Visual Explanation for Open-domain Question Answering with BERT

Zekai Shao, Shuran Sun, Yuheng Zhao, Siyuan Wang, Zhongyu Wei,
Tao Gui, Cagatay Turkay and Siming Chen

Abstract—Open-domain question answering (OpenQA) is an essential but challenging task in natural language processing that aims to answer questions in natural language formats on the basis of large-scale unstructured passages. Recent research has taken the performance of benchmark datasets to new heights, especially when these datasets are combined with techniques for machine reading comprehension based on Transformer models. However, as identified through our ongoing collaboration with domain experts and our review of literature, three key challenges limit their further improvement: (i) complex data with multiple long texts, (ii) complex model architecture with multiple modules, and (iii) semantically complex decision process. In this paper, we present VEQA, a visual analytics system that helps experts understand the decision reasons of OpenQA and provides insights into model improvement. The system summarizes the data flow within and between modules in the OpenQA model as the decision process takes place at the summary, instance and candidate levels. Specifically, it guides users through a summary visualization of dataset and module response to explore individual instances with a ranking visualization that incorporates context. Furthermore, VEQA supports fine-grained exploration of the decision flow within a single module through a comparative tree visualization. We demonstrate the effectiveness of VEQA in promoting interpretability and providing insights into model enhancement through a case study and expert evaluation.

Index Terms—Open-domain Question Answering, Explainable Machine Learning, Visual Analytics

1 Introduction

Question answering (QA) is an area of information retrieval (IR) and natural language processing (NLP) that focuses on building a model that automatically answers questions posed by humans in natural language formats. Open-domain QA (OpenQA) allows machines to provide accurate answers to users' questions without a given context and is considered the ultimate goal of QA research. With the support of this technology, modern search engines, such as Google and Bing, can not only return a list of relevant snippets or hyperlinks based on user queries as questions, but also generate appropriate answers to these questions that harmonize the search results. [20]. These search engines utilize queries as an input to an OpenQA model with the output of the model as the direct answer, thus offering enhanced user experience and efficiency. For example, asking the question "who was the first person to set foot on the moon?" to a search engine leads to the answer " Neil Armstrong" along with links for further reading. This area is growing intensely. ChatGPT [58], a recently released chatbot, has taken the Internet by storm with its advanced conversational abilities. It is based on a large QA model and can give consistent, accurate, creative answers. QA, especially OpenQA, is now an important research topic.

The modern methods for OpenQA consist of two parts: Retriever and Reader [94]. Given one question, the Retriever searches relevant passages from large unstructured corpora as top-k passages, and the Reader generates answers from these passages. With the advancement of deep learning techniques such as RNN [53] and Transformer [80], the Reader performs like a neural machine reading comprehension (MRC) model, such as BERT [17], and infers answers. Meanwhile, the Retriever can be considered an IR system that can be implemented by Transformer-based modules and retrieves passages. Considering the diversity of OpenQA models, this paper discusses models that adopt BERT as the basic architecture for the Retriever and the Reader.

Despite the rapid progress in OpenQA, the existing architecture can still be improved. For example, as illustrated by the latest survey for OpenQA [94], the retrieval effectiveness of models, namely, the ability to separate relevant passages from irrelevant ones for a given question, remains limited. Sometimes the model does not retrieve relevant passages, and sometimes the model detects noisy passages that contain the exact terms in the question but are irrelevant to the answer. Several studies [32], [41] have focused on enabling the modern neural Retriever to have a greater retrieval ability with a speed that is close to that of traditional IR systems. However, the behavioral logic of existing techniques that focus on the optimization of model architectures and training methods [60], [63], [66] have not been sufficiently demonstrated. Therefore, this paper provides visual explanations for model decision flow, thus providing experts insights into model improvement.

Through a collaboration with domain experts and a review of literature, we identify three main challenges in interpreting OpenQA models. Firstly, since OpenQA is an open-ended task that needs to select multiple relevant long passages from large corpora and involves the understanding and processing of natural language, the explanation of the model is difficult. Secondly, an OpenQA model consists of multiple modules each of which is a complex black box with numerous connections and parameters. Please note here that while **model** refers to the system or pipeline required to complete a whole OpenQA task, **module** refers to an independent structure or component that completes a certain independent part of a task within the model. Understanding the coordination of

- *Zekai Shao, Shuran Sun, Yuheng Zhao, Siyuan Wang, Zhongyu Wei, Tao Gui and Siming Chen are with Fudan University. S. Chen is the corresponding author. E-mail: {zkshao19, srsun20, yuhengzhao, wangsy18, zywei, tgui, simingchen}@fudan.edu.cn.*
- *Cagatay Turkay is with University of Warwick. E-mail: Cagatay.Turkay@warwick.ac.uk.*

Manuscript received April 19, 2005; revised August 26, 2015.

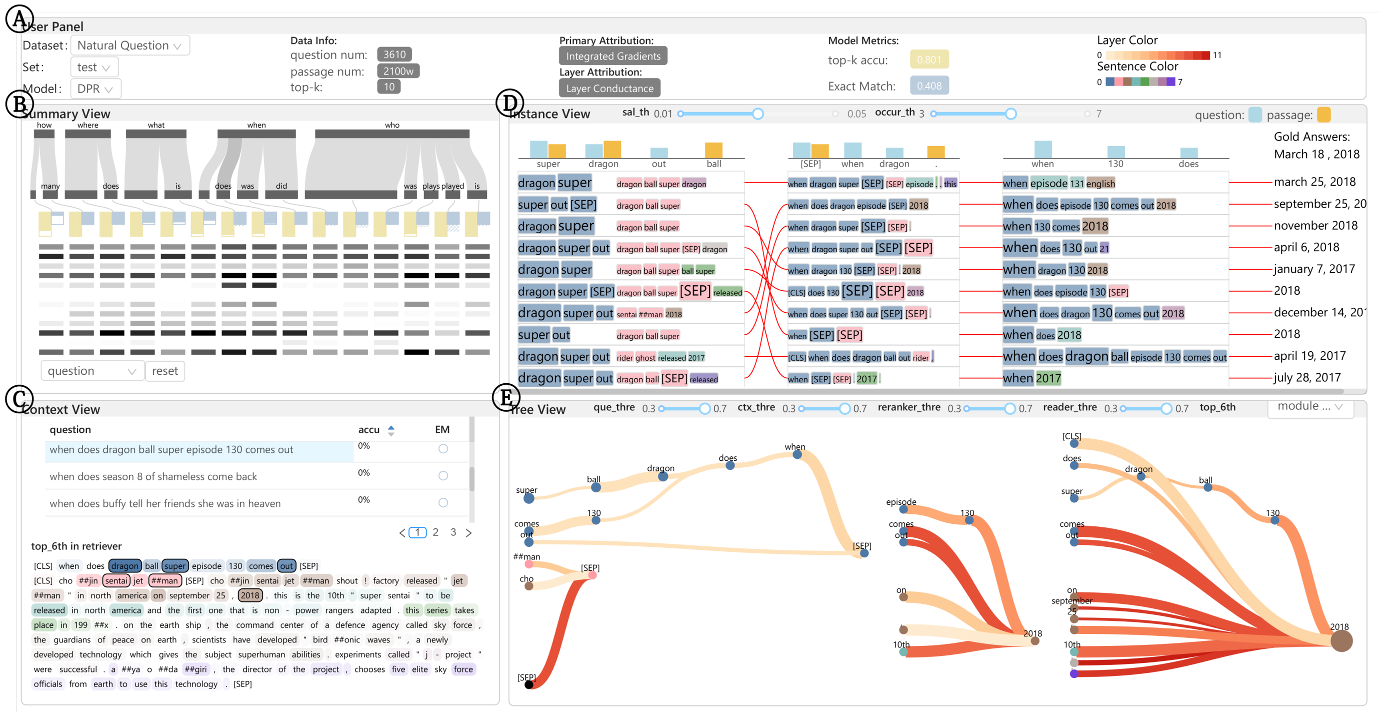


Fig. 1. Understanding the decision process of a neural model for OpenQA. The *User Panel* (A) displays the statistical information about the model and the dataset, as well as the color legends. The *Summary View* (B) provides a global summary of performance and module behavior related to each subset. The *Context View* (C) presents questions from the selected subset and context of passages retrieved for a selected question. The *Instance View* (D) summarizes the focused words of each candidate passage in different modules with ranking visualization incorporating text to analyze the selected instance. The *Tree View* (E) explains the local data flow within a single module or multiple modules in the model with comparable Sankey-tree layout.

the working of various parallel or serial modules is essential to identifying the bottleneck of a model and improving it. And finally, it is unclear how to go from explaining individual model parameters to building a holistic and semantic understanding of the decision process of a model.

Several visual analytic systems for explaining machine learning have been developed. For example, CNNVis [47] tries to help experts analyze CNNs by converting structures into directed acyclic graphs combined with various algorithms and Li et al. [45] proposes a unified structure to interpret deep NLP models for text classification. However, their method focuses on explaining a single model and is limited in exploring the internals of a multi-module model. Recent studies have investigated model interpretability for MRC tasks. For example, Ramnath et al. [61] uses t-SNE [79] for layer embeddings and an attribution method to account for the knowledge stored in BERT. QADiver [39] integrates analytical tools such as embedding analysis of hidden layers, attention matrices, and adversarial text, and offers an interactive and diagnostic framework for MRC models. However, current studies focus on MRC tasks and do not enable the analysis of the decision flow of the Retriever, which is the bottleneck that constrains the performance of OpenQA models [94] the most. In addition, in scenarios such as OpenQA, where the instance includes multiple long passages with a given question, analyzing the instance by visualizing layer embeddings with scatterplots and visualizing the attention matrices with heatmaps can potentially lead to visual clutter and often offers limited insights into the global semantic information. Our work aims to address these gaps in the visual analytics literature.

In this paper, we propose a general visual analytics system called VEQA (Fig. 1) to allow NLP experts to semantically understand the decision process of an OpenQA model and gain insights into how a model could be enhanced. Inspired by previous work [61] on semantic analysis of BERT using saliency methods, we use attribution methods to attribute the final and implicit outputs of each module of the OpenQA model in global and local levels. We also use a tree generation algorithm to capture the abstracted semantic information for understanding the decision of the OpenQA model through the analysis of module responses and instances.

Our technique uses the opening two words of a question as labels to partition the dataset into different question types and calculates the performance metrics for each subset. We also aggregate the attribution results within the module into responses for each module layer. These two items provide an overview of the module and the dataset, and guide experts in exploring the instances in the subset. At the instance-level, we use module-level attribution methods to summarize the focused words of each candidate passage in different modules, and show their distributions in a novel flow diagram with a ranking visualization incorporating contexts, which aims to help experts in understanding the similarities and differences in the decision flow across modules. Within a single module, we use layer-level attribution methods for attention matrices and hidden embeddings for generating dependency trees to express the progression of knowledge between layers through carefully designed comparable tree visualization. Through our integrated interpretability methods, users set thresholds to filter out keywords and key connections within each module, and we present the results in novel, comparable flow diagrams and tree visualizations. Overall, the visualizations provide experts an in-depth understanding of the decision process by visually exploring the complex attribution results, addressing the challenges related to the analysis of complex data with multiple long texts, and to the analysis of complex models with multiple modules.

To the best of our knowledge, ours is one of the first attempts to design a visual analytics approach to unravel the decision logic of an OpenQA model. The major contributions of this work could be listed as follows:

- Exploration of interpretability methods for Transformer-based QA and an enhanced combination of attribution methods with visualization to improve interpretability.
- VEQA, a visual analytics system to provide a multi-level explanation of the decision process of a complex multi-module model for OpenQA tasks.
- Case studies with a prevalent, representative OpenQA model and in-depth collaboration with domain experts that demonstrates the effectiveness of our approach to characterize the OpenQA decision process as a visualization problem, categorize success cases, and provide suggestions for model enhancement.

2 Related Work

In this section, we first review the literature on general visual explanation for machine learning. We then present a detailed review of related work on constructing, explaining, and visualizing Transformer-based QA models.

2.1 Visual Explanation for Machine Learning

Visualization for understanding, diagnosing, and improving machine learning models, especially deep neural networks, is eliciting profound attention, as evidenced by recent surveys [11], [92].

Visualization is usually adopted to display the structure and inherent details of models to help expert or non-expert users understand them. For example, CNNVis [47] converts CNNs into directed acyclic graphs and helps users understand the behavior of layers and neurons through clustering and edge-binding algorithms. Ming et al. [54] provided various levels of visualization to help understand RNNs by deeply mining the stored information of hidden states. Additionally, CNN Explainer [87] and Gan Lab [31] are interactive visualization tools designed for non-experts to learn and examine models.

Others have attempted to use visual evaluation methods combined with interpretability algorithms to help understand the decision-making process of models and provide insights into diagnosis and improvement. Some have focused on monitoring and diagnosing the training process of deep learning models, such as DGMTracker [46] and DQNViz [82]. Many studies have used post-hoc explanations to study model training results, and most of them are model-specific. Under this category, of special interest to our work are visualizations of the attention mechanism, which we review further in Sec. 2.4. In addition, many scholars have provided model-agnostic visual explanations, such as RuleMatrix [55], DECE [10] and M2Lens [85]. Sprinner et al. [71] designed a framework for explainable artificial intelligence (XAI) and operationalized it as explAIner, plugged into TensorBoard [1], the most widely used platform for model analysis and visualization. As for the XAI system for NLP, Li et al. [45] provided a unified interpretive method for interpreting NLP models for text classification. Attempts have also been made in the broaderer application scenarios of AI, such as healthcare [9] and autonomous driving [28], [83].

Although the aforementioned studies have achieved great success in providing visual explanations, most of them aimed to explain a single model or provide an analytical paradigm for multi-module models without explaining the internal structure. Seq2Seq-Vis [72] is an exception; it is a visual debugging tool for analyzing a five-stage text generation model by presenting the attention map and the full context while offering a projection analysis. However, this technique cannot be directly generalized to complex tasks, such as OpenQA, due to the potential visual complexity when interpreting massive, long contexts and multiple candidate passages, which will be discussed in detail in Sec. 2.4. To address the challenges posed by passages of long text and multi-module models in OpenQA tasks, we integrate interpretability methods and design novel views to explore and explain the decision flow of models.

This subsection reviews existing XAI systems and why they are difficult to apply directly to OpenQA tasks. In the next three subsections, we narrow our view to OpenQA and Transformer, and review mainstream OpenQA models (Sec. 2.2), related interpretability methods (Sec. 2.3) and visual analysis work (Sec. 2.4) to further illustrate the necessity of our work.

2.2 Transformers in OpenQA

OpenQA aims to answer a given question without any specified context. According to Zhu et al. [94], OpenQA models have evolved into a modern “Retriever-Reader” architecture. Given one question, Retriever is utilized to retrieve relevant passages from a large corpus such as Wikipedia, and Reader aims to infer the answer from the received passages.

Transformer [80] is a powerful deep neural network. The self-attention mechanism enables it to capture dependencies within long sequences. Transformer-based models, such as BERT [17], gain a large amount of knowledge by pre-training on large corpora, thus allowing the model to be fine-tuned with a small amount of data to achieve good results.

Recently, Transformer-based models have continuously allowed for performance gains in many areas of NLP, including OpenQA. Early studies [5], [56], [89] developed their Reader based only on BERT [17] or other language models, whereas current studies [32], [41] use them to develop both Retriever and Reader because language models have better semantic representations compared with sparse Retrievers based on keyword matching (e.g., TF-IDF, BM25). As a baseline for recent work, dense passage retriever (DPR) [32] uses a sophisticated sample mining and training approach to enable the potential of the dual-encoder retriever architecture, that is, using two independent BERTs to encode the question and the context separately and calculate the similarity between the two to select relevant passages. On the basis of this architecture, subsequent work includeed improvements in the calculation of similarity [33], [62], developing highly efficient training methods [60], [63], [66], and hierarchizing the retrieval process [40], [49]. However, such models act like a “black box,” and it is difficult to understand what knowledge has been accurately stored and how the models internally filter and process passages and output the final results. They lack the interpretability that is crucial for practical applications and further enhancement.

2.3 Interpretability Analysis Methods for Transformers

After the Transformer model penetrated the various fields of NLP, researchers started to explore interpretability analysis methods for Transformers in three directions.

Saliency Methods. Given a neural network parameterized by $f_c(x)$ to predict the probability of a class c, saliency methods produce a relevance score $R(x)$ of a token x denoting the relevancy of this token w.r.t. class c. The saliency methods commonly used in NLP include gradient-based [15], [43], propagation-based [6], and occlusion-based methods [2]. In other words, saliency methods can quantify whether the token is important in the decision made by the model for prediction. Saliency methods are common and considered reliable to understand the model in processing downstream tasks such as QA. Ramnath et al. [61] segregated passage words into three categories: answer, supporting, and query words. Then, they used the attribution method Integrated Gradients [76] on BERT [17] at the layer level to identify the passage words that are of primary importance at each layer for the answer.

Attention Mechanism. Given that the attention mechanism is the core part of the BERT model and has strong representation, it has always been the focus of experts. Shortly after BERT was proposed, Clark [13] and Kovaleva [36] showed that the attention distribution (attention map) of some heads/layers of BERT has specific patterns. Although controversy remains about whether attention is interpretable [29], [88] or not, a previous study [59] claimed that the distribution of attention maps often represents local aggregation, and the attribution score represents global aggregation. In addition, due to the ideological influence of saliency methods and the locality limitation of attention distribution, saliency methods combined with attention distribution [25], [35], [75], [91] have been continuously proposed.

Embedding Analysis. The combination of Dimension Reduction (DR) algorithms and visualization is a common method to analyze embeddings [21] and offering interpretability [22]. In the NLP context, researchers usually use PCA [27], t-SNE [79], UMAP [52], and other DR methods to reduce the high-dimensional word embeddings of each layer to 2D and visualize them in the form of scatter plots to understand the semantic information learned by the model through the change in distance between words. Ramnath et al. [61] visualized the t-SNE plot for the three categories of words listed above and special tokens of each layer and reported that layers that distinguish confusing answers cannot be found. Similarly, with PCA representations of tokens in different layers, Aken et al. [78] suggested that BERT for MRC goes through multiple phases while answering a question.

In general, using saliency methods to explain the importance of each token in the text results in a vector, while methods, such as attention and its variants, describing pairwise relationships between tokens result in a matrix. Ye et al. [91] verified the validity of the two abovementioned methods in QA tasks based on the assumption of counterfactual explanation and pointed out that the latter is a better explanation method than the former. Therefore, we choose the saliency method to describe global information and the attribution method for the attention matrix to describe the information flow in detail. In addition, mainstream DR methods are prone to visual confusion and loss of high-dimensional information when understanding large texts. Consequently, we choose a similar attribution method to analyze hidden embedding as a complement to attention attribution.

2.4 Visual Analytics for Interpreting Transformers

Researchers have developed several visualization methods and visual analysis systems to for experts to intuitively explore the interpretability of Transformer-based models. Early Transformer interpretability visualization tools focused on comprehensively visualizing the attention map from multiple levels (e.g., Bertviz [81]) and observing the global information learned by applying DR methods to hidden layer outputs, such as t-SNE [79] used in InterpretT [38]. Furthermore, Dodrio [86] is committed to combining model data with linguistic knowledge to help experts gain an enhanced understanding of the attention mechanism by comparing the attention weights of each part of the model with the syntactic structure and semantic information of the input text. T3vis [44] is another visual diagnosing framework that integrates almost all common analytical methods. Rather than individually inspecting attention heads or layers like the systems discussed above do, others [7], [16] pay attention to designing highly intuitive attention visualization methods. Attviz [7] focuses on exploring self-attention by different aggregations of the attention vector space for a single token, but it may not be useful in QA because it ignores token relations; Attention Flow [16] aims to provide a holistic view of the attention mechanism by deploying a radial layout, but its design is relatively complex and challenging to understand.

In visualization work on QA, Rücklé and Gurevych [65] highlighted the critical phrases in the context used to answer the question and compared between the two models. On this basis, Liu et al. [48] introduceed a hierarchical representation of attention through summary visualization to address the challenge of long sentences, but it is incompatible with today's prevalent models. QADiver [39] is another diagnosing framework for QA models, with diverse interactive visualization and analysis tools, including embedding analysis, model internals, and adversarial text, for a full pipeline of the attention-based QA model. However, directly understanding the overview of attention with a pure attention map is difficult, even though it provides cropping operations. Aside from traditional MRC and QA tasks, other specific branches of QA, such as visual question answering (VQA), have also elicited the attention of researchers. VisQA [30] is a visual analytics tool that explores questions of reasoning and bias exploitation by exposing attention maps in Transformers. It summarizes each attention head with a scalar that presents attention maps intensity and encodes functions of heads with stacked bar-charts.

Considering the complexity of the OpenQA model, we argue that a complete examination of the attention matrix of models one by one is laborious and unnecessary. In this study, we adopt a tree generation algorithm to summarize the multi-head attention of layers into dependency trees and aggregate the responses of heads to the layer level.

3 Overview

We introduce the background for the mainstream transformer-based OpenQA models, analyze tasks to be completed in our visual analytics system, and derive design requirements.

3.1 Background

The relevant background consists of two parts, attention mechanisms in Transformer [80] and Transformer-based OpenQA models.

3.1.1 *Multi-head attention mechanism*

Generally, a Transformer-based module in the OpenQA model is formed by stacking a series of layers, each of which contains multiple attention heads working in parallel to capture information from different feature subspaces [80]. Given the input lengths of N, for the h-th head in the l-th layer, the attention module calculates

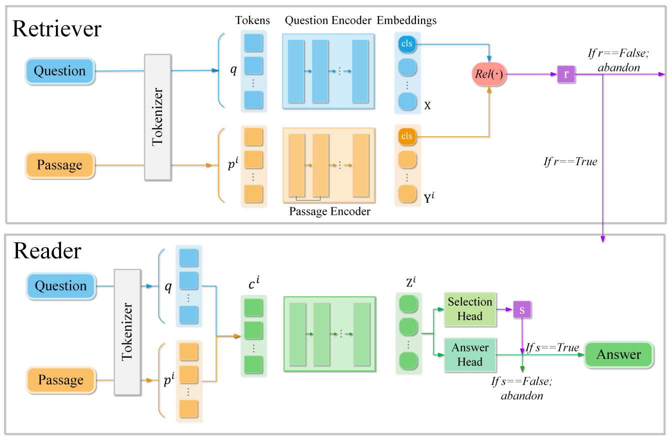


Fig. 2. An Illustration of the OpenQA architecture used in the paper. Given a question q and a candidate passage p^i, the Retriever judges whether p^i is retrieved to enter the Reader according to $Rel(q, p^i)$. The Reader extracts an answer from p^i in Answer Head and judges whether to use it as the final prediction according to the result of Selection Head.

an attention matrix $A_{l,h} = \{a_{i,j}\} \in \mathbb{R}^{N \times N}$, where $a_{i,j}$ represent interactions between pairwise tokens i and j to some extent. With the attention matrix and its variants illustrated in Sec. 2.3, we can analyze the information flow within modules in the OpenQA model.

3.1.2 Model architecture

In this study, we focus on a two-stage OpenQA model based on the "Retriever–Reader" architecture, where the Retriever and the Reader are both BERT-based modules for retrieving relevant passages and extracting answer spans from given passages respectively, as shown in Fig. 2.

Dense Retriever. A typical dense Retriever adopts two independent encoders, Question Encoder $Q_E(\cdot)$ and Passage Encoder $P_E(\cdot)$, to encode question q and i-th passage p^i respectively, into d-dimensional real-valued vectors, $Q_E(q)$ and $P_E(p^i)$. Specifically, similar to the general language model, special tokens ([CLS] and [SEP]) are added to the beginning and end of the question and passage, and another [SEP] is added into the passage to separate its title and body, resulting in two sequences of tokens: $q = [\text{[CLS]}, q_1, q_2, \cdots, q_{n_q-2}, \text{[SEP]}]$ (length n_q) and $p = [\text{[CLS]}, p^i_1, p^i_2, \cdots, \text{[SEP]}, \cdots, p^i_{n_p-2}, \text{[SEP]}]$ (length n_{p^i}). After padding and tokenization, the two sequences are mapped to the high-dimensional space as dense embedding and fed into Retriever. Considering layer l, we have two embeddings: $x^l \in \mathbb{R}^{n_q \times d}$ and $y^l \in \mathbb{R}^{n_{p^i} \times d}$. Then, the Retriever computes the relevance score $Rel(q, p)$ between the pairwise question and passage according to their two final outputs, $Q_E(q)$ and $P_E(p^i)$. A common method is to calculate the inner product of the pooled output of two final embeddings, i.e., X_{CLS} and Y^i_{CLS}. Then, Retriever filters out the top-k most relevant passages and other passages are abandoned.

Extractive Reader. Given one question, the Reader functions as both Re-ranker and Span-extractor[1]. It re-ranks the top-k passages filtered by the Retriever then extracts the answer span. Specifically, the Reader splices k question and answer pairs into k sequences of length l: $c_i = [\text{[CLS]}, q_1, q_2, \cdots, q_{n_q-2}, \text{[SEP]}, p^i_1, p^i_2, \cdots, p^i_{n_p-2}, \text{[SEP]}], 1 \leq i \leq k$. Similar to the encoders in the Retriever, the Reader maps c_i into representation and we denote it as z^l_i in layer l. Then, the model calculates the probabilities of the passage containing the answer as $P_{\text{selected}}(i)$ by the Selection Head. A token that is the starting position and ending position of an answer spans $P_{\text{start},i}(s)$ and $P_{\text{end},i}(t)$, from which a span score of the s-th to t-th tokens from the i-th passage $P_{\text{span},i}(s,t)$ is generated in the Answer Head. Given c_i as the input of the Reader, we denote the sequence output as $\mathbf{Z}_i \in \mathbb{R}^{l \times d}$ and concatenate the pooled outputs of k sequences as $\hat{\mathbf{Z}} = [\mathbf{Z}^1_{\text{CLS}}, \ldots, \mathbf{Z}^k_{\text{CLS}}] \in \mathbb{R}^{d \times k}$. The detailed calculation process can be summarized as:

$$\begin{aligned} P_{\text{start},i}(s) &= \operatorname{softmax}(\mathbf{Z}_i \mathbf{v}_{\text{start}})_s \\ P_{\text{end},i}(t) &= \operatorname{softmax}(\mathbf{Z}_i \mathbf{v}_{\text{end}})_t \\ P_{\text{selected}}(i) &= \operatorname{softmax}\left(\hat{\mathbf{Z}}^\top \mathbf{v}_{\text{selected}}\right)_i \\ P_{\text{span},i}(s,t) &= P_{\text{start},i}(s) \times P_{\text{end},i}(t) \end{aligned}$$

where $\mathbf{v}_{\text{start}}$, $\mathbf{v}_{\text{end}}$ and $\mathbf{v}_{\text{selected}} \in \mathbb{R}^h$ are learnt vectors. Then, the Reader predicts the answer span $(s,t)_i$ of each passage candidate according to $P_{\text{span},i}(s,t)$ and adopts the prediction of passage i with the highest probability $P_{\text{selected}}(i)$ as the final answer. By contrast, the predictions of other top-k passages are abandoned.

1. For convenience, "Reader" is used to denote the module with the task of extracting answers instead of "Span-extractor" in the following.

3.2 Methodology

The methodology of our preliminary study follows three steps: working with domain experts to distill the tasks, summarizing the design requirements needed to support the tasks, and interviewing external experts to justify the tasks and design requirements. We have collaborated with six domain experts (**E1**-**E6**). **E1** is one of the co-authors and **E2**-**E6** are external experts. We have worked closely with **E1** in conducting weekly discussions and performing literature research for around four months. On the basis of existing work and feedback from **E1**, we have refined the tasks and design requirements presented in Secs. 3.3 and 3.4. We have conducted interviews with five external experts (**E2**-**E6**) to confirm the comprehensiveness and rationality of the tasks and design requirements. In addition, all six experts have participated in the case studies (Sec. 6) and evaluations (Sec. 7) after the system was developed.

3.3 Task Analysis

Our primary goals are to help experts semantically understand the decision process of OpenQA models and provide new insights into model enhancement. We worked closely with **E1**, a co-author of this paper, who has four-years of experience in NLP and QA, for about four months to obtain her feedback and iteratively refine the design requirements. Through discussions with her and a literature review, we have come up with the following analytic tasks derived from our goals.

T1: Explain the data flow within a single module. The expert indicates that the ability of the Retriever is the main obstacle that limits the performance of the whole architecture. If no relevant passage containing the *gold answer* is retrieved, the Reader will fail to give correct predictions. Therefore, the expert is eager to understand the internal process of filtering out top-k passages in the Retriever. For example, what is the basis for the Retriever to calculate the relevance score for candidate passages? In addition, considering that the final predicted answer comes from the first ranked passages after re-ranking, understanding the decision process of the Re-ranker is equally important and instructive to improve the filtering ability of the Retriever.

T2: Compare the differences in information processing between multiple modules. Although similar Transformer-based modules are used in the OpenQA model, the internal decision process differs due to the different training objectives and structures. Usually, Reader re-ranks the retrieved top-k passages in a different order from that in the initial results in the Retriever. Moreover, the researcher is concerned about how the learned vectors work in the Reader to complete the two tasks with the same sequence output.

T3: Explore alternatives and options with high prediction scores. The expert is interested in whether the differences between alternative passages (i.e., other top-k passages not ranked first by the Re-ranker) and selected passages are distinguished by the model and whether the final orders are attributed to coincidence. Comprehensive knowledge of the decision flow can be obtained by exploring all the top-k retrieved passages of one instance.

T4: Explore the relationship between the decision process and type of tasks. A recent work [30] has found that the distribution patterns of attention heads may differ depending on the different task types in VQA. Therefore, the expert wants to infer if the decision flow of the OpenQA model is related to the task type. Identifying the association of model decisions with task types provide insights into model enhancement, such as designing task-specific models.

3.4 Design Requirements

On the basis of these main tasks, we have summarized the following design requirements to be supported in our system.

R1: Summarize the features of the dataset and model behavior. The system should provide an overview of the dataset distribution and model performance to help experts get started. The common methods used to describe the dataset without additional labels or descriptive information differentiate subsets with the first few words of questions [3]. The relevant statistics in OpenQA include the top-k Retriever accuracy (the percentage of top-k retrieved passages that contain the answer) and exact match (EM).

R2: Link module responses to task types. To further correlate model decisions with task types (**T4**), we need to provide the interactions between them, such as determining how the various parts of the module respond differently to a particular type of a problem and how a specific part of the module responds differently to different types of questions.

R3: Browse the information flow within the model for an instance. Once experts have selected the question category, they should be able to explore the decision process for a single instance. Therefore, we need to briefly show the information flow in the Retriever and the Reader for a particular question and the corresponding top-k passages to prepare users for exploring the candidates (**T3**).

R4: Layer-level analysis of a single module for each candidate. Given that the Transformer-based module is constructed from successive layers and that each layer of the module learns different semantic knowledge [78], layer-level analysis is required (**T1**). The system needs to support comprehensive inspection of important data such as embedding, attention values, and their variants, and must prevent visual clutter caused by long text.

R5: Provide comparable visualization to help with semantic understanding. The exploration process of the OpenQA model is a continuous comparison process. Users need to compare the performance and module responses of different subsets (**T4**), compare the decision flow of alternative passages in one instance (**T3**), and compare the semantic information learned by different layers within a single module (**T1**) and multiple modules (**T2**).

3.5 Expert Verification

To evaluate our motivation, tasks, and design requirements, we invited five external experts (**E2-E6**) with diverse backgrounds to conduct verification. **E2** and **E3** have two-years of experience in NLP, **E4** has four-years of experience, and **E5** and **E6** have eight-years of experience. All experts are familiar with BERT and Transformer. **E4**, **E5**, and **E6** have published research publications about QA, whereas **E2** and **E3** have not conducted related research. During the background checks, we learned that all experts have experience in understanding model mechanisms through attention visualization, with **E3** and **E6** reporting their experience in using BertViz [81] in their publications to corroborate the results. In addition, multiple experts have experience in using general interpretability techniques, such as SHAP [50] for **E3** and saliency methods for **E5** and **E6**.

We conducted one-on-one interviews with each external expert for about 30-50 minutes, and the interviews included a presentation of existing works, their generalizability and a description of distilled tasks and design requirements. First, we presented four important related techniques, namely, Dodrio [86], LIT [77], VisQA [30], and QADiver [39], and gave a detailed illustration of the challenges faced by existing works in utilizing OpenQA models, such as DPR. The experts were interested in these visualization tools and appreciated our elaboration of the three challenges in providing visual explanations for OpenQA models. We then illustrated the tasks and design requirements, which resonated well with the experts. In particular, **E2** and **E3** were interested in summarizing the decision flow of an instance (**T3**, **R3**) because they had not observed the model's decision-making process from a microscopic perspective before and believed that it is helpful for model understanding and improvement. **E6** wanted to gain insight related to model improvement, so the association between model response and data types (**T4**, **R2**) was of great interest. All external experts had expectations about how our system could meet the abovementioned requirements and solve challenges such as long context and multiple modules. **E4** commented, "*My questions related to text-based OpenQA are all included, and I'm curious about how visualization experts will explore them.*"

Overall, all the experts agreed with our design motivation, distilled tasks, and design requirements. The experts with different backgrounds and experiences emphasized tasks differently, but they all had high expectations for the developed system. Furthermore, we developed an explanation engine (Sec. 4) that supports the tasks and the interface of our system, VEQA (Sec. 5).

4 EXPLANATION ENGINE

Before introducing the interface and interaction design, we discuss the related technologies used in our system.

4.1 Data Preprocessing with Attribution Methods

Unlike some previous studies [16], [30] that only utilized attention maps for reasoning and explanation, we select saliency and attribution approaches to explain the module in terms of evaluating

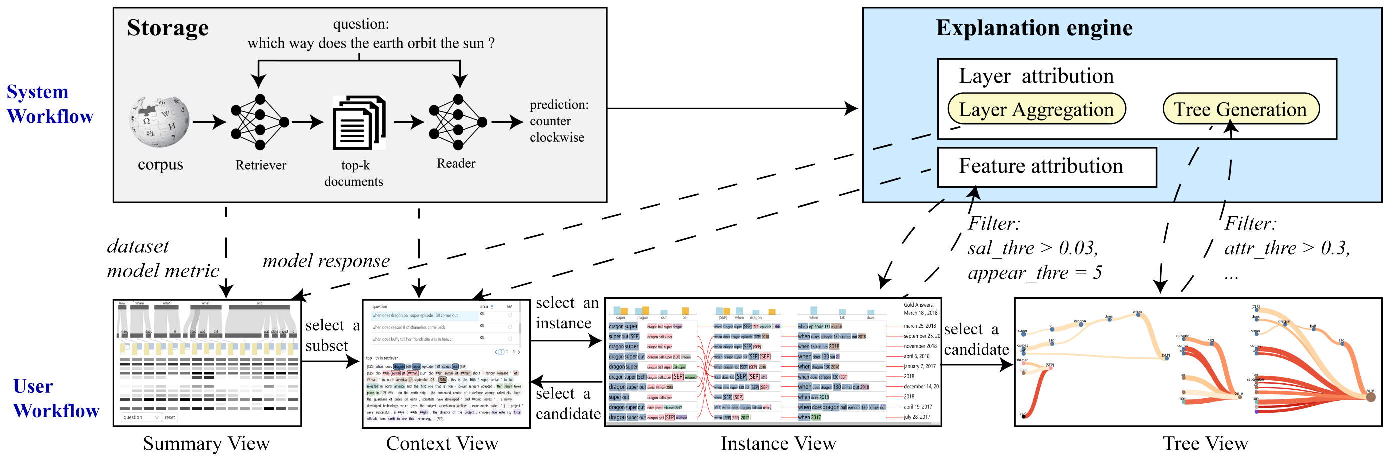


Fig. 3. The Workflow of VEQA. VEQA consists of three parts: Storage, Explanation engine, and Visual analysis interface. The data generated by the meticulously trained and evaluated OpenQA model is stored in Storage and used to provide users with basic information about the model and dataset. The Explanation engine adopts attribution algorithms on module outputs and layer outputs in the Storage and aggregates the layer attribution results into layer responses and dependency trees. The above data is provided in each view of the Visual analysis interface for users' exploration. After selecting a subset of interest in the *Summary View* based on the information retained in the Storage and layer responses provided by the Explanation engine, users further select instances in the *Context View* and gain an overview of data flow at the instance level by adjusting the saliency and occurrence thresholds in the *Instance View*, before finally selecting a candidate passage in the *Tree View* for fine-grained comparison and exploration by adjusting thresholds for edges of attribution trees.

the contribution of each input feature to the module output (**R3**) and the contribution of each layer to the module output (**R4**).

Considering that multiple modules are used in the OpenQA model and that a single module may carry multiple tasks, we use (M,T) to denote the module M that carries task T. As discussed in Sec. 3.1, a complete OpenQA model consists of four (M,T)s that determine the final prediction: two independent encoders in Retriever, i.e., question encoder (Q,E) and passage encoder (P,E), and one module in Reader that acts as Re-ranker (R,R) and Span-extractor (R,S). In addition, we use $F_{(M,T)}$ to denote the final output of (M,T), which serves as the attribution target.

For a given module M with a given task T, each token embedding $\mathbf{e}_i$ in input embedding $\mathbf{e}$ with length L is assigned a saliency (scalar) $\mathrm{Sal}_{(M,T)}(\mathbf{e}_i)$ by Integrated Gradients as follows [76]:

$$\mathrm{Sal}_{(M,T)}(\mathbf{e}_i) = \frac{1}{m}\sum_{k=1}^{m} \nabla_{\mathbf{e}_i} F_{(M,T)}\left(\mathbf{b} + \frac{k}{m}(\mathbf{e}-\mathbf{b})\right)\cdot(\mathbf{e}_i - \mathbf{b}_i) \tag{1}$$

where $\mathbf{b}$ refers to repeated [MASK] vectors as baseline and $m = 50$ refers to the execution of 50 steps in the Riemann approximation of the integral, which is the general default setting and ensures balance between accuracy and speed. Moreover, with the layer conductance methods [18], [69], we obtain the attribution scores of the task-independent output $F_M^l(\mathbf{e})$ of the l-th layer in module M for task T, i.e., $\mathrm{Attr}_{(M,T)}(F_M^l(\mathbf{e}))$, which has the same shape as $F_M^l(\mathbf{e})$.

$$\mathrm{Attr}_{(M,T)}(F_M^l(\mathbf{e})) = \sum_{k=1}^{m} \frac{\partial F_{(M,T)}\left(e^{(k)}\right)}{\partial F_M^l\left(e^{(k)}\right)} \left(F_M^l\left(\mathbf{e}^{(k)}\right) - F_M^l\left(\mathbf{e}^{(k-1)}\right)\right) \tag{2}$$

where $\mathbf{e}^{(k)} = \mathbf{b} + \frac{k}{m}(\mathbf{e}-\mathbf{b})$ and m is set to 50 for the same reason as before. Therefore, the outputs of the l-th layer in module M, i.e., embedding $E_M^l(\mathbf{e})$ and h-th head attention matrix $A_M^{l,h}(\mathbf{e})$, are expressed as $\mathrm{Attr}_{(M,T)}(E_M^l(\mathbf{e})) \in \mathbb{R}^{L\times d}$ and $\mathrm{Attr}_{(M,T)}(A_M^{l,h}(\mathbf{e})) \in \mathbb{R}^{L\times L}$ on the condition of task T. Eq. 1 is the Riemann approximation of the integral of gradients with respect to inputs along the path from a given baseline to the input. Similarly, Eq. 2 is approximated by

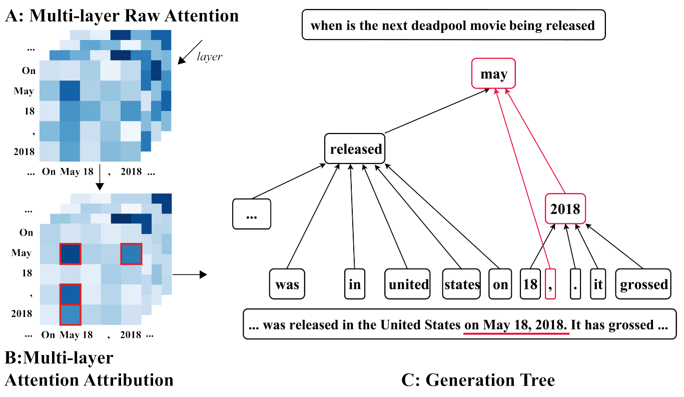

A: Multi-layer Raw Attention
layer
... On May 18 , 2018 ...
B:Multi-layer Attention Attribution
when is the next deadpool movie being released
may
released
2018
...
was
in
united
states
on
18
,
.
it
grossed
... was released in the United States on May 18, 2018. It has grossed ...
C: Generation Tree

Fig. 4. An illustration for the construction of a generation tree from an attention matrix. Multi-layer raw attention (A) is refined into attention attribution (B) by Eq. 2, and then words and word pairs with high attribution scores are selected as nodes and edges from top to bottom to build the tree (C). For instance, "may", with the highest sum of attribution scores in the top layer, is selected as the root node, and "," and "2018" are selected as the child nodes due to their high attribution scores with "may". Due to space constraints, the figure shows the main part of the tree.

the gradient integral flow of neurons in the layer. Interested readers may refer to the original literature [18], [69], [76].

4.2 Exploring Layer-level Information Flow

Hierarchical representation of text is familiar to NLP experts. Many studies [12], [68] have integrated hierarchical structures of natural language into DNNs for better representations, and others [19], [34] learned syntactic parsers for improved parsing accuracy. Additionally, using hierarchical representations to analyze DNN has become an area of interest in the NLP community. Zhang et al. [93] constructed a tree to encode salient interactions extracted by DNN, on the basis of Shapley values of words [50]. Considering that displaying the original attribution matrix in the case of long text causes visual clutter and that most elements of the attribution matrix are minimal (close to zero), we adopt the tree generation algorithm proposed by Hao et al. [25] to display the information flow inside the module (**R4**). The tree generation is based on the attention attribution derived in Sec. 4.1, as shown in Fig. 4.

First, we summarize the attribution scores of each attention head within the l-th layer with L2-norm as $\text{Attr}_{(M,T)}(A_M^l) = \{\hat{a}_{i,j}^l\}_{L\times L}$. Second, we take the L2-norm of the attribution scores of the embedding output at the l-th layer as $\text{Attr}_{(M,T)}(E_M^l) = \{e_i^l\}_L$, which is a measure of how salient the l-th layer of module (M,T) is to input sequence t that corresponds to embedding $\mathbf{e}$. Third, we select the token i with the highest token-wise attribution score e_i^l as the root node of the tree, and use a heuristic top-down greedy algorithm to traverse all nodes that are not in the tree from the l-th layer to the 1-th layer. Lastly, we select tokens i and j with pairwise attribution scores $\hat{a}_{i,j}$ greater than a certain threshold to join the tree. Most of the algorithm details are similar to those in the original paper [25], except for Reader, where we remove the special treatment for the last layer of the module, that is specific to classification tasks.

4.3 Interpreting Functions of Layers

To relate module behavior to task type (**R2**), we calculated the importance of the l-th layer of a given module (M,T) as $LI_{M,T}^l$ as the average of the maximum attribution $a_{i,j}$ in $\text{Attr}_{(M,T)}(A_M^l)$ for all instances in the dataset. Other head statistics, such as the $k-$*number* used in VisQA [30] to represent the distribution of attention heads, are not considered, because only a few strong interactions exist between tokens in each attribution matrix and the distribution within each head is similar to that in others. Considering that the OpenQA model consists of multiple modules, we limit the finest granularity of exploration to the layer level. Thus, we do not calculate the importance of each attention head separately.

5 Visual Analytics System

In this section, we introduce the workflow of VEQA and discuss the visual design and interaction with the user interface in detail.

5.1 Workflow

Through iterative design and frequent meetings with OpenQA experts, we establish the analysis workflow based on the generated explanation and user interface of VEQA, as indicated in Fig. 3.

In general, experts expect the analysis process to proceed from global to local (i.e., dataset $\rightarrow$ subset $\rightarrow$ instance $\rightarrow$ candidates in an individual case). They wish to explore the difference in the response of an individual module to different task types and the difference in the response of different modules to the given task type (**R1**, **R2**), which can be obtained by checking the statistics and importance of each part of the module in the *Summary View*. Then, the experts select an instance from the subsets of interest in the *Context View* to analyze and see the decision process within the whole model (**R3**, **R5**) in the *Instance View* with the complementation of the *Context View*. The experts then continue to explore the difference in the way that given question-passage pairs are processed in different stages (**R5**) with attribution tree comparison in the *Tree View*. The *Tree View* also allows them to further explore the variation in information flow layer by layer in a single module (**R4**).

5.2 User Interface

The interface includes the *Summary View*, *Context View*, *Instance View*, and *Tree View* to support the visual exploration of OpenQA (Fig. 1).

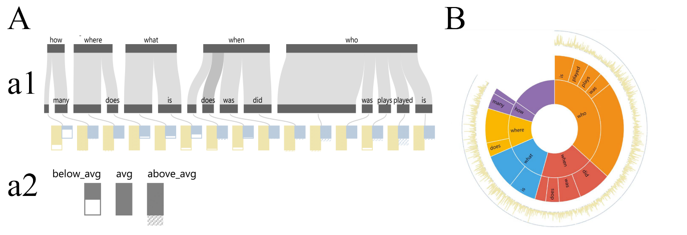


Fig. 5. Design alternatives for the *Summary View*. A: Our current design (a1) of augmented sankey diagram with grouped barchart and legends for barchart (a2). B: An augmented sunburst diagram.

5.2.1 *Summary View*

The *Summary View* (Fig. 1A) provides an overview of the response of the module at each stage under different categories of subsets and a visualization of the evaluation metrics (**R1**, **R2**). It helps experts understand the distribution of the dataset and the differences in the response of modules to subsets, thus guiding users in selecting specific subsets for further exploration.

Visual design. We follow the experts' suggestion to divide the dataset into multiple subsets based on the first two terms and visualize them through multiple Sankey diagrams with a tree-like layout. The width of each node is proportional to the number of questions it represents. Each subset is connected by a curve to two bars that represent the corresponding performance metrics in Retriever and Reader, i.e., *top-k accuracy* and *EM*, as shown in Fig. 5A. To clearly show the performance difference of subsets, we adopt a kind of bar chart design that is similar to the one in VBridge [9]. We calculate the average performance metric for the entire dataset as the standard value to delineate the bars with a constant height and use the hollow part of the bars to indicate performance below the mean. Meanwhile, shading is used for the part above the mean, as the legend shows in Fig. 5a2. To facilitate the exploration of module responses in relation to task type, we place rectangles, the number of which encodes the number of layers of each module, below each bar chart to encode the mean values of layer responses in the corresponding subset with gradient colors, which can be obtained by the layer aggregation method described in Sec. 4.3.

Design alternatives. Fig. 5B shows another option that we considered during the design process. Sunburst charts are a common form of visualization for representing QA datasets in terms of word divisions [3]. In such charts, words radiate outward from the inner circle, and the angle of the arc is proportional to the frequency of each word. In addition, radial linecharts are drawn outside Sunburst charts to indicate two performance metrics for each instance. However, compared with the Sankey diagram with a tree layout, although a Sunburst chart can clearly represent the data distribution, its radial layout makes it difficult for users to visually compare the differences between various problems in the model decision.

5.2.2 *Context View*

As a complement to the *Summary View*, the *Context View* (Fig. 1B) lists all question instances in the dataset or selected subset and the predicted results for each instance in a table and presents a heatmap of the selected passage at a certain stage.

Visual design. The question table in the *Context View* has three columns that present information about the instance and prediction. The first column records the question text, and the second column shows the *top-k accuracy* of the instance at the Retrieval stage by using a bar embedded in the table. The circles in the third column indicate the overall evaluation metrics, i.e., *EM*. The hollow circles indicate that the final result does not exactly match *gold answers*, and the solid circles indicate the opposite. The original text of a selected passage and its saliency score distribution at a selected stage are shown below the question table, whose content is controlled by the interaction with the *Instance View*.

5.2.3 Instance View

The *Instance View* (Fig. 1C) shows the overall data flow in the OpenQA model for the top-k passages retrieved from the corpus with a given question in the form of a ranking visualization combining text and bar charts (**R3**, **R5**).

Visual design. The *Instance View* consists of four columns representing the summary of important words in three tasks, i.e., Retriever, Re-ranker, and Reader, and the final prediction results. Inspired by LineUp [23] and Parallel Tag Clouds [14], we summarize the individual alternative passage being ranked to a set of words $V = t_i$ with saliency score $Sal_{M,T}(\mathbf{e}_i)$ above a threshold at a certain stage (M,T) and place all the candidates vertically in the ranking order of the current stage. The sizes of the presented words are proportional to their saliency score. Moreover, preliminary experiments have shown that the module usually focuses on important words within one or two sentences in long texts. Thus we use colors to encode word positions, i.e., we use the same color for words belonging to the same sentence, which brings interesting insights that will be discussed in Sec. 6.1.

To further abstract the decision process of the model from the top-k alternatives, we place barcharts above the first three columns indicating the context-independent words with occurrences above a certain threshold within that column and use two colors to distinguish their positions. For example, words that appear in both the question and the passages are encoded with side-by-side blue and yellow rectangles and placed on the left, whereas in the middle and on the right are blue bars representing words that appear multiple times only in the question and yellow bars representing words that appear multiple times only in the passages, respectively. Users can hover over the bars to explore the corresponding word occurrences in all three columns.

Red or green lines are used between the two columns to characterize the input and output of each stage. Specifically, the lines between the first and second columns present the results before and after the re-ranking, with red indicating that the candidate does not contain *gold answers* and green indicating the opposite. The lines between the second and third columns are horizontal, and the colors are similar to those before because Reader does not change the order of candidates. The horizontal lines between the third and fourth columns point to the final prediction of a single candidate, and their color indicates whether the final result is an exact match to the *gold answers*.

5.2.4 Tree View

On the basis of the generated attribution tree (Sec. 4.3) that abstracts a large number of connections within the complex module into a semantic tree structure, the *Tree View* allows users to further explore the processing flow of the pairwise question and passage throughout the whole model or in a certain module at a fine-grained level via comparable tree visualization (**R4**, **R5**).

Visual design. We use color and size, respectively, to encode the information embedded in the attribution tree. To effectively

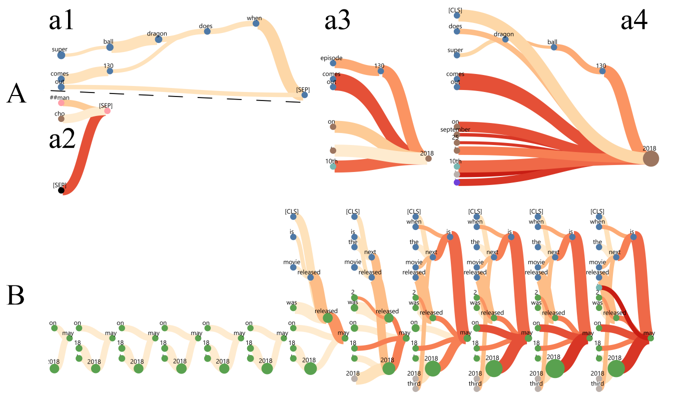


Fig. 6. Two modalities for the *Tree View*. A: Two trees for two encoders in Retriever placed vertically as a whole (a1, a2) and placed side by side with the other two trees for Re-ranker and Reader (a3, a4). B: Evolution of the attribution tree for 12 layers of one certain module.

perceive how the tree is constructed, we adopt the same color schemes used in the *Instance View* to encode the positions of tokens in the context and another gradient color that does not create the confusion with the former to encode the layer l from which the edge originates, as shown in the legend of Fig. 6. The width of the link between two nodes presents the pairwise attribution score $a^l_{i,j}$ for word i and j in layer l, and the diameter of the node encodes the token-wise attribution score e^l_k of corresponding word k in layer l. No direct connection exists between e^l_k embedding attribution and $a^l_{i,j}$ in attention attribution, as discussed in Sec. 4.1. However, this visualization approach can qualitatively gain insights into both methods, which will be further illustrated in the Evaluation (Sec. 6.1) and Discussion (Sec. 8) sections.

To increase information density and prevent visual clutter, we only displayed the words contained in the attribution tree instead of all the tokens because the number of the latter is usually more than 100, whereas the total number of words that appear in all trees is much smaller when the threshold of the algorithm is set normally. Meanwhile, we aim to ensure comparability between trees. Therefore, we determine the union of words that appeared in the attribution tree of each module and then equally spaced and aligned them. We encode the width of each tree based on its height to facilitate a comparison of hierarchical changes.

The *Tree View* supports attribution tree comparisons at each stage and layer-by-layer evolution in a single module with adjustable and independent thresholds for each module. When the view shows the former, the four trees corresponding to the four groups of (M,T)s are horizontally placed as three wholes, as shown in Fig. 6A, where two attribution trees corresponding to (Q,E) and (P,E) are vertically placed on the left side to align with the (R,R) and (R,S) trees. When the user switches the view to layer-level exploration, l attribution trees corresponding to the number of module layers are arranged horizontally (Fig. 6B), and only the threshold of the tree for that module can be changed by users.

Design alternatives. In the *Tree view*, the properties of each tree's nodes and edges are individually encoded by size and color maps. As alternatives, we have also considered other forms of hierarchical visualization [67] or graph visualization [26], such as treemap, packing, and icicle, in addition to the node-link tree. However, we need to encode the edges of trees to show the interactions between words, not just hierarchical relationships.

These implicit hierarchy visualizations are inapplicable, and we excluded them. After limiting the design space to the explicit node-linked tree visualization, we've considered various tree layouts. First, we've ruled out radial layouts or force-directed trees because they make it difficult to identify the contextual structure of a passage. The two remaining options are a contextually ordered horizontal layout or a vertical layout tree. Compared with the vertical layout, the horizontal layout can better display the hierarchical semantic structure, and the horizontal word order is more in line with how people read. However, the experimental results show that, in general, the height of the attribution tree is small, and the number of leaf nodes is large. Therefore, when multiple horizontal layout trees are placed side by side, the screen space cannot be effectively utilized, and the difference in tree structure cannot be intuitively displayed, which are critical shortcomings in terms of design requirement (**R5**).

Color map design. In addition, we carefully considered the use of color encoding. After repeated discussions with experts, the sources (*layer*) of the edges and the locations (*sentence*) of the nodes are determined to be the key attributes that need to be visualized. According to color theories and classification of data types proposed by Silva et al. [70], the *layer* attributes with relative order belong to ordinal data and should be encoded with sequential color scales. Given that the relative order of node positions is unimportant, it should be treated as nominal data encoded with qualitative color scales. Encoding a *sentence* in a similar way as that for a *layer* makes the user implicitly order the node positions, which does not make sense for understanding the model to identify the clustering features of important words. It may mislead users into thinking that these colors represent feature importance, because coloring the background of words by importance is a particularly familiar visualization for NLP experts. To alleviate the confusion caused by multiple color maps, we choose a color scheme that was as non-conflicting as possible, and applied the *sentence* colormap uniformly to the *Context View*, *Instance View* and *Tree View*. In addition, we provide clear legends in the *user panel* to provide users a clear understanding of the colormaps used in the system.

5.2.5 User Interaction

VEQA provides a rich set of smooth interactions, thus allowing users to perform multi-level exploration between different views.

Hovering. Users can hover over all elements that encode numeric data with a color or size to obtain precise information. In addition, hovering over a certain bar of the barcharts in the *Instance View* highlights all the same words in the table below for users to discover the differences in the distribution of focused words at each stage. Similarly, when users hover over a node of a tree in the *Tree View*, all the nodes in other trees representing the same contextualized word are highlighted.

Clicking. Many interactions among views in VEQA can be triggered by clicking. For example, users click on a link of the sankey diagram in the *Summary View* to update the question table in the *Context View*, which shows the question lists of the selected subset. In addition, clicking on a row of the table in the *Instance View* updates the original text and heatmap in the *Context View*.

Filtering. To enhance the perceived scope of information, we use sliders to filter the data where thresholds are involved, as in the *Instance View* and *Tree View*. Notably, the categories of questions presented in the *Summary View* also rely on threshold constraints, but the experts believe that the system can frame the categories given appropriate thresholds and that threshold adjustment here is unnecessary.

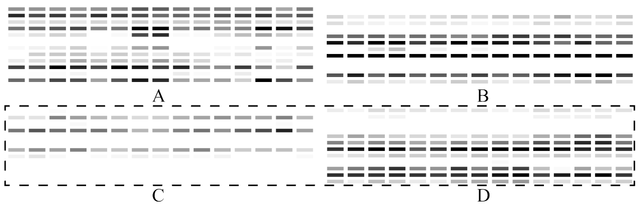


Fig. 7. Different layer response of four modules. A to D denote responses of two encoders in the Retriever, the Re-ranker and the Reader, respectively.

Guiding. As described in Sec. 5.1, the workflow of VEQA is mainly one way: from the high level to the low level. To help users quickly master the workflow and reduce their burden, the *Summary View* and *Context View* are only displayed when the user first logs into the interface. When the user selects an instance of interest to explore, the *Instance View* is unlocked. Similarly, the *Tree View* is unlocked after the user selects a candidate.

6 CASE STUDY

We conducted case studies with the six aforementioned NLP experts (**E1**-**E6**). We demonstrateed how VEQA explains the data flow in the DPR [32] architecture with the Natural Question (NQ) dataset [37], which contains 300,000 naturally occurring questions along with human-annotated answers from Wikipedia pages. Given the limitation in computational resources, we used two officially provided model checkpoints [2] trained on NQ via different training schemes. The first checkpoint was trained on negative samples mined by BM25 with high similarity to the question but do not contain answers as well as random and positive samples for other instances. The second checkpoint was trained on the data comprising previous samples and hard negative samples mined by the first checkpoint. We chose the same divided test set as that in the previous work, including 3610 instances with 100 retrieved passages. We evaluated the top-20 retrieved passages with attribution algorithms. The whole evaluation process consisted of three parts. First, **E1** followed our guide to observe each view freely and established preliminary conclusions about the decision process. Second, **E4** followed the general workflow to explore the model and find successful decision cases. Third, **E6** tried to use our system to explore how high-quality training data can help boost retriever performance.

6.1 Part I: Abstracting the decision process

In the first part, **E1** explored each of the three main views, except for the context view that serves as a complement, to obtain preliminary conclusions. After the initial exploration, **E1** stated that our system successfully abstracts and visualizes the decision flow of DPR.

6.1.1 Global Summary

After selecting DPR as the evaluation architecture and the first checkpoint and the test dataset in the *User Panel*, **E1** observed the distribution of the dataset, the performance metrics of the subsets, and the responses of multiple modules in the *Summary View* (**T4**).

2. https://github.com/facebookresearch/DPR

Evaluation metrics. As shown in Fig. 1B, the performance scores corresponding to each subset classified by the opening two words are unequal, especially the *top-k accuracy* used to measure the performance of the Retriever. The overall evaluation metric for the whole model, i.e., *EM*, differs among subsets. When the *top-k accuracy* is way below average, the model fails to give correct predictions without obtaining passages containing *gold answers*, such as the subset classified by "How-". Hence, **E1** claimed that future model design should consider different question types.

Module response. **E1** then clicked the button to observe the layer aggregation results for four (M,T)s as shown in Fig. 7. Each (M,T) had specific layers in it that play the main role to complete the task, especially (Q,E) and (P,E) in the Retriever that had a large number of layers with a relatively small effect, indicating the potential for optimizing the model representation space in terms of training methods, dataset selection and model compression. In addition, since (R,R) and (R,S) shared the same module, comparing Fig. 7C with Fig. 7D revealed that the Reader functioned as a Re-ranker in the low layers while focusing on extracting answers in the middle and high layers.

Potential relation. Meanwhile, **E1** found inconsistent layer responses corresponding to different subsets. That is, the color shades of different rectangles on the same row were inconsistent, presumably due to question types, which further explored in Sec. 6.2.

6.1.2 Instance exploration

After **E1** randomly selected multiple instances and observed the data flow changes in the corresponding top 10 candidate passages in the three tasks in the *Instance View*, some preliminary conclusions about multiple modules were drawn (**T2**).

Retriever and Re-ranker. A remarkable feature was that the first column indicating the Retriever was full of words with blue and pink backgrounds corresponding to sentence 0 (question) and sentence 1 (title), respectively. Therefore, **E1** believed that most of the words in question had a significant impact on the results of (Q,E) and that the embedding output of (P,E) was heavily dependent on the title. Meanwhile, by viewing the words presented in the grouped bar chart above the column, **E1** found that the Retriever relied on overlapped and relevant words. In the example shown in Fig. 1C, the words "dragon" and "super" appeared multiple times in the question and the selected title, whereas other (secondary) words, such as "episode" and "113", were ignored, which might be the reason for the failure of the Retrieval that all lines connecting the rows were in red. However, the Re-ranker seemed to focus on the overall structure and semantics when reordering, with a high frequency of separate-sentence words such as "[SEP]" and periods, as well as words outside the main entities (e.g., "episode" shown on the first row of the second column in Fig. 1C). **E1** argued that this confirmed the conclusion drawn by previous work [94] that interaction-based Retriever can capture more semantic information than representation-based Retriever such as DPR.

Re-ranker and Reader. As **E1** looked at the second and third columns representing the Re-ranker and the Reader respectively, she declared that the Re-ranker had begun to capture information about answers because some of the important words concluded from it were similar to those from the Reader and relevant to the final answer. Compared with the Retriever or the Re-ranker, she further found that the Reader usually paid much attention to the first word, which specified the type of question and other words that were important for answering the question, such as "when" and "130" presented in the grouped bar chart above the third column in Fig. 1C, rather than named entities and delimiters.

6.1.3 Candidate Exploration

Next, **E1** tried to compare attribution trees at different scales by adjusting the respective thresholds in the *Tree View*, verified the findings mentioned above because layer attribution generally yielded similar results as feature attribution did, and proposed new ideas (**T1**, **T2**).

Verification of previous conclusions. First, as shown in Fig. 1D, **E1** found that the semantic connections in (Q,E) were close and that (P,E) depended on the title as well as the "[SEP]", which was consistent with the findings in the *Instance View*. Second, she found that the edges in the tree for (R,R) originated from the lower and middle layers, which also appeared in the tree for (R,S). By contrast, the latter was different because it included edges related to the answers originating from the higher layers, which was consistent with the conclusion for the *Summary View*.

Questions about attribution trees. In the process of exploration, **E1** found that the results of the attribution of features from the whole were roughly similar to those of layer attribution together with differences. For example, the words filtered out in the *Instance View* did not appear in the attribution tree. Additionally, the results of hidden embedding attribution were different from those of attention attribution because the node with the largest diameter sometimes did not appear at the top. This result is related to the nature of the attribution and tree generation algorithm, which will be examined in detail in the Discussion section (Sec. 8).

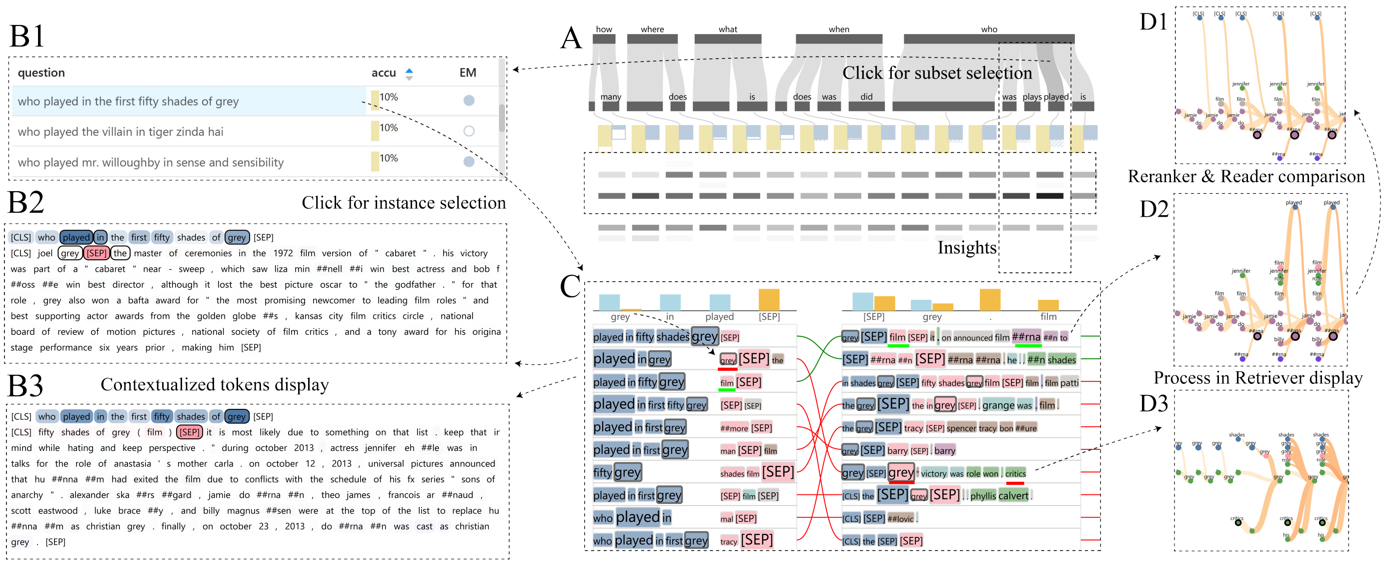
6.1.4 Summary

After the initial exploration of each view, **E1** used VEQA to successfully abstract the decision flow of DPR: the Retriever relied on overlapped words for coarse-grained exploration, and the Re-ranker corrected the ranking results based on the overall structure and prediction results mainly at the low layers. On such basis, the Reader deepened the connection between the prediction and the question at the high layers and extracted the exact answer. Accordingly, **E1** gave potential model enhancement solutions, such as using the Interaction-based Retriever to enhance the retrieval capability together with model compression to improve efficiency, considering questions types when designing models, and adopting better training methods.

6.2 Part II: Categorizing a successful case of decisions

On the basis of the insight obtained by **E1** in Part I, **E4** began a deeper study of individual instances with the general workflow described in Sec. 5.1 to confirm whether the decisions are reliable.

How does the *Summary View* guide the selection of instances? As mentioned in Sec. 6.1.1, **E4** found a potential relation between module response and performance for the different subsets in the *Summary View*. Specifically, by looking at the grouped barcharts in the *Summary View*, **E4** found that "who played" and "who plays" were the two groups with relatively poor *top-k accuracy* but good *EM*, and the corresponding response of the fourth layer of the Re-ranker was much higher than that of the others (Fig. 8A). **E4** speculated that the Re-ranker did a good job of sorting this type of questions, so that the candidate with a *gold answer* was reordered to the top. Therefore, **E4** clicked the link in the Sankey diagram and jumped to the instance table in the *Context View* to select instances in this subset to verify his hypothesis. (**T4**)


B1
A
Click for subset selection
Insights
D1
Reranker & Reader comparison
D2
B2
Click for instance selection
C
Process in Retriever display
D3
B3
Contextualized tokens display
how
where
what
when
who
many
does
is
was
did
plays
played

question	accu	EM
who played in the first fifty shades of grey	10%	●
who played the villain in tiger zinda hai	10%	○
who played mr. willoughby in sense and sensibility	10%	●

[CLS] who played in the first fifty shades of grey [SEP]
[CLS] joel grey [SEP] the master of ceremonies in the 1972 film version of " cabaret " . his victory was part of a " cabaret " near - sweep , which saw liza min ##nell ##i win best actress and bob f ##oss ##e win best director , although it lost the best picture oscar to " the godfather . " for that role , grey also won a bafta award for " the most promising newcomer to leading film roles " and best supporting actor awards from the golden globe ##s , kansas city film critics circle , national board of review of motion pictures , national society of film critics , and a tony award for his origina stage performance six years prior , making him [SEP]

[CLS] who played in the first fifty shades of grey [SEP]
[CLS] fifty shades of grey (film) [SEP] it is most likely due to something on that list . keep that ir mind while hating and keep perspective . " during october 2013 , actress jennifer eh ##le was in talks for the role of anastasia ' s mother carla . on october 12 , 2013 , universal pictures announced that hu ##nna ##m had exited the film due to conflicts with the schedule of his fx series " sons of anarchy " . alexander ska ##rs ##gard , jamie do ##rna ##n , theo james , francois ar ##naud , scott eastwood , luke brace ##y , and billy magnus ##sen were at the top of the list to replace hu ##nna ##m as christian grey . finally , on october 23 , 2013 , do ##rna ##n was cast as christian grey . [SEP]

Fig. 8. An example of categorizing a successful case of a decision by general user workflow. **E4** selected an instance in the *Context View* (B1) based on the insights about the special response of the Re-ranker and performance of the "what-play" subset in the *Summary View* (A). Candidate 2 and Candidate 3 in retrieval results were noticed by **E4** for their ranking variations and special distribution of important words in the *Instance View* (C) such as "grey", "critics", "film" and "#rma", followed by further exploration in the *Tree View* (D1, D2, D3), with the aid of the *Context View* (B2, B3).

Does the Retriever in DPR only rely on keyword matching? **E4** clicked on the row where "who played in the first fifty shades of grey" was located because the *top-k accuracy* of this instance was only 0.1, but model still predicted the correct answer (Fig. 8B1). In the *Instance View* (Fig. 8C), **E4** found that one candidate containing the *gold answer* moved from the third to the first place after re-ranking and the candidate in the second place after retrieval dropped to seventh. Then, **E4** looked at the barchart above the first column and found that "grey" was considered an important indicator by (Q, E) and was a part of the title of candidate 2, so **E4** suspected that candidate 2 was ranked high because of "grey". Furthermore, **E4** dived into the *Context View* to check and found that it was an irrelevant passage (Fig. 8B2). However, although the title of candidate 3 was "fifty shades of grey (film)", which overlapped greatly with the question, the highest scoring word was "film", except for [SEP] (Fig. 8B3). **E4** argued that Retriever noticed the "played" in the given question, that is, that the target of the passages to be searched for was a movie rather than a novel, which illustrated the superiority of dense Retriever over traditional methods that rely on keyword matching (**T3**).

What happens in the fourth layer of the Re-ranker? **E4** wanted to understand how Re-ranker found the difference between two candidates and modified their order, and whether it was related to the strong response of the fourth layer of Re-ranker presented in the *Summary View*. Looking at the Re-ranker tree for candidate 2 in the *Tree View* (Fig. 8D1), he found that the most obvious change in the fourth layer was the addition of all the "grey"s in the passage and the "critics" for the identity into the tree structure. Consistently, the two words above appeared in the second column in the *Instance View*. **E4** believed that Re-ranker confirmed here that candidate 2 was irrelevant. Meanwhile, the most significant change in the fourth layer of the Re-ranker tree for candidate 3 was the addition of another small part of the answer "#rma" and the "film" in the title into the tree structure (Fig. 8D2). **E4** believed that Re-ranker had already determined the possible answers here and used this as a basis for re-ranking. Additionally, after comparing it with the Reader tree at the layer level (Fig. 8D3), **E4** found that the structures and changes of the two trees for candidate 3 were similar, i.e., Reader correlates the possible answers and questions at high layers to determine the prediction (**T1**, **T3**).

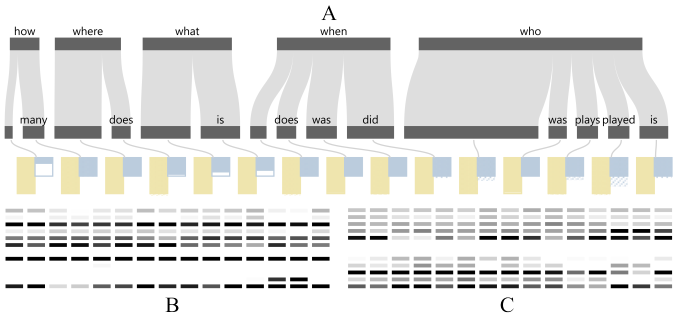


Fig. 9. An overview of performances (A) and attributions of the Question Encoder (B) and the Passage Encoder (C) in the new Retriever.

6.3 Part III: Exploring retrieval performance boost given by the new training scheme

E6 noted that by improving the training scheme, the retrieval performance of the second checkpoint, called DPR(adv_hn), was greatly improved. Specifically, the *top-20 accuracy* on the NQ test set increased from 0.801 to 0.813. Therefore, **E6** was interested in using VEQA to explore the reasons for the improvement and then imported DPR(adv_hn) into VEQA for a comparison.

In the *Summary View* (Fig. 9), **E6** noticed that compared with the original checkpoint (Fig. 1B), the retrieval performance of each subset became more balanced and close to the overall average. A comparison with Fig. 7A and Fig. 7B showed that attribution scores of Question Encoder and Passage Encoder in the middle and high layers generally increased, whereas the attribution of the low layers weakened. **E6** believed that the quality of the negative samples originally constructed for training was not good enough and DPR could only complete relatively easy tasks; the improvement in the

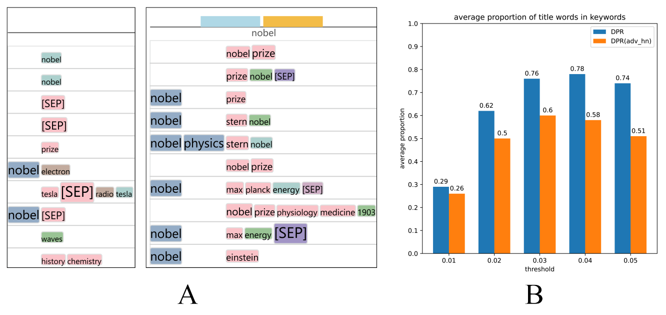


Fig. 10. Comparison of keyword positions. A) An example in which DPR(adv_hn) successfully extracts relevant paragraphs but DPR fails; The left shows the keywords concerned by DPR(adv_hn), and the right shows the keywords concerned by DPR (saliency above 0.03). B) The proportion of title words in keywords whose saliency values are higher than certain thresholds.

training method made each category of problems reach the upper limit of this architecture in the experimental sense by activating numerous high layers (**T4**).

Furthermore, **E6** selected 10 instances in the *Context View*. Here DPR(adv_hn) was able to retrieve relevant paragraphs of them, but DPR performed much worse. **E6** examined these instances in the *Instance View* to determine why DPR(adv_hn) succeed. A surprising finding was that the position diversity of the keywords that DPR(adv_hn) focused on was considerably improved, as illustrated by one instance shown in Fig. 10A. That is, DPR(adv_hn) paid much attention to the global information and no longer completely relied on the matching words in the title for retrieval. To test this hypothesis, we counted the positions of keywords captured by Passage Encoder, whose saliency was higher than the threshold. As indicated by the quantitative results in Fig. 10B, the new checkpoint reduced the focus on the title words. In particular, about half of the most concerned keywords (saliency above 0.05) were words in the body of the passages. **E6** believed that this is a good confirmation of the classical shortcut, which is also a consensus of existing research: current models rely on titles for retrieval because the source of the training set is Wikipedia and titles are a good summary of passages. DPR aims to strengthen the model's attention to global information through contrastive learning by constructing negative samples [24], [64]. However, using random passages or positive samples of other questions as negative samples is not good enough. Given their low correlation with questions, the model easily judges directly by titles. The negative samples used by the new model contain many noise samples with high similarity, forcing the model to pay more attention to the global information than to the title. Notably, this empirical consensus has not been supported by corresponding interpretable data before, and we are the first to observe it from a visual perspective and confirm it statistically. **E6** further mentioned that improved ways of constructing negative samples or limiting the attention to titles during model training can be established to increase model performance.

7 Expert Feedback

After the exploration, we conducted one-on-one interviews with each expert for 30-40 minutes and collected feedback from the aforementioned six experts (**E1**-**E6**), who were generally positive about the usability of VEQA and provided insightful suggestions.

7.1 Usability

Effectiveness. The entire workflow of VEQA was considered reasonable by all the six experts since the straightforward and level-progressive exploration is consistent with their understanding. They found interesting insights and confirmed the previous consensus as results (Sec. 6). Our explanation engine specifically gained their appreciation. **E6** stated that, "*Raw attention is often messy. Attribution visualization helps to visualize the important parts of words and model and increases my confidence in the system's interpretation*." As for the visual interface, all the experts claimed that the visualizations met the requirements and helped them understand the decision process of OpenQA models. The *Summary View* received unanimous praise because it provides an overall analysis of the dataset in terms of different question types and gives global statistical information. The *Instance View* and the *Tree View* were found to be useful by **E1** and **E4**, especially their ability to set thresholds to filter information. **E1** indicated that, *"I can flexibly interact with the system to decide which information to analyze, and set the thresholds to control the amount of information to show"*. In addition, the experts believed that the *Tree View* provides a new perspective of the attention map and an intuitive understanding of the decision process for the instance of interest. Moreover, all experts confirmed the necessity of the two colormaps used in the *Tree View*. Considering the colormaps and learning curve, **E3** said, *"I first focus on the structural changes of the tree, and I understand the view relatively easily. The slight complexity brought by the two colormaps doesn't significantly affect my understanding or the learning curve because there is no reason to think that the color of the nodes encodes the same information as the color of the edges."*

Suggested improvements. Although all experts stated that all the views in VEQA were necessary and provided sufficient information, they reported that VEQA is relatively complex because it consists of multi-level information and also provides an explanation of various modules. They spent 20-50 minutes on average to become familiar with the system, which could be partially improved by adding other indicator words for each sub-figure in each view or even direct text prompts for simple and quantifiable conclusions to help reduce the burden of understanding the various encodings and mappings. Moreover, when **E6** performed the model comparison described in Sec. 6.3, he stopped at the instance level because continuing to explore the *Tree View* requires repeatedly switching models and clicking instances. Therefore, a comparison module to compare different models over the same instance is a viable future capability.

7.2 Inspiration for Future Work

Experts' preference for all-level statistical analysis. VEQA provides visual explanations from the subset, instance, and candidate levels. The views corresponding to the first two levels provide rich statistical analysis and the candidate-level *Tree View* focuses on intuitive insights into the generation trees' evolution. During the interviews, **E6** raised further expectations from the design of our detailed view and said, *"the Tree View already serves the intended purpose well enough as a detailed view of a single candidate, but further statistical analysis could also be performed at the candidate level to obtain more detailed conclusions about the overall behavior of the model*." To this end, further graph-based statistical analysis of the structure of generated trees is promising future work. This expert's expectation leads to the discussion of quantitative analysis versus visual insight in Sec. 9.

Experts' desire for integrated plug-ins. During the interviews, all experts were impressed by the effectiveness of our system, and three of them were interested in the future development of VEQA. **E4** said, "*My brief exploration of the system convinced me of its validity. If it could be integrated into my development environment as a library for easy invocation, I would use it to perform idea evaluation when starting a new project. When finished, it will offer great help for case studies, intuitive analysis, and improving the quality of my articles.*" We have carefully considered this proposal and discussed the possible difficulties and approaches in Sec. 9.

8 Discussion

Here, we discuss VEQA's necessity and novelty, generalizability, scalability, reliability, learning curve and forgetting curve.

Necessity and Novelty. Although OpenQA is a crucial task, the decision-making process of the corresponding model is unclear, which hinders further improvement and understandability. Long texts, multi-modules, and the gap from parameters to semantics are critical challenges in interpreting complex models. After the careful review of general XAI systems, Transformer-specific interpretability methods, and visual analytic systems in Sec. 2, we argue that existing systems and technologies cannot support the in-depth exploration of multi-module models for OpenQA in the context of massive, long texts. Motivated by these conditions, VEQA innovates in the exploration of the decision flow of multi-module OpenQA models and integration of interpretability methods for Transformers and QA, and it offers a new visual design for tighter integration with algorithms, resulting in highly effective data presentation and feature understanding, which are necessary according to the latest surveys [4]. We create a new visual representation to deal with the large-scale data of the model through tree visualizations and flow diagrams, and address the visual confusion and scalability problems caused by attention matrices of long contexts and multiple candidate passages. In addition, on the basis of two novel visualization designs and our explanation engine, we extract keywords and key interactions between words by integrating users' factors into the generation tree algorithms and the interface, thus bridging the gap from saliency and attribution to semantic understanding of the decision-making processes. In summary, our work focuses on combining well-established interpretability algorithms in the ML community with visual analytics to exploit the potential of algorithms in complex multi-module models for OpenQA. It may provide a reference for other similar visual analysis work on explaining complex models.

Generalizability. We have demonstrated the effectiveness of VEQA through case studies on DPR by using the NQ dataset, which is also applicable to other homogeneous datasets. To clarify the generalization, we briefly illustrate the relationship between OpenQA and general QA tasks as well as models. From the perspective of tasks, OpenQA is the most general QA task and the ultimate form of a search engine because it has no restrictions on the field of the problem [8]. By contrast, closed domain means that the source of the question is restricted to a certain domain, or the evidence for the answer is a given text or knowledge. In accordance with the source of information, OpenQA can be divided into three types, namely, text-based, knowledge-based, and hybrid. As described in Sec. 3.1.2, for the extractive text-based OpenQA task considered here, the second step, i.e., *answer extraction*, overlaps with other relatively simple generalized QA tasks, such as MRC. With regard to models, BERT is the current state-of-the-art model, and it is widely used in various components such as the Retriever/Reader of text-based QA as discussed in Sec. 2.2. In addition, some QA models incorporate generative modules [51], such as BART [42], while they are still Transformer-related. For knowledge-based QA, graphical neural networks and other models have a pivotal impact [74], [90]. Therefore, we argue that VEQA could be generalized to Transformer-based models for sub-tasks of OpenQA. With MRC as an example, we just need to remove the *Instance View* because the source of evidence for the MRC task is the given context, and the *Instance View* is no longer needed. Additionally, although VEQA is designed and evaluated for a two-stage pipeline model, it can be effortlessly applied to other Transformer-based end-to-end, Retriever-free, or other attention-related generative models, because the attribution algorithms, generation trees, and corresponding visual design are available for the attention mechanism and its variants. We also note that few OpenQA models combined with reinforcement learning [84] or graphical neural networks [73] are able to directly abstract the interactions between words through layer attribution methods to construct attribution trees for models. Similarly, our systems cannot be generalized to knowledge-based QA models because they are not composed of Transformer-based modules. Other network-specific interpretable methods need to be investigated.

Scalability. Our approach has some scalability issues in terms of both the algorithm and the visual design. The bottleneck of computational cost in our system comes from the large number of long-input and attribution methods, especially for layer attribution for each module of the serial model. For 3610 instances, the system takes about 20 hours to generate attribution data on the top-20 candidate passages that are 100 words long through the entire architecture with the layer conductance method with default parameters on 4 GPUs. However, considering the several days consumed by the Retriever in generating representations of a large corpus and retrieval with the given question, the run-time overhead caused by the attribution method is acceptable. For this reason, our system cannot and does not need to include arbitrarily modified questions, pruning attention heads, and other free forms of interaction as VisQA does. As for the visual design, the structures of the attribution trees in the *Tree View* will be illegible, with nodes and edges overlapping, if the attribution threshold is small. A possible method to mitigate this is to use small multiples to represent structures at the expense of size information. Similarly, if the saliency threshold is reduced, the column width in the *Instance View* will increase with the number of words presented in each column because a fixed scale is used, and information will not be fully displayed due to the limitation in screen real estate, but this can be solved by scrolling. However, we believe that the user burden and visual clutter caused by the change in the threshold do not need to be considered because the purpose of the *Instance View* and *Tree View* is to abstract important information. Users are informed of the overall distribution by observing changes in the tree structure and word count as the threshold is reduced, and capture key information with the use of a relatively large threshold, which has been confirmed by the experts.

Reliability. VEQA has been proven to provide a comprehensive multi-level exploration of OpenQA models. However, we only provide an abstraction of the attribution score distribution, i.e., a dependency tree at the layer-level, without presenting the original data in certain way. Doing so may be necessary because the attribution tree is constructed by a greedy algorithm with top-down heuristics, and some important information may be lost in the process. For example, suppose that two words are closely related (high attribution scores) within multiple layers. In this case, the algorithm no longer considers their connection at the lower layer because they have been added to the tree at the higher layer, even if their attribution score at the lower layer is much higher. In addition, we follow previous work and construct the attribution tree by using only attention attribution for caution because the relationship between attention attribution and embedding attribution is still not fully investigated. As mentioned in Sec. 6.1.3, the two attribution algorithms are not directly related. In future work, exploring the relation between embedding and attention attribution, and merging them into an attribution tree could be challenging. In addition to the fact that algorithms may lead to information omission, we are temporarily unable to quantitatively prove the reliability of the model improvement solutions drawn from the case study. The training of DPR requires massive resources, and it is trained using dozens of high-performance GPUs on tens of millions of data. Although we were unable to conduct complete experiments due to resource constraints, the qualitative and quantitative confirmation of the empirical conclusions from the visualization perspective through case studies are highly encouraging.

Learning curve and forgetting curve. As mentioned in Sec. 7, the experts considered VEQA a relatively complex system; they took 20-50 minutes to master the system workflow and explore it freely. The two experts (**E2** and **E3**) who were unfamiliar with OpenQA and attribution algorithms found it difficult to master the system. However, they claimed that the rewards of exploring our system were worth the time and showed great interest in using our system for deep exploration of various models in the future. The experts also mentioned that the one-on-one demonstrations greatly reduced the difficulty of mastering the system. A high-quality Readme document is likely to be helpful here. In addition, we examined the experts' forgetting curves at the frequency of weekly interviews. **E2** and **E3** were confused about the *tree view* after two weeks because it was not a form of visualization that they were familiar with. The remaining experts were still able to recall the workflow after four weeks.

9 Implications

We distill several implications from our work, especially from the processes of developing the requirements with experts at the early stages and eliciting feedback from them at the later stages.

How to derive requirements from AI experts. We encountered difficulties in distilling the experts' requirements. The experts' opinions and our expectations showed mismatch, and the experts' opinions were initially vague. Initially, we considered showing the full model flow and presenting the schema of VisQA [30] to the experts, but **E1** rejected this design because she did not want to explore individual passages by examining attention heads and would rather explore options with high prediction scores (**T3**). **E1**'s opinion guided our identification of the requirement to focus on the information flow of question-passage pairs in one instance (**R3**) and motivated the design of the *Context View*. In addition, **E1** initially formulated her requirement as "understanding how the basic capabilities of the model, such as counting, selection, and filtering, are related to the parameters." The expectation was directional but not specific; thus, we developed it further. Through literature review and discussion, we refined the fuzzy phrase "basic capabilities" into specific task types distinguished by the first few words of questions, and the fuzzy term "parameters" into specific layer responses of the model (**T4**). During the collaboration with **E1**, we summarized actionable steps as follows: (1) understand and adopt experts' domain "language", (2) inform experts about visualization features, (3) discuss the prototype, and if possible, (4) integrate it into their environment.

How to address the balance between the quantitative analysis and visual insights provided by the XAI system. Quantitative analysis of model results alone is not descriptive enough, and users may be easily confused about how to interpret them. Meanwhile, if only visual insight is available and quantitative analysis is lacking, users will not feel confident with the conclusions derived from the system, as evidenced by **E6**'s comment in Sec. 7.2, where he expressed an interest in seeing further statistical metrics in relation to the *Tree View*. To this end, we need to support experts' desire for an all-level quantitative analysis and provide comprehensive visual insights.

How to plug visual analysis tool into experts' development environment. ULCA [21] and PipelineProfiler [57] targeting simple models and datasets are integrated with Jupyter Notebook, and a few other works focusing on deep models, such as explAIner [71], are TensorBoard [1] plug-ins. The target users of our work are experts developing QA models, who generally use Python and are familiar with TensorBoard. Therefore, developing our explanation engine and views as reusable TensorBoard plug-ins is a viable pathway. However, this is not a straightforward task because plug-ins should be able to adapt to various data and models. The relative generalization of plug-ins places high expectations on the generalization of components. Moreover, the various possible configurations of workflows in QA and NLP need to be systematically surveyed. Similarly, NLP experts do not have a deep understanding of the configuration and expected effects of visual analysis tools. Therefore, in-depth communication and cooperation between the two fields must be promoted to distill requirements effectively and develop general tools.

10 Conclusion

We propose VEQA, a visual analytic system that helps experts to understand the decision flow of OpenQA models and provides insights into model enhancements. VEQA offers multi-level interpretable analysis for various modules and subsets from subset, instance, and candidate levels via various attribution algorithms. Through case studies and expert evaluation, we confirm that VEQA can abstract and visualize the decision flow of OpenQA models, help experts classify successful examples of decisions, and provide suggestions for model enhancement.

In our future work, we plan to add a model comparison module for analyzing a broad range of OpenQA models with a wide selection of datasets and disassemble the various views of VEQA into component libraries that can be embedded in the Python environment. We also plan to generate significantly reliable attribution trees and perform further statistical analysis through collaboration with domain experts.

Acknowledgments

The authors want to thank Yi Shan, Jingdong Zhang, Zhaoyu Zhou and reviewers for their suggestions. This work is supported by National Natural Science Foundation of China (NSFC No.62202105), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01, 2021SHZDZX0103), General Program (No. 21ZR1403300), Sailing Program (No.21YF1402900) and ZJLab.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] S. Abnar and W. Zuidema. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online, July 2020. Association for Computational Linguistics.

[3] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, and D. Batra. Vqa: Visual question answering. *International Journal of Computer Vision*, 123:4–31, 2015.

[4] G. Alicioglu and B. Sun. A survey of visual analytics for explainable artificial intelligence methods. *Computers & Graphics*, 102:502–520, 2022.

[5] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. *ArXiv*, abs/1911.10470, 2020.

[6] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10, 2015.

[7] Blakrlj, N. Erzen, S. Sheehan, S. Luz, M. Robnik-Sikonja, and S. Pollak. Attviz: Online exploration of self-attention for transparent neural language modeling. *ArXiv*, abs/2005.05716, 2020.

[8] M. Caballero. A brief survey of question answering systems. *International Journal of Artificial Intelligence & Applications (IJAIA)*, 12(5), 2021.

[9] F. Cheng, D. Liu, F. Du, Y. Lin, A. Zytek, H. Li, H. Qu, and K. Veeramachaneni. Vbridge: Connecting the dots between features and data to explain healthcare models. *IEEE Transactions on Visualization and Computer Graphics*, 28:378–388, 2022.

[10] F. Cheng, Y. Ming, and H. Qu. Dece: Decision explorer with counterfactual explanations for machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1438–1447, 2020.

[11] J. Choo and S. Liu. Visual analytics for explainable deep learning. *IEEE computer graphics and applications*, 38(4):84–92, 2018.

[12] J. Chung, S. Ahn, and Y. Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

[13] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does bert look at? an analysis of bert's attention. In *BlackboxNLP@ACL*, 2019.

[14] C. M. Collins, F. B. Viégas, and M. Wattenberg. Parallel tag clouds to explore and analyze faceted text corpora. *2009 IEEE Symposium on Visual Analytics Science and Technology*, pages 91–98, 2009.

[15] M. Denil, A. Demiraj, and N. de Freitas. Extraction of salient sentences from labelled documents. *ArXiv*, abs/1412.6815, 2014.

[16] 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*, 27(2):1160–1170, 2021.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[18] K. Dhamdhere, M. Sundararajan, and Q. Yan. How important is a neuron? *ArXiv*, abs/1805.12233, 2019.

[19] A. Drozdov, P. Verga, M. Yadav, M. Iyyer, and A. McCallum. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. *ArXiv*, abs/1904.02142, 2019.

[20] J. Falconer. Google: Our new search strategy is to compute answers, not links. https://thenextweb.com/news/google-our-new-search-strategy-is-to-compute-answers-not-links, June 2011.

[21] T. Fujiwara, X. Wei, J. Zhao, and K.-L. Ma. Interactive dimensionality reduction for comparative analysis. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):758–768, 2021.

[22] T. Fujiwara, J. Zhao, F. Chen, Y. Yu, and K.-L. Ma. Network comparison with interpretable contrastive network representation learning. *Journal of Data Science, Statistics, and Visualisation*, 2(5), 2022.

[23] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. Lineup: Visual analysis of multi-attribute rankings. *IEEE Transactions on Visualization and Computer Graphics*, 19:2277–2286, 2013.

[24] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[25] Y. Hao, L. Dong, F. Wei, and K. Xu. Self-attention attribution: Interpreting information interactions inside transformer. *arXiv preprint arXiv:2004.11207*, 2020.

[26] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6(1):24–43, 2000.

[27] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[28] Y. Hou, C. Wang, J. Wang, X. Xue, X. L. Zhang, J. Zhu, D. Wang, and S. Chen. Visual evaluation for autonomous driving. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1030–1039, 2021.

[29] S. Jain and B. C. Wallace. Attention is not explanation. In *NAACL*, 2019.

[30] T. Jaunet, C. Kervadec, R. Vuillemot, G. Antipov, M. Baccouche, and C. Wolf. Visqa: X-raying vision and language reasoning in transformers. *IEEE Transactions on Visualization and Computer Graphics*, 28:976–986, 2022.

[31] M. Kahng, N. Thorat, D. H. Chau, F. B. Viégas, and M. Wattenberg. Gan lab: Understanding complex deep generative models using interactive visual experimentation. *IEEE transactions on visualization and computer graphics*, 25(1):310–320, 2018.

[32] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, Nov. 2020. Association for Computational Linguistics.

[33] O. Khattab, C. Potts, and M. A. Zaharia. Relevance-guided supervision for openqa with colbert. *Transactions of the Association for Computational Linguistics*, 9:929–944, 2021.

[34] N. Kitaev, S. Cao, and D. Klein. Multilingual constituency parsing with self-attention and pre-training. In *ACL*, 2019.

[35] G. Kobayashi, T. Kuribayashi, S. Yokoi, and K. Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *EMNLP*, 2020.

[36] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky. Revealing the dark secrets of bert. *ArXiv*, abs/1908.08593, 2019.

[37] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. V. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[38] V. Lal, A. Ma, E. Aflalo, P. Howard, A. Simoes, D. Korat, O. Pereg, G. Singer, and M. Wasserblat. InterpreT: An interactive visualization tool for interpreting transformers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 135–142, Online, Apr. 2021. Association for Computational Linguistics.

[39] G. Lee, S. Kim, and S. won Hwang. Qadiver: Interactive framework for diagnosing qa models. In *AAAI*, 2019.

[40] J. Lee, A. Wettig, and D. Chen. Phrase retrieval learns passage retrieval, too. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3661–3672, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[41] K. Lee, M.-W. Chang, and K. Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics.

[42] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[43] J. Li, X. Chen, E. H. Hovy, and D. Jurafsky. Visualizing and understanding neural models in nlp. In *HLT-NAACL*, 2016.

[44] R. Li, W. Xiao, L. Wang, H. Jang, and G. Carenini. T3-vis: visual analytic for training and fine-tuning transformers in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 220–230, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[45] Z. Li, X. Wang, W. Yang, J. Wu, Z. Zhang, Z. Liu, M. Sun, H. Zhang, and S. Liu. A unified understanding of deep nlp models for text classification. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[46] 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*, 24(1):77–87, 2017.

[47] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017.

[48] S. Liu, T. Li, Z. Li, V. Srikumar, V. Pascucci, and P.-T. Bremer. Visual interrogation of attention-based models for natural language inference and machine comprehension. In *EMNLP*, 2018.

[49] Y. Liu, K. Hashimoto, Y. Zhou, S. Yavuz, C. Xiong, and P. S. Yu. Dense hierarchical retrieval for open-domain question answering. In *EMNLP*, 2021.

[50] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[51] Y. Mao, P. He, X. Liu, Y. Shen, J. Gao, J. Han, and W. Chen. Generation-augmented retrieval for open-domain question answering. *arXiv preprint arXiv:2009.08553*, 2020.

[52] L. McInnes and J. Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *ArXiv*, abs/1802.03426, 2018.

[53] T. Mikolov, M. Karafiát, L. Burget, J. H. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[54] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 13–24, 2017.

[55] Y. Ming, H. Qu, and E. Bertini. Rulematrix: Visualizing and understanding classifiers with rules. *IEEE transactions on visualization and computer graphics*, 25(1):342–352, 2018.

[56] Y. Nie, S. Wang, and M. Bansal. Revealing the importance of semantic retrieval for machine reading at scale. In *EMNLP*, 2019.

[57] J. P. Ono, S. Castelo, R. Lopez, E. Bertini, J. Freire, and C. Silva. Pipelineprofiler: A visual analytics tool for the exploration of automl pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):390–400, 2020.

[58] OpenAI. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/, November 2022.

[59] D. Pascual, G. Brunner, and R. Wattenhofer. Telling bert's full story: from local attention to global aggregation. *ArXiv*, abs/2004.05916, 2021.

[60] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, X. Zhao, D. Dong, H. Wu, and H. Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *NAACL*, 2021.

[61] S. Ramnath, P. Nema, D. Sahni, and M. M. Khapra. Towards interpreting bert for reading comprehension based qa. *arXiv preprint arXiv:2010.08983*, 2020.

[62] R. Ren, S. Lv, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J.-R. Wen. PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183, Online, Aug. 2021. Association for Computational Linguistics.

[63] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J.-R. Wen. RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[64] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

[65] A. Rücklé and I. Gurevych. End-to-end non-factoid question answering with an interactive visualization of neural attention weights. In *ACL*, 2017.

[66] D. S. Sachan, M. A. Patwary, M. Shoeybi, N. Kant, W. Ping, W. L. Hamilton, and B. Catanzaro. End-to-end training of neural retrievers for open-domain question answering. In *ACL/IJCNLP*, 2021.

[67] H.-J. Schulz, S. Hadlak, and H. Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE transactions on visualization and computer graphics*, 17(4):393–411, 2010.

[68] Y. Shen, S. Tan, A. Sordoni, and A. C. Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. *ArXiv*, abs/1810.09536, 2019.

[69] A. Shrikumar, J. Su, and A. Kundaje. Computationally efficient measures of internal neuron importance. *ArXiv*, abs/1807.09946, 2018.

[70] S. Silva, B. S. Santos, and J. Madeira. Using color in visualization: A survey. *Computers & Graphics*, 35(2):320–333, 2011.

[71] T. Spinner, U. Schlegel, H. Schäfer, and M. El-Assady. explainer: A visual analytics framework for interactive and explainable machine learning. *IEEE transactions on visualization and computer graphics*, 26(1):1064–1074, 2019.

[72] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush. S eq 2s eq-v is: A visual debugging tool for sequence-to-sequence models. *IEEE transactions on visualization and computer graphics*, 25(1):353–363, 2018.

[73] H. Sun, T. Bedrax-Weiss, and W. W. Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *ArXiv*, abs/1904.09537, 2019.

[74] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[75] K. Sun and A. Marasović. Effective attention sheds light on interpretability. In *FINDINGS*, 2021.

[76] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *ArXiv*, abs/1703.01365, 2017.

[77] I. Tenney, J. Wexler, J. Bastings, T. Bolukbasi, A. Coenen, S. Gehrmann, E. Jiang, M. Pushkarna, C. Radebaugh, E. Reif, and A. Yuan. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models, 2020.

[78] B. van Aken, B. Winter, A. Löser, and F. A. Gers. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1823–1832, New York, NY, USA, 2019. Association for Computing Machinery.

[79] L. van der Maaten and G. E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[80] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

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

[82] J. Wang, L. Gou, H.-W. Shen, and H. Yang. Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE transactions on visualization and computer graphics*, 25(1):288–298, 2018.

[83] J. Wang, Y. Li, Z. Zhou, C. Wang, Y. Hou, L. Zhang, X. Xue, M. Kamp, X. Zhang, and S. Chen. When, where and how does it fail? a spatial-temporal visual analytics approach for interpretable object detection in autonomous driving. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[84] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*, 2018.

[85] X. Wang, J. He, Z. Jin, M. Yang, Y. Wang, and H. Qu. M2lens: visualizing and explaining multimodal models for sentiment analysis. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):802–812, 2021.

[86] Z. J. Wang, R. Turko, and D. H. Chau. Dodrio: Exploring transformer models with interactive visualization. *ArXiv*, abs/2103.14625, 2021.

[87] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. P. Chau. Cnn explainer: learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1396–1406, 2020.

[88] S. Wiegreffe and Y. Pinter. Attention is not not explanation. In *EMNLP*, 2019.

[89] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[90] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*, 2021.

[91] X. Ye, R. Nair, and G. Durrett. Connecting attributions and qa model behavior on realistic counterfactuals. In *EMNLP*, 2021.

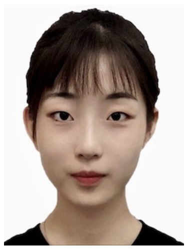
[92] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. *Computational Visual Media*, 7(1):3–36, 2021.

[93] D. Zhang, H. Zhang, H. Zhou, X. Bao, D. Huo, R. Chen, X. Cheng, M. Wu, and Q. Zhang. Building interpretable interaction trees for deep nlp models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14328–14337, May 2021.

[94] F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, and T.-S. Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *ArXiv*, abs/2101.00774, 2021.

Zekai Shao is an undergraduate at School of Information Science and Technology, Fudan University. His main research interests include visual analytics and explainable machine learning.

Shuran Sun is an undergraduate at School of Software, Fudan University. Her main research interests include visual analytics for interpretable machine learning.



Yuheng Zhao is currently a Ph.D student at the School of Data Science, Fudan University. Her main research interests include social media visualization and text visual analytics.

Siyuan Wang received the B.S. degree in software engineering from Fudan University, Shanghai, China, in 2018. She is currently working toward the Ph.D. degree with School of Data Science, Fudan University. Her research interests include natural language processing, question generation and answering, and machine reasoning.

Zhongyu Wei is an associate Professor in School of Data Science at Fudan University and he serves as the secretory in Social Media Processing (SMP) comiittee of Chinese Information Processing Society of China (CIPS). At Fudan, he is the director of Data Intelligence and Social Computing Reseach Lab (Fudan DISC), and member of a larger NLP group directed by Prof. Xuanjing Huang. Before joining Fudan, he was a postdoctoral researcher in Human Language Technology Research Institute at University of Texas at Dallas. He got his Phd in The Chinese University of Hong Kong in 2014. His research focuses on natural language processing, machine learning, with special emphasis on multi-modality information understanding and generation cross vision and language, argumentation mining and some cross-disciplinary topics. He has published more than 60 papers on top-tier conferences in related research fields, including ACL, EMNLP, ICML, ICLR, IJCAI, AAAI and so on.

Tao Gui is an associate professor at the Institute of Modern Languages and Linguistics of Fudan University. He is the key member of the FudanNLP group. He is a member of ACL, a member of the Youth Working Committee of the Chinese Information Processing Society of China, and the member of the Language and Knowledge Computing Professional Committee of the Chinese Information Processing Society of China. He has published more than 40 papers in top international academic conferences and journals such as ACL, ENNLP, AAAI, IJCAI, SIGIR, TASLP, and so on. He has served as area chair or PCs for SIGIR, AAAI, IJCAI, TPAMI, and ARR. He has received the Outstanding Doctoral Dissertation Award of the Chinese Information Processing Society of China, the area chair favorite Award of COLING 2018, the outstanding Paper Award of NLPCC 2019, and a scholar of young talent promoting projects of CAST.

Cagatay Turkay is a Professor at the Centre for Interdisciplinary Methodologies at the University of Warwick, UK and a Turing Fellow at the Alan Turing Institute, London, UK. His research investigates the interactions between data, algorithms and people, and explores the role of interactive visualisation and other interaction mediums such as natural language at this intersection. He frequently publishes his research on visualisation journals such as IEEE TVCG, CGF, and IEEE CG&A, as well as journals in machine learning and data mining, and also recently co-authored a coursebook titled “Visual Analytics for Data Scientists’. He has been awarded the EuroVis Young Researcher 2019 award and named a EuroGraphics Junior Fellow in 2019.

Siming Chen is an Associate Professor at School of Data Science, Fudan University. Prior to this, he was a Research Scientist at Fraunhofer Institute IAIS (Intelligent Analysis and Information Systems) and a Postdoc Researcher at the University of Bonn in Germany. He received his Ph.D. in computer science at the School of EECS, Peking University and received his BS degree in computer science at Fudan University. His research interests are visualization and visual analytics, with the emphasis on social media visualization, spatial-temporal visual analytics, and cybersecurity visual analytics. He has published 70 papers and more than 20 in top conferences and journals, including IEEE VIS, IEEE TVCG, EuroVis, etc. He served as multiple organizing chairs, committees and reviewers. He was awarded 10+ best paper/poster awards and honorable mentioned awards in multiple conferences, including EuroVA, ChinaVis, AGILE, IEEE VIS Poster and won multiple IEEE VAST Challenge Excellent Awards. For more information, please visit http://simingchen.me.