t-SNE projection projects the sample embedding in the last hidden layer of the model onto the x-axis. The sample color encodes the class of the sample. For example, in Fig. 1, orange and blue encode the "business" and "sci/tech" news, respectively. In the hexagonal heatmap, a blue hexagon ⬢ represents a set of true-positive samples, an orange hexagon ⬢ represents a set of true-negative samples, a blue cross ✖ represents a set of false-negative samples, and an orange cross ✖ represents a set of false-positive samples. The darker the sample color of a hexagon/cross is, the more samples it represents. To provide a comprehensive overview of how samples distribute over prediction scores (y-axis), we bin them into 16 consecutive stripes with an interval of 0.0625. Each stripe contains a pie chart on the right to illustrate the class distribution over the samples.

A group of representative words is placed on the other side in a layout close to the samples. Different colors represent words that are relevant to different classes, and gray represents class-irrelevant words. The size of each word encodes its importance to the model prediction. In NLP, the term frequency–inverse document frequency (TF-IDF) weighting scheme is widely used to measure how important a word is to a *document* [37]. This weighting scheme assigns higher weights to the words whose occurrence is frequent in a small number of documents, but rare in the other documents of the corpus. Inspired by TF-IDF, we compute Importance(w), which measures how important a word is to a *model* in a *corpus* by 1) its term frequency tf(w) in the corpus; and 2) its average contribution over the associated samples, Contribution(w):

$$\text{Importance}(w) = \log(\text{tf}(w) + 1.0) * \text{Contribution}(w), \quad (2)$$

where the first term is the term frequency of word w in the document collection. As the tf values span a large range, a logarithmic operation is applied to normalize the frequency values. All the tf values are also increased by 1 in the log normalization to avoid zero output. The second term measures the contribution of w to the model prediction. Accordingly, a large importance value is attained by a high term frequency and a high contribution score. Such a weighting method tends to filter out common words with little contribution to the final prediction or rare words with low frequency.

To explore the overall model behavior at the corpus level, rich interactions are designed. For example, when a user finds a stripe of interest, s/he can click to enlarge the stripe and examine at a finer scale with more keywords.

4.3.2 Sample-Level Visualization

The sample-level visualization consists of two coordinated components: a sample list and an information flow (Fig. 1).

The sample list (Fig. 1(d)) allows users to examine multiple samples in terms of their text content, class labels, and prediction

(a) Context Vector Calculation

(b) Learned Phrases Extraction

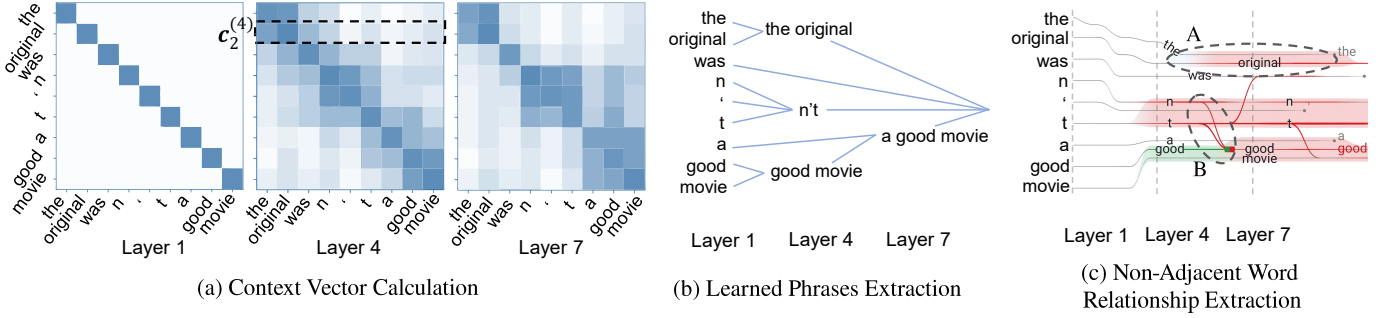(c) Non-Adjacent Word Relationship Extraction

Fig. 6: Illustrating the learned phrases and non-adjacent word relationship of a sample.

scores. Sorting by these attributes makes it possible to identify the samples of interest from multiple perspectives (*R2*). The list is coordinated with other visualizations. For example, the samples in the list will be updated according to the selected sample hexagons or words in the corpus- or word-level visualizations. Users can choose a sample from the list and perform a deeper analysis of the word relationships and their changes by using the information flow.

The information flow (Fig. 1(e)) facilitates users to analyze how NLP models process a sample through layers in a unified way (*R3*). The design is inspired by storyline visualization [38]. We introduce the visual encoding and layout algorithm below.

Visual encoding. As shown in Fig. 6(c), in the flow visualization, each word is represented by a line. The **intra-word** information is encoded by the line width and colors. A wider line indicates that the word contributes more to the final prediction. The color of the line indicates which class the word contributes to (polarity). Additionally, we highlight two types of important changes along a line. First, if a word contributes little to the final prediction after layer ℓ, we will end the line with ●. Second, we highlight the class change on a line by using glyphs 🔴🟢 and 🟢🔴. The **inter-word** information includes the learned phrases and non-adjacent word relationships. For the learned phrases, we use the distances between lines to represent the distances between word context vectors. In this way, words that belong to a phrase are naturally placed close to each other. The clusters are further highlighted by using a background area (Fig. 6A), the color of which is determined by the primary color of the lines. The relationships between non-adjacent words are encoded by the curves that connect different lines (Fig. 6B). The width of a curve from word i at layer ℓ to word j at layer $\ell+1$ is determined by mutual information $e_{ij}^{(\ell)}$ (Sec. 4.2). A thicker curve indicates a larger contribution from word i to word j. To avoid visual clutter, we only display the most important curves. A curve is shown if 1) its weight $e_{ij}^{(\ell)}$ is among the top 5%, or 2) it is useful for illustrating the color/width change of a line. For example, the curve displayed in Fig. 6B is helpful for explaining why the word "good" becomes negative at layer 6, which is caused by "good" absorbing the information of "n't."

Layout algorithm. The layout of the storyline needs to preserve both stability and readability [38], [39]. To achieve this goal, we formulate the layout as a constrained optimization problem.

Denote the y-coordinate of word i at layer ℓ as $y_i^{(\ell)}$. **Stability** prevents the y-coordinate of a line from changing dramatically when its context vector does not change much. This ensures that the wiggles of the line, which easily draw users' attention, are meaningful and worth investigating. We consider two types of stability losses: a continuous loss $(y_i^{(\ell)} - y_i^{(\ell-1)})^2$ and a discrete

loss $\mathbb{I}(y_i^{(\ell)} \neq y_i^{(\ell-1)})$. $\mathbb{I}(\cdot)$ is an indicator function with $\mathbb{I}(\text{true}) = 1$ and $\mathbb{I}(\text{false}) = 0$. While the continuous loss penalizes large changes, the discrete loss limits the number of line wiggles. **Readability** measures how clear and easy it is to understand the relationships between words in a sample. In addition to the phrase relationships between adjacent words, the order of words is also important for many NLP tasks. For example, *isn't he lovely* and *he isn't lovely* have different sentiments. As a result, readability requires that 1) the distances between lines accurately reveal the distances between word context vectors; and 2) the order of words in a sample is preserved. Accordingly, readability is maintained by minimizing the loss $(||y_i^{(\ell)} - y_{i-1}^{(\ell)}|| - D_i^{(\ell)})^2$ and satisfying the constraint $y_i^{(\ell)} \geq y_{i-1}^{(\ell)}$, for $\forall i, l$, where $D_i^{(\ell)} = ||\mathbf{c}_i^{(\ell)} - \mathbf{c}_{i-1}^{(\ell)}||$ is the distance between word context vectors.

Based on the analysis of stability and readability, we formulate the storyline layout as a constrained optimization problem:

$$
\begin{aligned}
\min_{\{y_i^\ell | \forall i, \ell\}} & \sum_{i=1}^{M} \sum_{\ell=1}^{L} C(i, \ell), \quad s.t., y_i^{(\ell)} \geq y_{i-1}^{(\ell)}, \ \forall i, \ell \\
C(i, l) &= \alpha[(y_i^{(\ell)} - y_i^{(\ell-1)})^2 + \beta \mathbb{I}(y_i^{(\ell)} \neq y_i^{(\ell-1)})] \\
&\quad + (1-\alpha)(||y_i^{(\ell)} - y_{i-1}^{(\ell)}|| - D_i^{(\ell)})^2
\end{aligned}
\tag{3}
$$

The first two terms of $C(i,l)$ maintain stability, and the third term maintains readability. M is the number of words in a sample, and L is the number of layers in the NLP model. $\alpha \in [0,1]$, $\beta > 0$ are hyperparameters that balance different terms in the loss. In our implementation, $\alpha = 0.4$, and $\beta = 5$.

The constrained problem defined in Eq. (3) can be solved by dynamic programming in pseudopolynomial time.

4.3.3 Word-Level Visualization

The word-level visualization consists of a **word contribution view** and a **word context view**.

Word contribution view. The word contribution view helps identify the words and layers of interest based on the words' contribution to different layers in the model (*R4*).

Visual encoding. As shown in Fig. 1(c), the x-axis denotes layers, and the y-axis corresponds to the word contribution percentiles. Words are divided into 10 equal-size groups based on their contribution at each layer. The groups with larger contributions are placed higher on the y-axis. In this way, the visualization reveals whether the model leverages the correct words for prediction. For example, we can debug a news classification model by checking whether class-relevant words like "antitrust" and "google" are placed on the top at the last few layers, and other words like "this," and "has" are placed at the bottom.

Following the corpus-level visualization, the size of a word encodes its importance. The color of a word is determined by its dominant polarity in its associated samples. A pie-chart-based glyph 🥧 is used to show the distribution of word polarity in all the associated samples. To reduce visual clutter, we only show pie charts for the top five most important words at each layer.

Layout. The words are placed based on the sweepline algorithm [40], which places important words close to the centroid of a given contour. We slightly modify the algorithm by placing each word w close to its desired position. This is achieved by replacing the centroid with $(\tilde{x}_w^{(\ell)}, \tilde{y}_w^{(\ell)})$. Here, $\tilde{y}_w^{(\ell)}$ is the contribution percentile of w. To ensure stability, we try to maintain the relative position of $\tilde{x}_w^{(\ell)}$. If the word usually appears at the left (or right) side of previous layers, we prefer to place it at the left (or right) side of the current layer.

Interaction. This view is coordinated with other views to understand the model from the word perspective. For example, the words displayed will be updated upon the selection of samples or words in other views. We can hover over a word to inspect its polarity distribution over samples with a pie chart and highlight its appearances across layers. To help identify interesting words, we enable two types of automatic pattern searching functions based on the the experts' suggestions. Trending button ⤵ (or ⤴) in the top right corner is used to show the words whose contributions keep decreasing (or increasing) through layers (Fig. 7C or D).

Word context view. The word context view facilitates the understanding of a word by revealing how the model processes it based on its context (*R5*).

The context of word w is depicted by a list of words that are considered to be the most relevant to it according to the NLP model. The word context view illustrates the context of w across different *samples* and *layers*. Fig. 7(c) shows an example context visualization of the word "like." Each rectangle in the visualization represents a cluster of samples with similar context words for "like." The size of the rectangle encodes the number of samples in the cluster, and the color encodes the majority polarity of "like" in these samples. For example, the meaning of "like" in cluster F at layer 11 is "favor," and its sentiment is mostly positive (e.g., "if you like" and "might like"). Phrases, such as "if you like" are extracted by identifying words that are both relevant and adjacent to "like" in the samples, and the font size is used to encode the importance score. The sample clusters are computed by performing agglomerative clustering [34] on the word context vectors (Fig. 6(a)). We initialize the clusters at each layer by using the clusters extracted in the previous layer to maintain stability. The cluster positions are determined by using the directed acyclic graph layout algorithm employed in TextFlow [41]. The width of the edge encodes the proportion of samples that come from the previous cluster. To reduce visual clutter, we only show the results of representative layers, whose similarity with the previously selected representative layer is smaller than a threshold.

The word context view is coordinated with other views during the analysis. It is triggered when a user selects a word in the distribution view or the word contribution view. By selecting a rectangle in the word context view, the corresponding samples are highlighted in the distribution view and sample list as well.

5 CASE STUDIES

We conducted case studies involving three tasks with experts E_3 and E_4. E_3 is interested in understanding and diagnosing BERT

models for binary/multi-class classification tasks, and E_4 would like to compare different models using DeepNLPVis.

In conducting the case studies, DeepNLPVis requires the inter- and intra-word information for each sample in the training set. The calculation for the whole set is time-consuming and is thus carried out offline. All the other required data can be obtained with real-time processing (within one sec), and are thus calculated online.

5.1 Binary Sentiment Classification

In this case study, E_3 carried out the sentiment classification task on the Stanford Sentiment Treebank (SST-2) [42], which consists of sentences from movie reviews and human annotations of their sentiments. The sentiments are of two classes: positive and negative. The GLUE SST-2 splits [43] were used for the training (67,349 samples), validation (872 samples), and test (1,821 samples) sets. The BERT model, as the most widely used NLP model, was applied as the baseline and achieved 93.23% accuracy. Starting with the BERT model, E_3 emphasized on gaining a comprehensive understanding of the model's working mechanism, which would in turn facilitate the subsequent model diagnosis.

5.1.1 Understanding

Understand the overall performance (R1, R4). E_3 began the analysis by examining how the prediction scores were distributed over the data in the distribution view (R1). He immediately noticed the long-tailed distributions (Fig. 7(a)), showing that most samples were predicted with high confidence (vertically away from the center), while those with low confidence were horizontally closer to the center. E_3 also noticed the keywords extracted from the positive and negative samples had relevant sentiments. The overall distribution in the distribution view gave E_3 confidence in the model's performance. He then turned his attention to the word contribution view (Fig. 7(b)) to analyze how the model understands the words at different layers (R4). He noticed that with the layers going deeper, the model recognized more words with sentiment tendencies, i.e., more colored words. Moreover, at layers 11 and 12, words with strong positive or negative meanings (e.g., "absurd," "laughs," "charm") contributed highly. These sentimental words replaced those class-irrelevant ones (e.g., "flick," "screen," "filmmaking") in previous layers.

Analyzing how the contributions of words changed through the layers (R4) is helpful to understand the model's working mechanism. Thus, E_3 clicked the trending hint buttons to display the top two words with the largest contribution changes (decreasing or increasing). He noticed that the contributions of "movie" and "film" decreased slowly until layer 10 and then decreased rapidly (Fig. 7C). While the contributions of "laughs" and "care" increased rapidly in the last few layers (Fig. 7D). This observation verifies his hypothesis that the BERT model, as a fine-tuned model, has the early layers more dedicated to learning transferable representations of language that are invariant to the prediction task, while the deeper layers paying more attention to words that are relevant to the prediction task [44].

The word contribution view gave E_3 an overview of how the model gradually adapted to the tasks through the layers. To get a deeper understanding of the model, E_3 then analyzed how it understood the meaning of individual words and processed individual samples.

Understand words in context (R5). Some keywords in the distribution view aroused E_3's attention. He noticed that the word
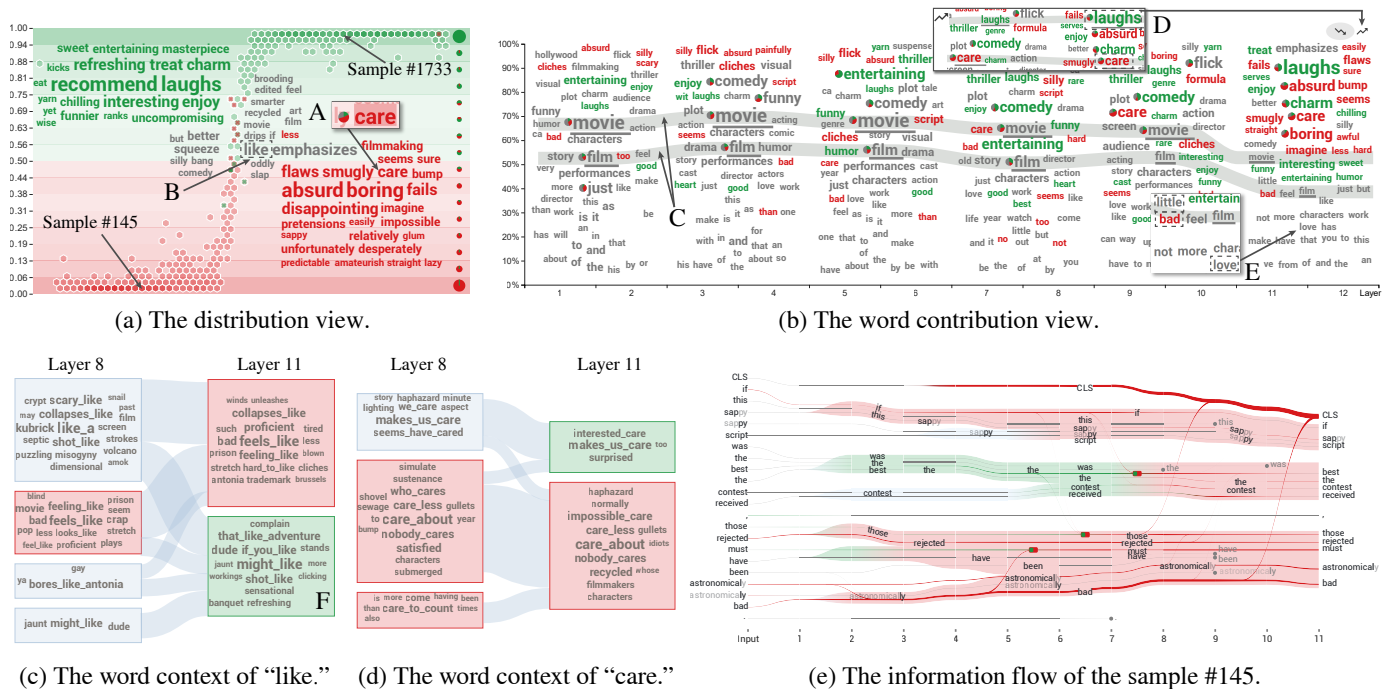
(a) The distribution view.

(b) The word contribution view.

(c) The word context of "like."   (d) The word context of "care."   (e) The information flow of the sample #145.

Fig. 7: The analysis of the BERT model on sentiment classification.

"care" (Fig. 7A) contributed more to negative predictions that went against his intuition. He thus examined its word context view (Fig. 7(d)). He observed that when the samples went through the layers, the model tended to embed "care" in a negative context , such as "care less," "nobody cares," "who cares." Only a small number were embedded in a positive context. This is due to the more frequent presence of "care" in the negative samples (191 negative and 100 positive). Considering the SST2 data came from movie reviews, E_3 thought it was reasonable because "care" is indeed found more often in negative reviews.

E_3 further noticed the word "like" that appeared to be class-irrelevant rather than positive (Fig. 7B). Examining its word context view (Fig. 7(c)), there were four main clusters formed at layer 8. Three of them were class-irrelevant, i.e., non-apparent sentiment tendency at this layer, while one of them was negative. The negative cluster contained "feels like," "looks like," where "like" was a proposition with no strong sentiment tendency. However, the negative context in this cluster, such as "crap," "blind," and "criticizing," associated this cluster with a negative sentiment. Then through the further exchange of context, the class-irrelevant clusters split. Parts of them fused into the negative cluster, while other parts formed a positive cluster at layer 11. In the positive cluster, "like" appeared more in phrases such as "if you like," and "might like" with the meaning of "favor." From the word context view, E_3 was more confident in the BERT model's ability to disambiguate words with multiple meanings.

Understand the prediction of samples (R3). To understand how the model processes a sample across layers, E_3 selected the samples of interest by coordinating the distribution view and sample list. He first selected a set of samples with the highest confidence in the distribution view and then examined their content in the sample list to find samples with interesting structures. By repeating this step, he finally selected samples #145 and #1733 (Fig. 7(a)).They are predicted as negative and positive with turning structures.

Fig. 7(e) shows the information flow for predicting the negative sample #145 "if this sappy script was the best the contest received,

those reject must be astronomically bad." This sample is of a turning structure with the word "if." At layer 1, the phrase "contest received" was initially formed with no-apparent sentiment tendency. Then at layer 6, "was the best the contest received" was formed together and regarded as positive due to the positive sentiment of "best." However, it turned into negative at layer 7 where the information from "if" was transferred to it, which indicates that the model correctly interpreted the turning structure. Before layer 6, the information was transferred more locally such as from 'astronomically bad' to "must," changing its sentiment into negative. At layers 6 and 7, the information from "if" was transferred to the second half of the sentence. So far the model recognized the overall structure, and then at layers 7 and 10, it transferred the information from the second half of the sentence to [CLS], which ultimately determined the final negative prediction of this sample.

From this negative sample, E_3 also gained insights into how the model understands a sentence. At the beginning layers, words and phrases were formed, such as "contest received," "was the best," and "astronomically bad". Along with the layers going deeper, more sentiment information transfers were observed between words and phrases. It shows that the model was mostly devoted to understanding the sentence structure at early layers, while shifted its attention to transferring information relevant to the prediction task.

E_3 had similar observations for the positive sample #1733 "though the film is static, its writer-director's heart is in the right place, his plea for democracy and civic action laudable." Again, this sample has a turning structure with the word "though," which was recognized at layer 8. The recognition changed the sentiment of "the film is static" to positive, which, together with the positive sentiment of "is in the right place" and "laudable" resulted in the final positive prediction.

5.1.2 Diagnosis

With a deeper understanding of how the model understands words and samples, E_3 then attempted to identify the deficiencies in the current model and improve its performance (R2).
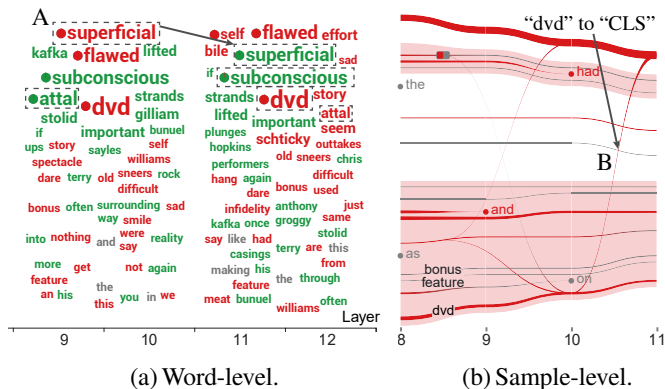
(a) Word-level. (b) Sample-level.

Fig. 8: Diagnosis of misclassified samples.

He selected all the misclassified samples in the distribution view (the red and green crosses in Fig. 7(a)) and turned to the word contribution view (Fig. 8(a)) to examine the important words for predicting them. He first noticed the word "superficial" (Fig. 8A), which remains strongly negative but turns to positive in the last layer. Wondering why, he clicked on it to examine the associated samples in the sample list and found that there was a conflicting label for "a superficial way" (positive) and "the superficial way" (negative). As a fine-tuned model, the last few layers of BERT are more influenced by the training samples. E_3 considered the conflicting label confused the model about the sentiment of "superficial" in the last layer and resulted in some of the misclassifications. He then corrected the conflicting labels for the two samples.

Another word that drew his attention was "dvd," which was an important word through the layers. However, it was strongly negative, which went against his intuition. Checking the associated samples, he found that there were far more negative than positive samples that included "dvd" (39 negative and 14 positive). E_3 thus considered this was a case of the shortcut issue [45] caused by the limited diversity of training data. Shortcut refers to the phenomenon that a model learns spurious correlations between words and labels, e.g., classifying a word to be positive/negative only based on its occurrence in according samples rather than understanding its inter/intra-word relationships. To see the influence to prediction, E_3 then turned to an individual sample to check the information flow. He selected sample #76, which is a positive sample but wrongly predicted as negative. The information flow of this sample (Fig. 8(b)) showed that "dvd" was regarded as a negative word from the beginning and throughout the layers. Its negativity was passed to [CLS] at layer 10 and contributed to the final prediction (Fig. 8B). This confirmed that the spurious correlation between "dvd" and negative label was indeed an important reason for the wrong prediction. E_3 thus decided to remove "dvd" from all 53 samples, as class-irrelevant words can be removed in text classification without sacrificing accuracy [46].

From the analysis of "superficial" and "dvd," E_3 wondered whether conflicting labels and shortcut issues might be present in other important words, accounting for most of the wrong predictions. He thus hovered over the words in the word contribution view and found other words that have the same problems (e.g., "attal," "subconscious"). All together, E_3 corrected two conflicting labels ("superficial," "attal") and removed two words ("dvd," "subconscious") with sample bias. After fine-tuning the model,
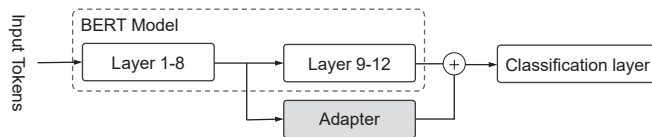

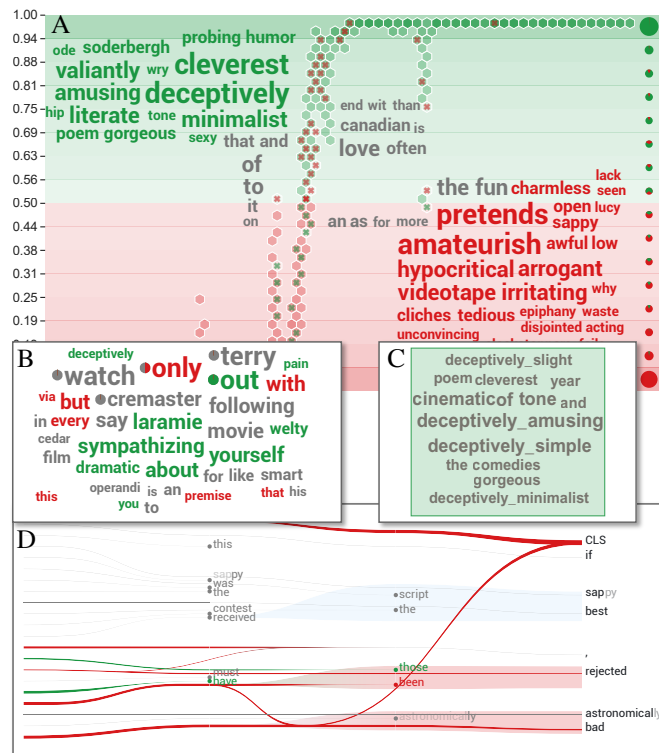Fig. 9: Improving the BERT model by adding an adapter module.

Fig. 10: The analysis of ELMO+LSTM model.

the eight previously misclassified samples were now predicted with correct sentiments.

E_3 was also interested in other deficiencies in the model besides the data problem. He noticed that in the last two layers, several words with strong positive or negative tendencies, such as "love," "little," "bad," still do not contribute highly, although they have the majority of the associated samples in the according sentiment class (Fig. 7E). For example, "love" occurred in 641 positive samples and 147 negative negative samples, and the similar were observed for "little" and "bad." This indicated that the model was still coarse-grained for the specific task. Therefore, E_3 adopted the adapter method to enhance the model performance by inserting an adapter module into the last four layers (Fig. 9). He followed the adapter architecture proposed by Houlsby et al. [47] with the hidden layer size of 64 and the tanh activation function. The parameters of the original fine-tuned model were frozen, and the new adapter module was trained. After training, the importance of "love," "little," and "bad" increased, showing that the ability of the new model to process these class-relevant words was improved. Accordingly, the test accuracy of the new model was increased from 93.23% to 93.92%.

5.1.3 Model Comparison

E_4 was interested in using the unified measure to directly compare how different models work. He carried out the sentiment classification task on the same SST-2 dataset using two models.

One is the 12-layer BERT model. The other is the 4-layer Bi-LSTM model with input enhanced by a pre-trained ELMo+LSTM model [48]. It achieved 98.28% accuracy on the training set and 88.53% accuracy on the test set. E_4 then imported each of the models into DeepNLPVis for comparison.

From the distribution views (Fig. 7(a) and Fig. 10A), E_4 first noticed that there were many more misclassified samples using ELMo+LSTM than using BERT. Further examination showed that there was an obvious difference in the keywords extracted. The keywords extracted in BERT are mostly common words ("laughs," "recommend," "boring," etc.), while those extracted in ELMo+LSTM included more rare words such as "valiantly," "hypocritical," "amateurish." These words are of strong sentiments but have low frequency in the SST-2 dataset. E_4 considered the pretraining from ELMo helped the LSTM model to better understand the sentiments of these rare terms.

"deceptively" was considered the most important positive keyword in ELMo+LSTM, which is against E_4's intuition. Further examination of samples showed that there were far more positive samples than negative samples containing "deceptively" (35 positive and 1 negative). E_4 thus considered ELMo+LSTM introduced the shortcut issue, the same as E_3 when analyzing "dvd." To verify this, E_4 checked the word context for "deceptively" (Fig. 10C). Many positive words, such as "amusing," "gorgeous," "cleverest," were presented in the context, which falsely supported the positive sentiment of "deceptively." E_4 considered the presence of shortcuts in ELMo+LSTM also explained its high accuracy on training data but a noticeable performance drop on the unseen test data.

E_4 then turned to analyze the misclassified samples in ELMo+LSTM. From the word contribution view (Fig. 10B), he found that the important words included quite a few names (e.g., "laramie," "terry"), nouns (e.g., "movie," "yourself") and, stop words (e.g., "with," "in"), which are irrelevant to the prediction task. E_4 commented, "this is consistent with the observation in a previous study that the recurrent structure in LSTM limits its ability to filter some noisy words from the sequence [12]. It is undesirable for sentiment classification." E_4 then compared the information flow of sample #145 using the two models. He observed that the information transition in ELMo+LSTM (Fig. 10D) was more local, showing the limitation of the ELMo+LSTM model in dealing with the association in non-adjacent words. Unlike BERT, the successful prediction was largely based on several strong negative words, such as "reject" and "bad," rather than recognizing the turning structure of the sample.

The comparison showed superior performance of BERT over ELMo+LSTM on the sentiment classification task. E_4 thus considered to improve the ELMo+LSTM model using the knowledge distilled from the BERT model. He followed the knowledge distillation method in [49], where the ELMo+LSTM model was the student model while the BERT was the teacher model. The idea of knowledge distillation is to refine the student model by minimizing the difference (measured using KL divergence) between its output and the teacher model's output, and thus to improve the performance of the student model. In this case, after knowledge distillation, the test accuracy of the new ELMo-LSTM model was increased from 88.53% to 89.33%.

5.2 Multi-Class Classification

E_3 was satisfied with the assistance of DeepNLPVis for analyzing sentiment classification. To see how this assistance generalizes to other classification tasks, he performed a further classification task on news. A subset was randomly selected from the AG News topic classification dataset [50]. There are four classes: "world", "sports", "business", "sci/tech", each containing 30,000 training samples and 1,900 testing samples. The BERT model achieved 94.79% test accuracy.

To unify the analysis, a confusion matrix (Fig. 1(a)) was provided to select two classes for further analysis. E_3 observed that the greatest confusion was between the "business" and the "sci/tech" news. He selected the two classes and then, following a similar process in the first case, started the analysis with the distribution view (Fig. 1(b)). He observed that most samples of the two classes were distributed apart (left and right). However, a small number of "business" news (orange) were horizontally closer to the "sci/tech" distribution (blue), indicating these "business" news have some similarities to "sci/tech" news (Fig. 1A). It seemed consequential that there were more "business" news misclassified as "sci/tech" news than the other way around. E_3 then selected these "business" news samples in the middle of the distribution view for further examination in the word contribution view (Fig. 1(c)). He noticed that there was a significant number of important words that are typically related to technology, such as "antitrust," "google," "hp," etc. Checking in the sample list, he further found many of these samples were "business" news related to "sci/tech" companies. Their content is indeed similar to "sci/tech" news.

Among these technology words, "google" was of high importance throughout the layers and drew the attention of E_3. Checking its associated samples in the sample list, there were quite a few misclassified samples. Selecting the one with the lowest confidence, E_3 turned to check its information flow (Fig. 1(e)). The selected sample, "Google IPO faces Playboy slip-up The bidding gets underway for Google's public offering, despite last-minute worries over an interview with its bosses in Playboy magazine," was a "business" news. However, "google," which was regarded as a "sci/tech" word throughout the layers, played an important role in the model's understanding of the sample. The two "googles" had the "sci/tech" tendency from the beginning of the layers. When the first "google" transferred its information to "google's" at layer 2 (Fig. 1B), the tendency was reinforced and then passed to the [CLS] at layer 10 (Fig. 1C). This contributed to the misclassification.

6 EXPERT FEEDBACK AND DISCUSSION

After the case studies, five semi-structured interviews were conducted with the experts we worked with. In the interview, we first introduced the visual design and interactions, and then explored the tool together with the experts through an example case. Each of the interviews took 50-70 minutes. Overall, the experts gave positive feedback on the usability of DeepNLPVis. They also pointed out several limitations that provide opportunities for future research.

6.1 Usability

Informative visualization and deeper understanding. All the experts agreed that the visualization was informative and helped them deeply understand the models. They particularly mentioned that the polarity in the distribution view, the most contributed and the less contributed words in the word contribution view, and the phrase formation in the information flow view facilitated their understanding of the models. E_3 commented, "The polarity of contribution is very useful to detect the shortcut issue that learns spurious correlations between words and category labels (e.g., "dvd"

in the first case). Such spurious correlations help identify which types of words (e.g., a single word, a word in context, or ordered pairs) lead to the limited diversity of training data. Knowing the causes, I can enhance the data accordingly." Although the experts took 25.5 minutes on average (STDEV=2.89) to get familiar with the tool, they believed that the enlightening information and deeper understanding gained through the exploration deserved the efforts.

Improving analysis efficiency. The experts especially liked the analysis process driven by a set of interactions. They commented that existing tools, such as TensorBoard, only allowed them to examine the samples one by one. Without a comprehensive understanding of the training process from different perspectives, diagnosing a performance issue typically relied on a time-consuming trial-and-error process. After trying DeepNLPVis, the experts praised its efficient analysis process brought by the integral exploration at the corpus, sample, and word levels. E_1 said, "The analysis process from the sample distribution to the word contribution and the information flow inside a sample looks natural to me. The interactions enable me to find interesting information quickly. For example, the trending hint button in the word contribution view helps me identify the changing patterns of the words of interest, especially those conflicted with my intuition."

Promoting effective communication in deployment. In the interview, the experts were impressed with the explanation capability of DeepNLPVis and the provided informative information. They believed that it could be used for effective communication between different sectors inside an institution. This is because the employed intuitive visualization provides a common ground for communication between different types of practitioners, such as model developers, consumers, and project managers. For example, E_4 commented that the visual explanation provided by the information flow view could well explain how the model worked at the sample level. Such explanation is very helpful to illustrate the developed deep NLP model to the model consumers who are not machine learning experts, e.g., the developers in a product group. With a clear understanding of the model, the developers could better maintain it in the product.

6.2 Limitations and Future Work

Task Generalization. In the prototype, text classification is used as an example to illustrate how DeepNLPVis supports the unified understanding of NLP models. Although the prototype supports the analysis of the classification tasks with a pair of sentences as input, it cannot distinguish intra-sentence and inter-sentence relations. To handle this problem, we consider designing proper visual encodings to distinguish the difference between these two relationships and enable a better analysis of such tasks. In addition to classification, the experts also express the need to apply DeepNLPVis to other tasks such as text summarization, machine translation, and question answering. These tasks can be regarded as a multi-class classification task with a relatively larger class number. In these tasks, the class number equals to the number of words/phrases. As a result, for these tasks, how to handle a large class number in visualization is a key challenge faced in the future work.

Visual scalability. The experts mentioned that larger NLP models, such as GPT-3, usually contained dozens of layers, and a sample might contain hundreds or even thousands of words. With the increased number of layers and sample length, the scalability issue will arise in the word-level and sample-level visualizations. A possible solution is to utilize the layer clustering technique

and overview + detail visualization. An example is the flow visualization that is affected by the sample length, as each line represents a word. This visualization will quickly become cluttered if hundreds or thousands of words are included in a sample. The experts indicated that their analysis usually started from representative samples and representative words in each sample. Thus, in the future, we are interested in identifying these representative samples and words to balance the informativeness and readability of this visualization.

Model refinement. After identifying the performance issues, the experts prefer a mechanism that tightly integrates interactive visualization with machine learning to refine the model semi-automatically rather than to improve the model architecture manually. For example, in sentiment analysis, if the expert corrects the sentiment of several words through the visualization, s/he would expect the model can be automatically refined based on the corrections. Thus, an interesting direction for future work is to explore how to transform the provided feedback into a prior or constraint for the model and progressively refine the model. Another related interesting direction is how to integrate active learning into the system to give more hints and reduce the number of samples to be verified by users.

7 CONCLUSION

We have presented a visual analysis method, DeepNLPVis, to facilitate a unified understanding of deep NLP models. This method is built upon an information-based measure to illustrate how a deep NLP model maintains the information of input words in a sample with a multi-level visualization. The effectiveness and usefulness of our method are demonstrated through case studies, in which the experts utilize DeepNLPVis to understand and analyze the model behaviors in text classification tasks and explore the root causes of the successful and unsuccessful cases. The experts are generally satisfied with the developed method as it provides a unified understanding of different deep NLP models allow them to conveniently compare different types of models. Moreover, it helps identify the underlying reason for low performance and thus makes informed improvements in the models.

REFERENCES

[1] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: A comprehensive review," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–40, 2021.

[2] Y. Kim, "Convolutional neural networks for sentence classification," in *the Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1746–1751.

[3] 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*, vol. 24, no. 1, pp. 667–676, 2018.

[4] 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*, 2017, pp. 5998–6008.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171–4186.

[6] A. Y. Kim and J. Hardin, ""playing the whole game": A data collection and analysis exercise with google calendar," *Journal of Statistics and Data Science Education*, vol. 29, no. sup1, pp. S51–S60, 2021.

[7] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu, "A survey of visual analytics techniques for machine learning," *Computational Visual Media*, vol. 7, no. 1, pp. 3–36, 2021.
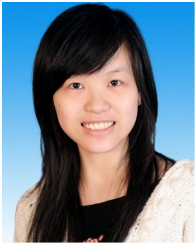
[8] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, G. Cao, D. Jiang, M. Zhou *et al.*, "K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP*, 2021, pp. 1405–1418.

[9] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu, "Understanding hidden memories of recurrent neural networks," in *IEEE Conference on Visual Analytics Science and Technology*, 2017, pp. 13–24.

[10] J. F. DeRose, J. Wang, and M. Berger, "Attention flows: Analyzing and comparing attention mechanisms in language models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1160–1170, 2021.

[11] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *arXiv preprint arXiv:2105.01601*, 2021.

[12] C. Guan, X. Wang, Q. Zhang, R. Chen, D. He, and X. Xie, "Towards a deep and unified understanding of deep neural models in NLP," in *International Conference on Machine Learning*, 2019, pp. 2454–2463.

[13] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *International Convention on Information and Communication Technology, Electronics and Microelectronics*, 2018, pp. 0210–0215.

[14] Y. Ming, P. Xu, H. Qu, and L. Ren, "Interpretable and steerable sequence learning via prototypes," in *ACM International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 903–913.

[15] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller, "How to explain individual classification decisions," *Journal of Machine Learning Research*, vol. 11, pp. 1803–1831, 2010.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *ACM International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

[17] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, "Squares: Supporting interactive performance analysis for multiclass classifiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 61–70, 2017.

[18] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert, "Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 364–373, 2019.

[19] P. Chawla, S. Hazarika, and H.-W. Shen, "Token-wise sentiment decomposition for convnet: Visualizing a sentiment classifier," *Visual Informatics*, vol. 4, no. 2, pp. 132–141, 2020.

[20] Z. Dong, T. Wu, S. Song, and M. Zhang, "Interactive attention model explorer for natural language processing tasks with unbalanced data sizes," in *IEEE Pacific Visualization Symposium*, 2020, pp. 46–50.

[21] 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*, vol. 24, no. 1, pp. 77–87, 2018.

[22] X. Ji, Y. Tu, W. He, J. Wang, H.-W. Shen, and P.-Y. Yen, "Usevis: Visual analytics of attention-based neural embedding in information retrieval," *Visual Informatics*, vol. 5, no. 2, pp. 1–12, 2021.

[23] S. Gehrmann, H. Strobelt, R. Krüger, H. Pfister, and A. M. Rush, "Visual interaction with deep learning models through collaborative semantic inference," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 884–894, 2020.

[24] S. Liu, Z. Li, T. Li, V. Srikumar, V. Pascucci, and P.-T. Bremer, "Nlize: A perturbation-driven visual interrogation tool for analyzing and interpreting natural language inference models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 651–660, 2019.

[25] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," in *the Workshop at International Conference on Learning Representations*, 2016.

[26] J. Li, X. Chen, E. Hovy, and D. Jurafsky, "Visualizing and understanding neural models in NLP," in *the Conference of the North American Chapter of the Association for Computational Linguistics*, 2016, pp. 681–691.

[27] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019.

[28] D. Cashman, G. Patterson, A. Mosca, N. Watts, S. Robinson, and R. Chang, "RNNbow: Visualizing learning via backpropagation gradients in rnns," *IEEE Computer Graphics and Applications*, vol. 38, no. 6, pp. 39–50, 2018.

[29] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush, "Seq2Seq-Vis: A visual debugging tool for sequence-to-sequence models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 353–363, 2019.

[30] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo, "RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 299–309, 2019.

[31] I. Tenney, J. Wexler, J. Bastings, T. Bolukbasi, A. Coenen, S. Gehrmann, E. Jiang, M. Pushkarna, C. Radebaugh, E. Reif *et al.*, "The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models," in *the Conference on Empirical Methods in Natural Language Processing*, 2020, pp. 107–118.

[32] Y. Yang, R. Khanna, Y. Yu, A. Gholami, K. Keutzer, J. E. Gonzalez, K. Ramchandran, and M. W. Mahoney, "Boundary thickness and robustness in learning models," in *Advances in Neural Information Processing Systems*, 2020, pp. 6223–6234.

[33] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. Van Durme, S. R. Bowman, D. Das *et al.*, "What do you learn from context? probing for sentence structure in contextualized word representations," in *International Conference on Learning Representations*, 2018.

[34] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[35] L. Van Der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[36] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[37] I. Yahav, O. Shehory, and D. Schwartz, "Comments mining with tf-idf: the inherent bias and its removal," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 3, pp. 437–450, 2018.

[38] Y. Tanahashi and K.-L. Ma, "Design considerations for optimizing story-line visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2679–2688, 2012.

[39] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu, "Storyflow: Tracking the evolution of stories," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2436–2445, 2013.

[40] L. Shi, F. Wei, S. Liu, L. Tan, X. Lian, and M. X. Zhou, "Understanding text corpora with multiple facets," in *IEEE Conference on Visual Analytics Science and Technology*, 2010, pp. 99–106.

[41] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong, "Textflow: Towards better understanding of evolving topics in text," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2412–2421, 2011.

[42] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *the Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1631–1642.

[43] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *International Conference on Learning Representations*, 2019.

[44] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, "On identifiability in transformers," in *International Conference on Learning Representations*, 2020.

[45] T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," in *the Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3428–3448.

[46] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classification via reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2018.

[47] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*, 2019, pp. 2790–2799.

[48] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *the Conference of the North American Chapter of the Association for Computational Linguistics*, 2018, pp. 2227–2237.

[49] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[50] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

Zhen Li is a first-year Ph.D. student of Software School, Tsinghua University. His research interest is explainable artificial intelligence. He received a B.S. degree from Tsinghua University and a M.Phil. degree from Hong Kong University of Science and Technology.

Xiting Wang is a senior researcher at Microsoft Research Asia. Her research interests include explainable machine learning and visual text analytics. She has published academic papers on reputable international conferences and journals in her research area, such as KDD, TKDE, AAAI, IJCAI, TVCG and VAST. One of her first author papers has been chosen as the TVCG spotlight article for Dec. 2016. She is a senior program committee member of AAAI and is a program committee member of many top conferences.

Weikai Yang is a second-year Ph.D. student at Tsinghua University. His research interests lie in integrating the Machine Learning into Visual Analytics, which can facilitate the understanding of large-scale data and make it easier for the practitioners to use the machine learning techniques.

Jing Wu is a lecturer in computer science and informatics at Cardiff University, UK. Her research interests are in computer vision and graphics including image-based 3D reconstruction, face recognition, machine learning and visual analytics. She received BSc and MSc from Nanjing University, and Ph.D. from the University of York, UK. She serves as a PC member in CGVC, BMVC, etc.

Zhengyan Zhang is a second-year Ph.D. student of the Department of Computer Science and Technology, Tsinghua University. His research interests include natural language processing and social computing. He has published papers in international conferences and journals, including ACL, EMNLP, and TKDE.

Zhiyuan Liu is an associate professor at Tsinghua University. He got his BEng degreeand his Ph.D. from Tsinghua University. His research interests are natural language processing, information extraction, knowledge graphs, and social computation. He has published over 80 papers in international journals and conferences, including ACM/IEEE Transactions, AAAI, IJCAI, ACL, and EMNLP. He has also served as PC/Area Chair of several international conferences, including ACL, EMNLP, WWW, CIKM, COLING, etc.

Maosong Sun is a professor at Tsinghua University. He got his BEng degree and MEng degree from Tsinghua University, and got his Ph.D. degree from City University of Hong Kong. His research interests include natural language processing, Chinese computing, Web intelligence, and computational social sciences. He serves as a vice president of the Chinese Information Processing Society, the council member of China Computer Federation, and the Editor-in-Chief of the Journal of Chinese Information Processing.
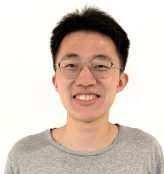
Hui Zhang is an Associate Professor at School of Software, Tsinghua University, China. She received her B.Sc. and Ph.D. in Computer Science from Tsinghua University, in 1997 and 2003, respectively. Her research interests include computer aided design and computer graphics.

Shixia Liu is a professor at Tsinghua University. Her research interests include explainable artificial intelligence, visual text analytics, and text mining. She worked as a research staff member at IBM China Research Lab and a lead researcher at Microsoft Research Asia. She received a B.S. and M.S. from Harbin Institute of Technology, a Ph.D. from Tsinghua University. She is a fellow of IEEE and an associate editor-in-chief of IEEE Trans. Vis. Comput. Graph.