Interpreting BERT-based Text Similarity via Activation and Saliency Maps

Itzik Malkiel*  
Microsoft  
Tel-Aviv University  
Israel  
Avi Caciularu  
Bar-Ilan University  
Israel

Dvir Ginzburg*  
Microsoft  
Tel-Aviv University  
Israel  
Jonathan Weill  
Microsoft  
Israel

Oren Barkan  
The Open University  
Microsoft  
Israel  
Noam Koenigstein  
Microsoft  
Tel-Aviv University  
Israel

ABSTRACT
Recently, there has been growing interest in the ability of Transformer-based models to produce meaningful embeddings of text with several applications, such as text similarity. Despite significant progress in the field, the explanations for similarity predictions remain challenging, especially in unsupervised settings. In this work, we present an unsupervised technique for explaining paragraph similarities inferred by pre-trained BERT models. By looking at a pair of paragraphs, our technique identifies important words that dictate each paragraph’s semantics, matches between the words in both paragraphs, and retrieves the most important pairs that explain the similarity between the two. The method, which has been assessed by extensive human evaluations and demonstrated on datasets comprising long and complex paragraphs, has shown great promise, providing accurate interpretations that correlate better with human perceptions.

CCS CONCEPTS
• Computing methodologies → Machine Learning, Natural Language Processing; • Information systems → Presentation of retrieval results.

KEYWORDS
Self-supervised, Deep Learning, Transformers, Attention Models, Explainable AI, Interpretability

ACM Reference Format:
Itzik Malkiel, Dvir Ginzburg, Oren Barkan, Avi Caciularu, Jonathan Weill, and Noam Koenigstein. 2022. Interpreting BERT-based Text Similarity via Activation and Saliency Maps. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512045

1 INTRODUCTION
The increasing availability of artificial intelligence systems has brought about an increasing demand for methods to intuitively explain machine inference and predictions [2, 24, 42]. Machine explanations are utilized for a multiplicity of applications. From increasing users’ trust [9, 27, 38] and improving interpretability [4, 10, 19, 20, 24] to debugging [31]. In some cases, explanations are used to enhance the autonomy of people and facilitate direct oversight of algorithms [18]. As a result, some countries have regulations that make explanations mandatory in sensitive cases [29, 41].

Transformer-based models have revolutionized the fields of natural language processing [22, 33, 46] and recommender systems [34, 44], improving upon other neural embedding models [3, 5, 7, 8, 13–15, 17, 36, 37, 40]. Significant strides were made in tasks such as machine translation [46], sentiment analysis [49], semantic textual similarity [16, 23, 35], and item similarity [6, 11, 12, 28, 34]. However, transformers employ a complex attention-based architecture comprising hundreds of millions of parameters that cannot be decomposed into smaller more interpretable components. Hence, explaining transformer-based models is still an open question.

In this paper, we present BTI - a novel technique for explaining unlabeled paragraph similarities inferred by a pre-trained BERT model. BTI produces interpretable explanations for the similarity of two textual paragraphs. First, both paragraphs are propagated through a pre-trained BERT model, yielding contextual paragraph representations. A similarity score such as the cosine similarity is used to measure the affinity between the paragraphs. Gradient maps for the first paragraph’s embeddings are calculated w.r.t. the similarity to the second paragraph. These gradient maps are scaled by multiplication with the corresponding activation maps and summed across the feature dimensions to produce a saliency score for every token in the first paragraph. The token saliency scores are then aggregated to words, yielding word scores.

Next, BTI performs the same procedure, with the first paragraph and second paragraphs reversed. This yields word scores for the second paragraph, calculated w.r.t to the similarity with the first. Finally, the algorithm matches word-pairs from both paragraphs,
scoring each pair by the importance scores associated with its elements and the similarity score associated with the pair. The most important word-pairs are then detected and retrieved as explanations.

Our contributions are as follows: (1) we present BTI, a novel method for interpreting paragraph similarity in unlabeled settings. (2) we show the effectiveness of BTI in explaining text similarity on two datasets and demonstrate its ability to pass a common sanity test. (3) we compare BTI with other alternatives via extensive human evaluations, showcasing its ability to better correlate with human perception.

2 RELATED WORK

A seminal breakthrough in visualization and interpretation of deep neural networks was presented in [51]. In this work, the authors proposed to employ the deconvolution operations [52] on the activation maps w.r.t. the input image, generating visual features to interpret model predictions.

A different approach was employed by [43, 50] where guided backpropagation (GBP) was proposed as an explanation method that is not restricted to visual classification models. GBP visualizes the output prediction by propagating the gradients through the model and suppressing all negative gradients along the backward pass, resulting in saliency maps of the most influenced input parts w.r.t. the downstream task.

To test the reliability of common explainability methods, Adebayo et al. [1] proposed the parameter randomization test. The test suggests propagating the same input through a pre-trained and randomly initialized network to estimate the method’s dependence on the underlying network weights. The proposed sanity check reveals that most of the methods described above do not pass the test, i.e., they are independent of the model parameters and therefore are not adequate for providing satisfactory model explanations. In Section 4.1, we show that BTI passes those important sanity checks.

Based on these canonical works, the NLP community introduced interpretability tools for architectures other than CNNs. The first works that proposed interpretation of textual-based networks include [32], and [39]. In [32], the authors offer to justify model predictions by generating important fragments of the inputs. Their technique utilizes explanations during training, integrating an encoder-generator paradigm that can be used to predict explanations during inference. In [39] the authors propose a “black-box” classifier interpretability using local perturbation to the input sequence to explain the prediction. However, since both methods rely on supervision, they cannot be employed in unlabeled settings.

Another vast problem of explainability methods for NLP tasks is the variation of the input length. To adhere to this problem, the authors in [53] use a Variational Autoencoder for the controlled generation of input perturbations, finding the input-output token pairs with the highest variation. While several authors adopted the sequence-pair explainability scheme, they only tackle input-output relations, hence incompatible for unlabeled text-similarity tasks.

Since the emergence of the transformer architecture [46], several groups have tried to offer interpretation techniques for the models. exBERT [30], for example, published a visual analysis tool for the contextual representation of each token based on the attention weights of the transformer model. AllenNLP Interpret [47] offers token-level reasoning for BERT-based classification tasks such as sentiment analysis and NER using published techniques such as HotFlip [25], Input Reduction [26] or vanilla gradients analysis.

In integrated gradients (IG) [45], the authors propose an explainability method that approximates an integral of gradients of the model’s output w.r.t. the input. To this end, IG generates a sequence of samples by interpolating the input features with 0, computing gradients for each sample in the sequence, and approximating an integral over the produced gradients. The integral is then used to score the importance of each input feature. The authors of SmoothGrad proposed a similar yet simpler technique, by feed-forwarding noisy versions of the input, and averaging the gradients of the different outputs with respect to the objective function.

3 METHOD

In this section, we present the mathematical setup of BTI and its applications for explainable text-similarity in self-supervised pre-trained language models. We build our method upon BERT, however, BTI can be applied with other Transformer-based language models.

3.1 Problem Setup

Let \( \mathcal{V} = \{v_i\}_{i=1}^V \) be the vocabulary of words in a given language. Let \( S \) be the set of all possible sentences induced by \( \mathcal{V} \). Additionally, let \( P \) be the set of all possible paragraphs generated by \( S \).

BERT can be defined as a function \( B : P \rightarrow \mathbb{R}^{N \times h} \), where \( h \) is the hidden layer size, and \( N = 512 \) is the maximal sequence length supported by the model. In inference, BERT receives a paragraph \( p \in P \) and decomposes it into a sequence of \( q \in \mathbb{N} \) tokens \( (p_j)_{j=1}^q \), by utilizing the WordPiece model [48]. The sequence is then wrapped and padded to \( N \) elements, by adding the special \( CLS, SEP \) and \( PAD \) tokens. This token sequence can be written as \( Ip := (CLS, (p_j)_{j=1}^q, SEP,...,PAD) \).

In BTI, all tokens are encoded by three learned functions: token, position and segment embeddings, denoted by \( T, O, \) and \( G \), respectively. The token embedding transforms tokens’ unique values into intermediate vectors \( T(Ip) \in \mathbb{R}^{N \times h} \). The position embedding encodes the token positions to the same space, \( O(Ip) \in \mathbb{R}^{N \times h} \). The segment embedding is used to associate each token with one out of two sentences \( G((0, 1)^N) \in \mathbb{R}^{N \times h} \) (as standard BERT models are trained by sentence pairs).

In this work, we feed BERT with single paragraphs and leave the use of paragraph-pairs sequences to future investigation. The principle behind this choice stems from two aspects: (1) in many cases, paragraph similarity labels do not exist, therefore, we cannot fine-tune the language model to the task of paragraph-pairs similarity. This entails the use of a pre-trained language model, that is commonly trained by sentence pairs or chunks of continuous text, and does not specialize in non-consecutive paragraph pairs. Therefore, the inference of non-consecutive paragraph-pairs may introduce instabilities, due to the discrepancy between both phases.
Algorithm 1 BTI algorithm. The MeanShift function returns the cluster index for each sample. The clusters are sorted in descending order w.r.t. to their centroids (i.e. the top-performing cluster receives the index 0). Params: top_k = 2, p1, p2 ∈ P.

```
function MatchWords(wi1, wi2, w1, w2, reversed)
    M := {}
    for 1 ≤ i ≤ |wi1| do
        m, c ← 1, C(wi1, wi2)
    for 2 ≤ j ≤ |wi2| do
        c ← C(w1, w2)
        if c ≤ c: m, c ← j, c
    end for
    if reversed then
        M ← M ∪ \{(w2_m, w2_c), (w2_j · (w1_m · c))\}
    else
        M ← M ∪ \{(w1_m, w1_c), (w1_j · (w2_m · c))\}
    end if
end function

function F_p(p)
    F_p ← \frac{1}{|p|} \sum_{i=2}^{p} B(p_i)
    return F_p
end function

function TokenSalienty(p1, p2)
    g ← \frac{\nabla \Phi(p1, p2)}{\|\Phi(p1, p2)\|_2}
    s ← \min\sum_{k=1}^{n} \phi(E(p1), \phi(g1))_k \sum_{j=1}^{n}
    s ← \min\max(s)
    return s
end function

function TopWords(m1, m2, top_k)
    M ← m1 ∪ m2
    M* ← (M1), 2|j=1
    S := \phi(m1 + m2) \leftrightarrow MeanShift(M*) \rightarrow word-pair scores
    for each sample
        T := {}
        for 1 ≤ i ≤ |M| do
            T ← T \cup \{M_i\}, if Si ≤ topC
        end for
    return T
end function
```

(2) the technical limitation of maximal sequence length in BERT architecture, for which feeding two paragraphs as a unified sequence may exceed the limit of 512 tokens.

3.2 BERT Interpretations (BTI)

By propagating p through BERT, the model produces a sequence B(p) ∈ R^{N×h} of latent tokens. Each element in B(p) is associated with its matched element in IP. A feature vector can be computed for each paragraph by average pooling the output tokens as follows:

\[
F_p := \frac{1}{q+1} \sum_{i=2}^{q+1} B(p_i),
\]

omitting the latent elements associated with the CLS, SEP and PAD input tokens.

The similarity between paragraph-pairs can be interpreted by highlighting and matching important words from each element. The highlighted words in both elements should dictate the semantical similarity of the two paragraphs in a way that correlates with human perception. Hence, given two paragraphs denoted by p1 ∈ P and p2 ∈ P, we build BTI to identify important word-pairs with similar semantics M := \{(x_i, y_i)\}_{i=1}^e, where x_i ∈ p1 and y_i ∈ p2, for all i, and e is the number of pairs detected by BTI.

Since each paragraph is fed as a separate sequence, we build our method upon the T and O embeddings. Specifically, given the paragraphs p1 and p2, BTI first calculates saliency maps, for each paragraph, by utilizing the activation map values E(IP1) ∈ \mathbb{R}^{q1×h} and E(IP2) ∈ \mathbb{R}^{q2×h} where

\[
E(IP) := T(IP) + O((i)^N)
\]

along with the gradients calculated on the same activations w.r.t. to a cosine score between the feature vectors of both paragraphs. q1 and q2 denote the number of tokens in the WordPiece decomposition of p1 and p2, respectively.

The BTI algorithm is depicted in Alg.1. In lines 1-2, BTI invokes the TokenSaliency function to infer a token-saliency score for each token in the given paragraph-pair. The TokenSaliency is first applied to (p1, p2), then, in line 2, the roles of p1 and p2 are reversed. Since both sides are analogue, we will describe TokenSaliency by its first application. For a given paragraph pair (p1, p2), the function propagates each paragraph through BERT, calculates the feature vectors F_1, F_2, and
the partial derivations, \( g(p_1, p_2) \in \mathbb{R}^{N \times q_2} \), of the input embeddings \( E(p_2) \) w.r.t. to the cosine between the two feature vectors \( F_{p_1} \) and \( F_{p_2} \) (the first feature vector is used as a constant, the second is generated by propagating \( p_2 \) through the model to derive gradients on its intermediate representations). Formally:

\[
g(p_1, p_2) = \frac{\partial C(F_{p_1}, F_{p_2})}{\partial E(p_2)}
\]  

where \( C \) is a similarity function. In our experiments we utilize the cosine similarity function\(^1\). Importantly, the positive gradients of the above term are the ones that maximize the cosine similarity between the two paragraphs (whereas stepping towards the direction of the negative gradients, would entail a smaller cosine similarity). An illustration of this gradients calculation can be seen in Fig. 1. For completeness, we also provide a figure demonstrating the effectiveness of the gradients, calculated on a representative sample of seed-candidate descriptions. In Fig. 2 we present the gradients’ norm calculated on the words of a given candidate description w.r.t. to the similarity with its matched seed description.

The gradients are then multiplied by the activation maps of the same embeddings:

\[
s = \text{NRM} \left[ \sum_{k=1}^{h} \phi \left( \left( E(p_2) \circ g_1 \right)_k \right) \right]^{q_1}
\]  

where \( \phi \) is the ReLU activation function, \( \circ \) is the Hadamard product, and \( \text{NRM} \) is the min-max normalization, which transforms the data to \([0,1]\).

The motivation behind Eq. 4 is as follows: we expect important tokens to have embedding values and gradients that agree in their sign - namely both positive or both negative. This indicates that the gradients that stem from \( C(F_{p_1}, F_{p_2}) \) emphasize the relevant parts of the embeddings that are important for the similarity between the two paragraphs. Additionally, embeddings and gradients with higher absolute values are more significant than those with values near zero.

In line 3-4, BTI applies the WordPiece\(^{-1}\) function. This function receives (1) the tokens, (2) the latent token representations\(^2\), and (3) the token-saliency scores of a given paragraph. The function then aggregates all arguments to word-level representation, retrieving (1) whole words (rebuilt from tokens), (2) latent representation for each word, and (3) word-saliency scores. The second and third aggregations employ predefined functions \( \varphi \) and \( \psi \) on the latent tokens and token-saliency scores associated with the same word, respectively. The result word-level representation is then retrieved as an output. \( \tilde{w}^1 \) and \( \tilde{w}^2 \) denote the words sequences produced by aggregating the tokens of \( p_1 \) and \( p_2 \), respectively. \( \tilde{w}^1 \) and \( \tilde{w}^2 \) denote the aggregated latent word-level representation of \( p_1 \) and

\[^1\] \( C(v, u) = \frac{w_{v \cdot u}}{\|v\| \cdot \|u\|} \) where \( v, u \in \mathbb{R}^h \)

\[^2\] \( \text{generated by propagating the paragraph through BERT} \)
where $\widetilde{w}_i^1 \in \widetilde{w}^1$ and $\widetilde{w}_j^2 \in \widetilde{w}^2$ are the means of the latent tokens associated with the words $w_i^1$ and $w_i^2$, respectively.

In addition to conducting matches between word pairs, the MatchWords function calculates a word-pair score for each pair. The word-pair score represents the accumulated importance of the pair and is defined as the multiplication of the word scores of both words along with the cosine similarity between the latent representation of the words. Formally, the word-pair score of the pair $(w_i^1, w_i^2)$ can be written as:

$$U(w_i^1, w_i^2) := c_{i1} \cdot s_{i1} \cdot s_{i2}$$  \quad (6)

where $c_{i1}$ is the cosine similarity between the embeddings of $w_i^1$, $w_i^2$, and $s_{i1}$, $s_{i2}$ are the saliency scores of the words $w_i^1$ and $w_i^2$, respectively.

In line 7, BTI calls the TopWords function, which retrieves a sub-sequence of the most important word-pairs by clustering the word-pairs scores, and identifying the top-performing clusters. For retrieving the most important word-pairs, we run the MeanShift algorithm [21] on the set of word-pairs scores, to obtain the modes of the underlying distribution. MeanShift is a clustering algorithm that reveals the number of clusters in a given data and retrieves the corresponding centroid for each detected cluster. In our case, MeanShift is applied to the 1D data of all $U(w_i^1, w_i^2)$ and identifies the subsets of the most important pairs, as the cluster associated with the top_k centroids. In BTI, top_k is a predefined hyperparameter. The detected most important word-pairs are retrieved as a sequence, which can be then visualized to interpret the similarity between the given two paragraphs.

4 EXPERIMENTS

We first demonstrate BTI’s ability to explain the similarity between items with similar descriptions, evaluated on a fashion and wine reviews\(^3\) datasets, comprising ~1000 and ~120K items, respectively. The items from both datasets contain textual descriptions in the form of single paragraphs. In our experiments, we train a separate RecoBERT [34] model for each dataset. RecoBERT is a BERT-based model trained for item similarity by optimizing a cosine loss objective and a standard language model. The choice of RecoBERT stems from its ability to effectively score the similarity between two bodies of text, and since it does not require similarity labels for training.

RecoBERT training begins by initializing the model with a pre-trained BERT model and continues training by feeding the model with title-description pairs, extracted from the items in the dataset. The training objective of RecoBERT comprises a standard masked language model (similar to BERT) along with a title-description model, which enforces the model to embed item descriptions under a well-defined metric (such as the cosine function). Specifically, RecoBERT optimizes the BERT-based backbone model to embed text pairs extracted from the same item as vectors pointing to the same direction, while text pairs sampled from different items are reinforced to have in opposite directions.

\(^{3}\)https://www.kaggle.com/zynicide/wine-reviews
Figure 3: BTI results for explaining item similarities in the fashion (a-b) and wines (c) datasets. Each sample comprises two paragraphs: (1) the seed item description (top left), and (2) the description of the most similar item w.r.t. the seed (bottom left). The words retrieved by BTI are associated with saliency scores (shown next to each word). The “matched-words” column depicts the word-pairs chosen by BTI. The “word similarity” column presents the cosine scores between the word pairs. The “word-pair score” column exhibits the score from Eq.6.

Given a trained model, we infer item-to-item similarities for each dataset by propagating all item descriptions through the specialized RecoBERT model. For each item, we extract a feature vector, as presented in Eq. 1. Then, given a seed item \( s \), we calculate the cosine similarity between \( F_s \) and the feature vectors of all the other items in the dataset. The \emph{candidate} item that maximizes the cosine similarity is retrieved as the most similar item. Finally, in order to interpret the similarity between each seed and candidate item, we employ BTI on the descriptions of both.

Figure 3 presents the interpretations of BTI for three representative seed-candidate items from the fashion and wine datasets. In the first sample, comprising two paragraphs of two glove items, BTI highlights the words “gloves” in both items, which indicates the type of the item, as well as the phrases “touchscreen-compatible” and “smartphone-friendly”, that strongly explain the similarity between the items. For the second sample, BTI highlights the category of both items (“shirt”) and other key characteristics including “running” and “sports”, as well as “ventilating” and “breathable”, that strongly correlate. The third sample in the figure, consists of two white wine items from the wines dataset. For this sample, BTI highlights the variety of wine grape (“Viura”), the color of both wines (“light”, “translucent”) as well as a dominant flavor in both items (“citrus”, “citrusy”). By obtaining a faithful interpretation for the similarity between paragraphs that relies on intermediate representations extracted from the model, BTI reveals the reasoning process behind RecoBERT’s embedding mechanism. Importantly, any use of other techniques for interpreting paragraph similarities that do not utilize RecoBERT’s intermediate representations would be independent
of the model weights and, therefore, will not assess the sanity of the model. This is not the case in BTI, which strongly relies on the model intermediate representations, and therefore, can assess the validity of the model. Due to the above property, BTI can allow researchers to debug their language models by analyzing the embeddings of similar paragraphs, even when similarity labels do not exist.

4.1 Sanity Test

To assess the validity of BTI for explanations, we conduct the parameter randomization sanity test from [1]. This test uses the explainability method twice, once with random weights and once with the pre-trained weights.

Fig. 4 exhibits two representative samples processed twice by BTI. In the first application, BTI is applied with a randomly initialized network and BTI applied to the pre-trained weights.
4.2 Human Evaluations of Explainability

The compared baselines are Vanilla Gradients (VG), Integrated Gradients (IG) [45], and a TF-IDF based method (denoted by TF-IDF-W2V).

- Vanilla Gradients (VG) - This method interprets the gradient of the loss with respect to each token as the token importance.
- Integrated Gradients (IG) - This method defines an input baseline \( x' \), which represents an input absent from information. We follow [47] and set the sequence of all zero embeddings as the input baseline. For a given input sentence \( x \), word importance is defined by the integration of gradients along the path from the baseline to the given input. IG requires multiple feed-forward to generate the explainability of each sentence.
- TF-IDF-W2V - As a baseline method, we consider an alternative model to the word-pair scoring from Eq. 6:

\[
\Lambda(w_i^1, w_j^2) := T(w_i^1)T(w_j^2)C(W(w_i^1), W(w_j^2))
\]

where \( T \) and \( W \) are the TF-IDF scoring function and the word-to-vector mapping obtained by a pretrained word2vec (W2V) model [36], respectively. Hence, we dub this method TF-IDF-W2V. We see that \( \Lambda \) incorporates both the general words importance (captured by TF-IDF scores) as well as their semantic relatedness (captured by W2V).

Table 1 presents an ablation study for BTI and a comparison with vanilla gradients (VG), integrated gradients (IG), and TF-IDF-W2V on the fashion dataset. For the ablation, the following variants of BTI are considered: (i) utilizing the gradients and activations of the last BERT layer instead of the first one. (ii) using token-saliency scores based on the activation maps alone, and (iii) using scores solely based on gradients. The last two variants eliminate the Hadamard multiplication in Eq. 4, utilizing one variable at a time.

To compare BTI with VG, and IG, we replace the score being differentiated by the cosine similarity score (rather than the class logit). Since the above methods are not specific, i.e., they produce intensity scores for all words in the paragraphs, the comparison is applied via scoring the correlation of the most highlighted words, retrieved by each method, with the human perception.

To assess our full method’s performance against other alternatives and compare it to the ablation variants above, we report interpretation scoring, using a 5-point-scale Mean Opinion Score (MOS), conducted by five human judges. The same test set, comprising 100 samples, was ranked for all four BTI variants (full method and three baselines) and three baselines above. The scoring was performed blindly, and the samples were randomly shuffled. Each sample interpretation was ranked on a scale of 1 to 5, indicating poor to excellent performance.

The results in Tab. 1, highlight the superiority of BTI compared to other alternatives, indicating a better correlation with human perception. As for the ablation, the results indicate the importance of utilizing the gradients on the embedding layer from Eq. 2 as described in this paper and emphasize the importance of the multiplication between gradient and activations.

### Table 1: Human evaluations of explainability. Shown are mean and standard deviation results over all human judges.

| Model                  | MOS    |
|------------------------|--------|
| (i) BTI (last layer)   | 2.6 ± 0.5 |
| (ii) BTI (activations) | 2.4 ± 0.6 |
| (iii) BTI (gradients)  | 3.9 ± 0.3 |
| TF-IDF-W2V             | 2.5 ± 0.5 |
| IG                     | 3.5 ± 0.4 |
| VG                     | 3.6 ± 0.5 |
| BTI                    | 4.3 ± 0.3 |

5 SUMMARY

We introduce the BTI method for interpreting the similarity between paragraph pairs. BTI can be applied to various natural language tasks, such as explaining text-based item recommendations. The effectiveness of BTI is demonstrated over two datasets, comprising long and complex paragraphs. Additionally, throughout extensive human evaluations, we conclude that BTI can produce explanations that correlate better with human perception. For completeness, we show that BTI passes a sanity test, commonly used in the computer vision community, that estimates the reliability of explainability methods.

Finally, we believe that BTI can expedite the research in the domain of language models by identifying failure modes in transformer-based language models and assessing deployable language models’ reliability.
[44] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM).

[45] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In International Conference on Machine Learning. PMLR, 3319–3328.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

[47] Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models. arXiv:1909.09251 [cs.CL]

[48] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016).

[49] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. arXiv preprint arXiv:1904.02232 (2019).

[50] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding Neural Networks Through Deep Visualization. arXiv:1506.06579 [cs.CV]

[51] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In European conference on computer vision. Springer, 818–833.

[52] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. 2011. Adaptive deconvolutional networks for mid and high level feature learning. In 2011 International Conference on Computer Vision. IEEE, 2018–2025.

[53] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples. arXiv preprint arXiv:1710.11342 (2017).