Transformer-Vis: Visual Analysis on Attention Mechanism

Mengxi Wu
New York University
mw4355@nyu.edu

Yuan Li
New York University
yl6606@nyu.edu

Abstract

Transformer with attention mechanism is widely used for plenty of natural language processing tasks. This project is a visualization tool to interpret how to gain attention and how attention works. It helps researchers understand their transformer models. We use different statistical graphs to illustrate the Q, K, V matrices which are obtained during the training process and the softmax values for each word in the input sentence. In this report, we introduce some basic concepts of transformer and attention mechanism including how the attention is calculated and the widely used structure of transformer. Previous works and feedback from domain experts will also be discussed in the report.

1. Introduction

One of challenge sub-fields of natural language processing is sentiment analysis. Through sentiment analysis, businessmen can make more suitable plan for the development and improvement of future products. Together with many recent literatures which have decent performance in solving sentiment analysis, transformer, a novel neural architecture based on attention mechanism has achieved state-of-art results. However, the principles that illustrate how attention mechanism can make outstanding performance possible still remains mystery. So far, for most of researchers, attention mechanism is still a black box.

In this project, we use visualization techniques to examine the internal structure and the calculation process of attention mechanism. The results illustrate how each word associate with each other according to the softmax values. We also use graphs to interpret the distribution of values in the Q,K,V matrices in calculation process of attention mechanism. The source code can be found in GitHub[1].

2. Related Work

In this section, we review the attention mechanism currently popular in NLP tasks and some related approaches to visualize attention mechanism.

2.1. Attention

The attention mechanism was firstly used in [3] for RNN decoder-encoder architecture. After that, it gradually gained more and more interest, like [5], [2] and [6]. Compared with RNN network, attention mechanism has more ability to concentrate on the relationship between preceding and the following of a sequence at the same time. So it is better than RNN network for the modeling of long dependencies.

Firstly, Bahdanau et al.[3] proposed attention to receive the most helpful features from the past encoder hidden states with a average score as a context feature for later calculation. A one-dimensional array is given to maintain the contributions from different hidden states to the average score.

A specific calculation process is that set the past decoder states as the query matrix, and the past encoder hidden states as key and value matrices. We can firstly compute the average scores by multiplying the query matrix and the key matrix. Then we use it as a weight and perform calculations on value matrix to get the final output.

The more formal equations can be written as follow: Consider query matrix q, key matrix $(k_1, k_2, ..., k_n)$ and value matrix $(v_1, v_2, ..., v_n)$, the output z is:

$$z = \sum_{i=1}^{n} \alpha_i(v_i) \tag{1}$$

$$\alpha_i = \frac{expf(k_i, q)}{\sum_{j=1}^{n} expf(k_i, q)} \tag{2}$$

$$f(k_i, q) = \frac{(k_i)(q)^T}{\sqrt{d_k}} \tag{3}$$

where z_k is the dimension of the key vectors. The equations are from [2] and [13].

In order to solve sequence-related problem, Ambartsoumian et al. [2] introduced a new novel mechanism called Self-attention. It is modified based on original attention mechanism from [3]. Self-attention can apply the attention

[1]GitHub link: https://github.com/FoxerLee/Transformer-Vis

mechanism to all positions of the input sentence. Similar to the original version, self-attention creates three matrices to represent the geographic information for each word in the input sentence.

Set the three matrices as Q, K and V, the formal equation is:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4}$$

2.2. Visualization on Neural Network

Some novel work has been done on visualizing neural network.

In order to create an unattended-automatic neural network training method, and analyze the expression of the model, a bunch of visualization systems are designed to track parts of a neural network. For instance, iVisClassifier [4] is applied to analyze classification results based on linear discriminant analysis (LDA). It full interacts all the reduced dimensions instead of only two-dimensional results. Alsallakh et al. [1], on the other hand, gives users a fresh perspective to observe multi-dimensional, multi-class data.

Unlike visualizing convolutional neural network, which would be easy to directly map parameter values in different layers to the original pictures, words need to be converted into word vectors in the NLP task, adding more difficulties to match the parameters with the word vectors. Ming et al. [10] shows that co-clustering can be appropriate to understand the response of the input sentence from parameters of hidden state units. Liu et al. [9], instead, combine visualization tools and models with a flexible integration, giving users a change to interact the model with checking how internal hidden states and outputs change in some parts of the pipeline.

3. Method

The purpose of Transformer-Vis is to support experts who want to know the internal exchanges of attention layer of transformer model. Therefore, we need to find out what are experts most care about. In this section, we will firstly analysis the requirements for understanding the attention layer. Then the proposed techniques and UI design are discussed.

3.1. Task Analysis

The major driving force of Transformer-Vis is mean to let users better understand the attention layer in the transofrmer model. So based on communications with domain experts and previous related literature, the formal tasks of Transformer-Vis are:

T1: Intuitively judge the degree of convergence of model training. A basic requirement of visualization tools is to help users interpret the predicted results, i.e., how can an input prediction reflect the behavior of the model.

T2: Detect dynamic changes inside the attention layer. Some current visualization tools focus on different attention layers [14]. But the inside layer weights are also very important. Therefore, our needs should include visually explaining how the attention layer captures sentiment information inside attention layer.

T3: Explain the correlation between words and predict results. By displaying word vectors in various forms, you can better understand the relationship between words, which is beneficial for evaluating the accuracy of the model, as mentioned by [12].

3.2. System Overview

Our visualization system is designed to visualize the SSAN model [2], which is a sentiment analysis model based on the modification of the transformer. Our visualization tool uses D3.js. Unlike using the basic D3.js, we develop on its official online visual development platform Observable. All visualization codes can be found here[2].

Based on the previous discussion of the task, transformer-vis mainly has two large visualization modules: Matrix collections and Word collections. The following sections will specifically discuss the implementation and effects of the two modules.

3.3. Words collections

Word Cloud

A word cloud is a clear and popular visualization of words. It is a beautiful image that conveys the essential information in a single glance. Word cloud provides a visual illustration of the data of text and text. For example, the font size is associated with the frequency of a word (e.g.,how many times a word appear in a text). The font size of the word is larger when the frequency is larger. Similar to the font size, the color of the word can also be used to distinguish the prominence of attribute of a word.

Our word cloud is generated according the frequency of a word. The frequency of a word is calculated in the following way. We first obtain the word embedding vector for each word in the input sentence. Each word embedding vector is 1×300 in size. If the input sentence contains 8 words, then the word embedding matrix w for the first sentence is 8×300 in size. For each each column in w, we find the maximum element. Then, we find which row the maximum element is at and increase the frequency of the word that corresponds to the row. For example, if third column has the largest element at seventh row, we add 1 to the frequency of the seventh word. The frequency of each word is the number of time that the word has maximum

[2]https://observablehq.com/@wmx567?tab=notebooks

value in a column. The sum of frequencies of all the words is 300 in this case.

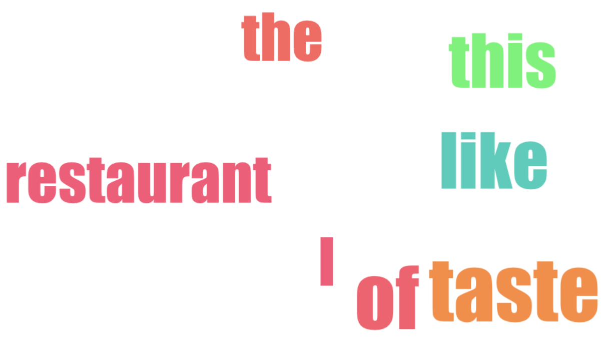


Figure 1. Word cloud for sentence: I like the taste of this restaurant

The font size of the word is based on the frequency of the word. We use $15 \cdot \log(frequency)$ as the font size. Similarly, The color of a word is selected by using value $\log(frequency)$.

Comparison Matrix

A comparison matrix is usually used to visualize similarities. It can also illustrate the differences between items. It is a great visualization tool to help user structure and classify items which the user want to compare. It can apply to ideas, strategies, services, products or more complex and abstract concepts. Additionally, the color of each cell in the comparison matrix can show the how strong association is between each item.

We use softmmax values to generate the comparison matrix. First, we train the model to obtain W_q, W_k, W_v matrices. To find the softmax values for a word w, we first calculate the queries (q), keys (k), values(v). The q is obtained from multiplying the word embedding vector with matrix W_q. The k is obtained from multiplying the word embedding vector with matrix W_k. The v is obtained from multiplying the word embedding vector with matrix W_v. The second step is to calculate the scores. We need to calculate a score for every word of the input sentence against the selected word. For example, in Figure 2, we want to calculate the score for "Machine" that is against "Think". The value is $q_1 \times k_2$. The score for "Machine" that is against "Machine" is $q_1 \times k_1$. After getting the scores, we divide each score by the square root of the dimension of the key vectors. Then, we apply softmax operation on these scores. The output values from the softmax operation is the softmax values for word w. We repeat the softmax values calculation process for each word in the input sentence. After we obtain all the softmax values, we create a matrix with size $L \times L$ where L is length of the input sentence. Then, we fill each cell with corresponding softmax value. For example, in Figure 3, x position is "like" and y position is "the" should be the softmax value from score that "like" is against by "the". In this case, it is 0.0773. The color of a cell is indicating whether the value is large or small.

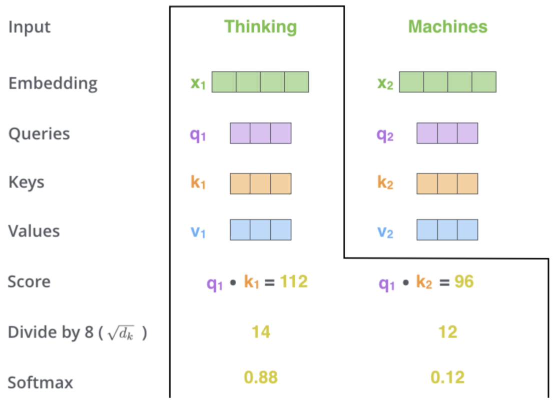
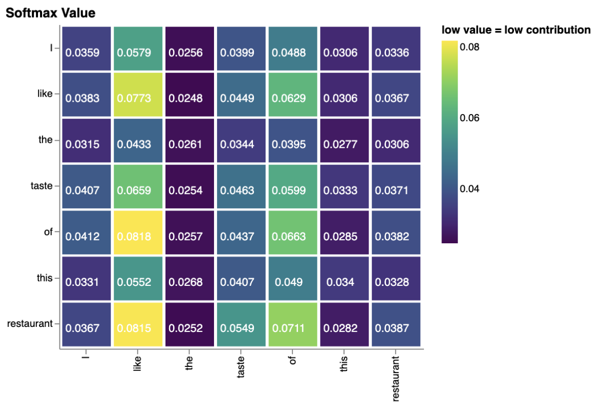


Figure 2. Softmax values calculation process



	I	like	the	taste	of	this	restaurant
I	0.0359	0.0579	0.0256	0.0399	0.0488	0.0306	0.0336
like	0.0383	0.0773	0.0248	0.0449	0.0629	0.0306	0.0367
the	0.0315	0.0433	0.0261	0.0344	0.0395	0.0277	0.0306
taste	0.0407	0.0659	0.0254	0.0463	0.0599	0.0333	0.0371
of	0.0412	0.0818	0.0257	0.0437	0.0663	0.0285	0.0382
this	0.0331	0.0552	0.0268	0.0407	0.049	0.034	0.0328
restaurant	0.0367	0.0815	0.0252	0.0549	0.0711	0.0282	0.0387

Figure 3. Comparison Matrix

Multi-line Chart

A multiple line graph visualizes the multiple set of data. Like single line graph, it shows the relationship between independent and dependent values. Usually the independent values are in time series so it can show the trends over time. It can also be categorical illustrating the values under different circumstances. Each data value is represented as a point and there is a line connected the values belong to the same set of data.

To better examine the softmax values, we also use the multi-line chart to show these values. Each line in the graph represent a series softmax values for a single word. For example, for sentence, "I like the taste of the restaurant", there will be a line connected the soft values of word "I" and other lines for other words. To distinguish each line, when user selects one line, the selected line will be blue and the other lines will be grey. Which word the selected line is corresponded to the line is also labeled beside the line. In a multi-line chart, the maximum and minimum for a line can be seen very clear. Thus, unlike comparison matrix which provides a value and a color, the shape of the multi-line chart can be more straightforward to illustrate how much focus is placed on other parts of the input sentence as we encode a word at a certain position.

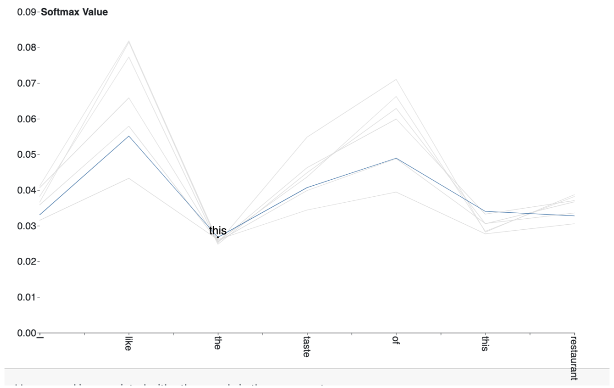


Figure 4. Multi-line Chart

3.4. Matrix collections

Mean / Max matrix

After training, we plot the Q, K, V matrices using Barcode plots. A Barcode Plot can show the distribution of values in a data set. The largest parameters in the matrix will be colored as darkest blue and the smallest parameters in the matrix will be colored as darkest orange. The color will be closer to orange if the value of the parameter is closer to the smallest value. The color will be closer to blue if the value of the parameter is closer to the largest value. For each matrix, we show the parameters in every row. In max/mean matrix, each cell is computed by taking the maximum from the five neighbors of the parameter or averaging the values of the five neighbors of the parameter. From this plot, we see how the large values and small values of parameters distributed.

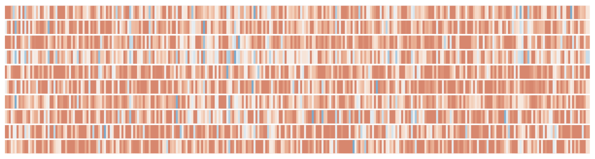


Figure 5. Q Matrix

PCA matrix

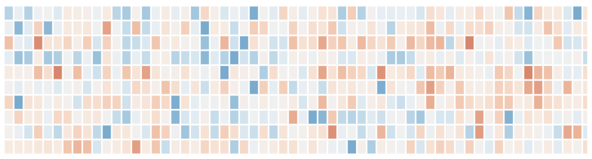


Figure 6. Q Matrix Max

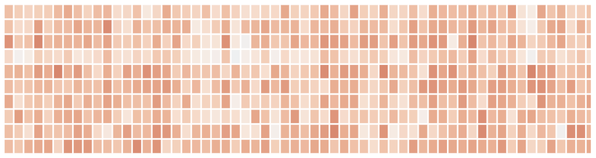


Figure 7. Q Matrix Mean

The word embedding matrix is $n \times 300$ where n is number of the words in the input sentence. To understand how similar each word embedding vector (a single row in the word embedding matrix) is with each other, we reduce the dimension of each word embedding vector. Each word embedding vector is 1×300. We use principle component analysis to reduce its dimension to 1×1. To see how similar each dimension is with other, we take each column in the word embedding matrix. Then, we also apply principle component analysis on each column. The dimension is reduced from $n \times 1$ to 1×1. After reducing the dimension, every vector can be treated as a data point. We plot these data points in the scatter plot.

The reduced results for word embedding vector are shown in Figure 8, and the reduced results for dimensions are shown in Figure 9.

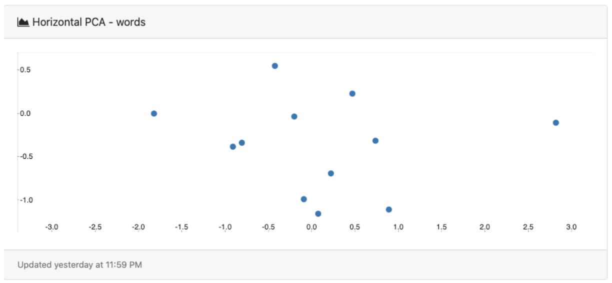


Figure 8. Horizontal PCA - words

4. Evaluation

In this section, two use cases are presented to show that Transformer-Vis is efficient for experts to interpret how attention layer works. We also demonstrate the feedback from some experts from related field.

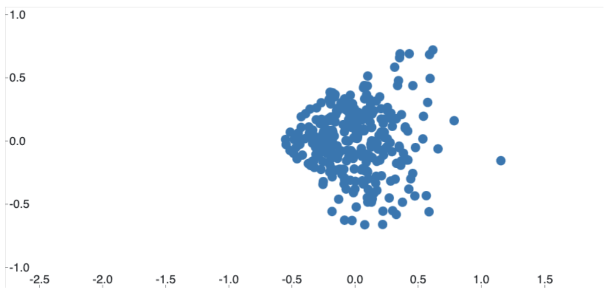


Figure 9. Vertical PCA - dimension

4.1. Case 1: Detecting Model Correctness

As discussed in Section 3.3, max and mean matrices can detect the outlier points. When a certain point in the matrix differs greatly from the surrounding points (the display in the visualization results shows that the color difference is greater), this point can be considered as a outlier point. Considering that the values of parameters in the converged model are usually continuous, the appearance of outlier points in two matrices implicitly says that the model may not converge completely. As a result, some wrong predictions may appear.

For instance, Figure 10 is the max Q-matrix for the input sentence:

- ***I like him, I hate her.***

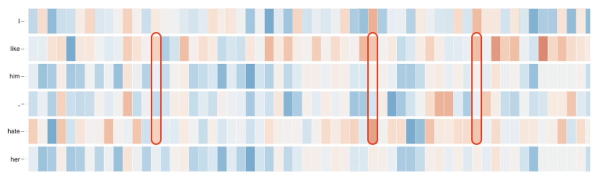


Figure 10. Max Q-matrix

The model's emotional judgment for this sentence is strong negative, which is incorrect. By observing the matrix, we find that the like and hate vectors are similar in some dimensions (such as the red circle in the figure), which does not conform to the usual logic, and it is also consistent with the final error prediction result.

4.2. Case 2: Model Compression

Model compression was firstly proposed by Han et al. [7] in 2016. Han et al. Three compression methods are introduced – Pruning, Quantization and Huffman coding (Optional). For the pruning operation, the method in the paper is to set some small (considering not important) weights to 0, so that the weight matrix is converted into a sparse matrix. He et al. [8] and Rastegari et al. [11] work both prove that pruning and quantization can indeed reduce the size of the model while maintaining accuracy.

Referring to the calculation method of mean matrix mentioned in the section 3.3, we can think of the lighter part of the matrix, that is, the part with a value close to 0. Then this part can be pruned when do model compression. We can use matrix to visually detect which vector and which dimensions are deleted, as shown by the red circle in the figure 11.

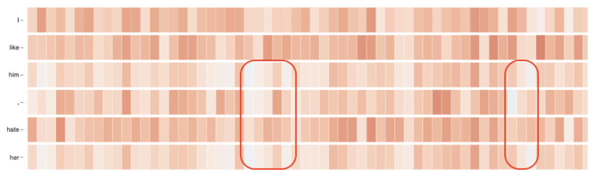


Figure 11. Mean Q-matrix

Vertical PCA - dimension, on the other hands, can help experts to find similar dimensions in word vector, so that when doing quantization after pruning, weights can be shared according to similarity, thereby further reducing space consumption and maintaining accuracy. Figure 12 shows three similar dimensions in the V matrix.

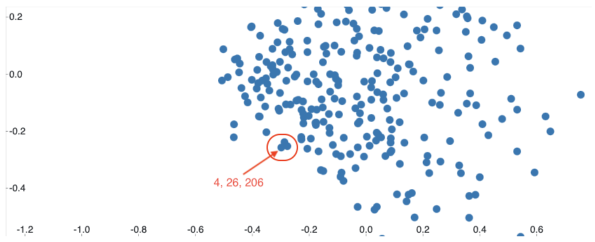


Figure 12. Vertical PCA - dimension

4.3. Expert Reviews

In order to figure out the advantages and disadvantages of our visualization tool, we invited two domain experts to evalutae our tool. E1 has published some articles in the field of neural network visualization. E2 is currently a PhD student in the field of NLP. We first introduced the usage of our tool and the meaning of each chart, and then showed the experts two cases mentioned in section 4.1 and section 4.2. Finally, we asked whether experts can better understand the working mechanism of attention after observing two cases.

Interpretation. After the experts watch the demonstrations of the two cases, they can accurately judge whether the model has converged. They believe that the result of visualization is a side criterion for the accuracy of the model. E1 also considered that given a visualization of similarity between dimensions and words is essential to understand how model compression works. E2 commented that by comparing the degree of word contribution, you can understand the relevance of words to a certain extent.

Suggestion. Some valuable suggestions are also provided. E1 suggested that we can add a model import interface to the front end to simplify the operation process. E2 mentioned that dimensionality reduction should be customized. E1 and E2 thought that we can further explore other tasks in NLP, such as language translation, and so on.

5. Conclusion

In this project, we offer a visualization tool to examine the calculation process and the internal structure of the transformer model with attention mechanism. With word cloud, we understand how each word in the input sentence associated with the word embedding vectors. With Barcode plot, we learn the distributions of parameters of Q, K, V matrices. With multi-line chart and comparison matrix, we can summarize how each word is associated with other words in the input sentence. From the scatter plot, we visualize the word embedding vectors after reducing their dimension through principle component analysis. We can tell how the representation for each word is different from each other in lower dimension.

Beyond the potential applications mentioned above, our visualization system can be used in detecting model correctness. Researchers can use the system to debug their transformer models. This system can also be applied on model compression. Researchers can learn which dimensions are redundant and can be reduced from the graphs.

However, the system still needs improvements. In future work, we will label the values aside the cell in the Barcode plot to let user can see the values of the parameters. The user interface will be designed to be more easily manipulated.

References

[1] B. Alsallakh, A. Hanbury, H. Hauser, S. Miksch, and A. Rauber. Visual methods for analyzing probabilistic classification data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1703–1712, 2014.

[2] A. Ambartsoumian and F. Popowich. Self-attention: A better building block for sentiment analysis neural network classifiers. *arXiv preprint arXiv:1812.07860*, 2018.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] J. Choo, H. Lee, J. Kihm, and H. Park. ivisclassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *2010 IEEE Symposium on Visual Analytics Science and Technology*, pages 27–34, 2010.

[5] M. Daniluk, T. Rocktäschel, J. Welbl, and S. Riedel. Frustratingly short attention spans in neural language modeling. *arXiv preprint arXiv:1702.04521*, 2017.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[8] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[9] S. Liu, Z. Li, T. Li, V. Srikumar, V. Pascucci, and P.-T. Bremer. Nlize: A perturbation-driven visual interrogation tool for analyzing and interpreting natural language inference models. *IEEE transactions on visualization and computer graphics*, 25(1):651–660, 2018.

[10] 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. IEEE, 2017.

[11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[12] 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.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. arxiv 2017. *arXiv preprint arXiv:1706.03762*.

[14] J. Vig. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*, 2019.