Knowledge Graph for Better NLP Model Explanations

Hyein Koo
Technical University of Munich
hyein.koo@tum.de

Anagha Radhakrishnan Moosad
Technical University of Munich
anagha.moosad@tum.de

Abstract

Natural Language Processing (NLP) models have made significant progress in recent years, achieving remarkable results in many NLP tasks such as text classifications. However, model explainability is still an active research area with scope for improvements. NLP model explanations are essential, especially for applications in which the information of how the model predicted a particular result is crucial, for example, in medical diagnosis. This project proposes a Multi-hop K-BERT model that leverages the knowledge graph to enrich the input data to create better model explanations.

1 Introduction

NLP models have made significant progress recently achieving good results, especially in language models like BERT (Devlin et al., 2019). However, as models provide more accurate results, the models become more complex, have more parameters and therefore have become difficult to interpret. Explaining this black box model has become crucial, especially in NLP application areas such as medical diagnosis (Rasmy et al., 2020) and legal decisions (Noguti et al., 2020). Currently, only input data can be leveraged to create better model explanations using existing explainability frameworks such as SHAP (Lundberg and Lee, 2017). Adding additional information related to input data could help provide better model explainability. One way to add information is by using knowledge graphs. A knowledge graph is a knowledge base in a graph data structure that represents real-world entities like objects and concepts and shows their relation (Hogan et al., 2021).

This project tries to create better explanations for text classification tasks using BERT model by combining input data and knowledge graph data. We also explore if more than one hop neighbourhood data from the knowledge graph would provide additional information for model explainability.

2 Related Works

Many works have been done to combine knowledge graphs and input data to create better model results and model explainability.

The paper by Annervaz et al. (2018) proposes a method to reduce labeled training data for deep learning models. The method first encodes the entity and relationships of the knowledge graph. They are then separately fed into LSTM and form context embeddings. The entity vector and relation vector are constructed by attention on each context embedding and a fact triplet is formed used for class prediction. This paper gives us an overall idea of how knowledge graph can be used better predictions.

Another paper, Generating Commonsense Explanation by Extracting Bridge Concepts from Reasoning Paths (Ji et al., 2020), tries to explain an input sentence by extracting bridge concepts from an extracted subgraph of a knowledge graph using GPT-2 language model. This is not used for classification task but for explaining the meaning of the input sentence.

In the paper K-BERT: Enabling Language Representation with Knowledge Graph (Liu et al., 2019), the input sentence is injected with knowledge graph data thus creating a sentence tree with one-hop neighbours of original word tokens. This sentence tree is passed into a pre-trained BERT model for better predictions in domain specific tasks.

Explainable Natural Language Processing with Knowledge Graphs (Poppel, 2021) introduces an extension of K-BERT that retrieves multiple hop neighbourhood data from knowledge graph using a recursive method to add additional information to the input sentence. This method follows the original K-BERT approach but the input data is contextually embedded first.

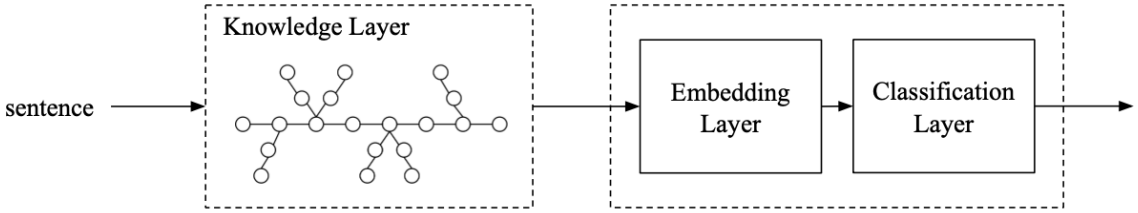

Figure 1: The architecture of Multi-hop K-BERT. The sentence tree is created in the knowledge layer. We have pre-trained BERT for the embedding layer and fully-connected layer for the classification layer. The output is the class of the sentence.

3 Model

Our Multi-hop K-BERT is inspired by K-BERT (Liu et al., 2019). K-BERT has knowledge layer which injects knowledge into sentences. We integrated multi-hop neighbourhood into the knowledge layer to further enrich the sentences and get better explanations.

3.1 Model Architecture

The model consists of three modules - knowledge layer, embedding layer and classification layer.

3.1.1 Knowledge Layer

Tokenization We first need to tokenize an input sentence to find neighbours of each token. Since the knowledge graph contains both single and multi-word entities, we used multi-word tokenizer from NLTK [1] to keep meaningful multi-word expressions and find their neighbours.

Part-of-speech Tagging To reduce the number of neighbours added to the sentence, we filter tokens by part-of-speech tags. Given an input sentence $S = \{w_0, w_1, ..., w_n\}$ we have corresponding part-of-speech tags $P = \{p_0, p_1, ..., p_n\}$. In the next step we add knowledge of w_i if corresponding part-of-speech tag p_i is a noun, i.e., 'NN', 'NNS', 'NNP', or 'NNPS'.

Sentence Tree Generation In this step we generate a sentence tree. The original sentence is the trunk of the sentence tree and new knowledge added to the original sentence is a branch. Generating sentence tree is done in a recursive way. In a knowledge graph we look for all triplets which have the noun token w_i as a subject. We select maximum m triplets $\{(w_i, r_{i1}, o_{i2}), ..., (w_i, r_{im}, o_{im})\}$ and expand branches by adding relation and object tuples to the original sentence.This creates a new branch in the sentence tree:

$$T = \{w_0, ..., w_i\{(r_{i1}, o_{i2}), ..., (r_{im}, o_{im})\}, ..., w_n\}$$

If the object o_{ik}, which is a neighbour of w_i, is also a noun, we again look for triplets which have o_{ik} as a subject and add the relation and object tuples to the branch. Now the sentence tree is:

$$\begin{aligned} T = \{w_0, ..., w_i\{(r_{i1}, o_{i2}), ..., \\ (r_{ik}, o_{ik}\{(r_{ik1}, o_{ik2}), ..., (r_{ikm}, o_{ikm})\}), \\ ..., (r_{im}, o_{im})\}, \\ ..., w_n\} \end{aligned}$$

We recursively expand the branch until it reaches the maximum depth d. If the number of tokens in the sentence tree is 512, which is the input size of the embedding layer, we stop adding knowledge. The final sentence tree is a knowledge-enriched sentence which has d hop m neighbours.

3.1.2 Embedding Layer

We create a sentence embedding of the sentence tree. We use BERT (Devlin et al., 2019) as the embedding layer of our model. BERT is a bidirectional transformer based model, which allows its embeddings to represent context as well.

Soft-positioning To create position embeddings, which are contextual representations, we need to give proper position indices. We keep absolute position indices $I = \{0, 1, ..., n\}$ for the original sentence in the sentence tree. For the neighbour token of w_i, we give $i + 1$ as the position index. The example sentence tree is in Figure 2.

3.1.3 Classification Layer

For classification we added one fully-connected layer. This layer outputs the prediction for the correct class of the input sentence.

[1] www.nltk.org/_modules/nltk/tokenize/mwe.html

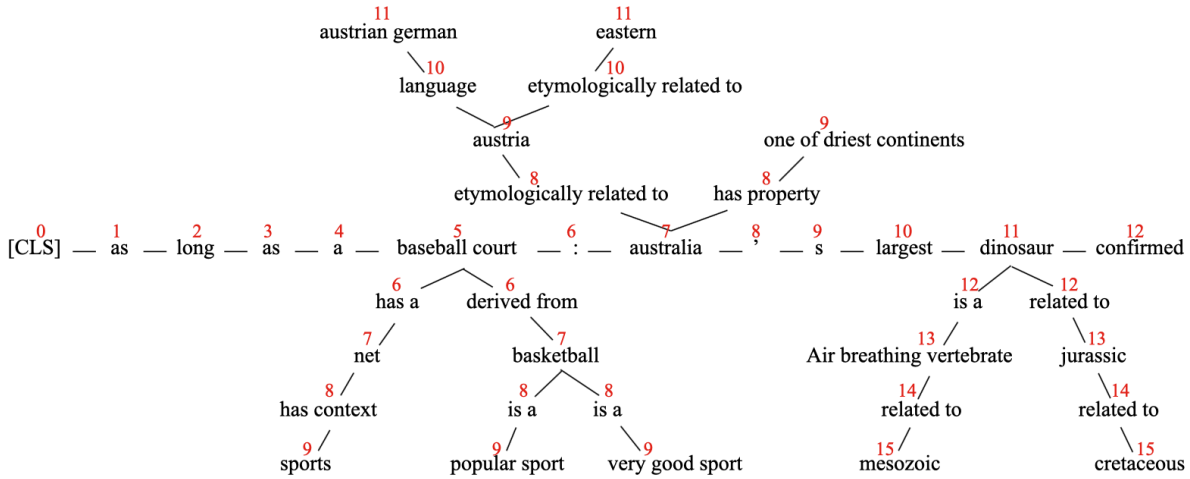

Figure 2: The example of the sentence tree created in the knowledge layer. Each token can have maximum 2 neighbours and the maximum depth is 2. Soft position indices are shown as red numbers on the tokens.

3.2 Loss Function

Since our task is multi-class classification, we used cross-entropy loss L:

$$L = -\log\left(\frac{\exp(x[\text{class}])}{\Sigma_j \exp(x[j])}\right)$$
$$= -x[\text{class}] + \log\left(\Sigma_j \exp(x[j])\right)$$

Hyper-parameter	#
Batch size	4
Learning rate	1e-5
# Epochs	10
Sequence length	512
Learning rate decay	0.1
# Epochs for learning rate decay	1

Table 1: Hyper-parameters for training

4 Experiment and Evaluation

4.1 Dataset

ConceptNet ConceptNet is a large-scale semantic network created by collection of information from many different sources (Speer et al., 2018). It represents general knowledge that explains the meanings and relations of entities. Each entry consists of a triplet $T = (s, r, o)$ that has a subject s, an object o and the relation r between them. We used more than 3 million entries that has English entities and relations. We excluded certain relations such as 'antonym', which don't add much information to the original sentence and used 47 relations out of 50.

AG's corpus of news articles For classification task we used AG's corpus of news articles[2]. It has more than 1 million news articles which are collected from more than 2000 news sources. It has 4 classes, i.e., world, sports, business and sci/tech, and each class contains 30,000 training samples and 1,900 testing samples. We used pre-defined train and test splits. The train set contains 120,000 documents and the test set contains 7,600 documents. We further split the train set randomly and used 80% as the train set and 20% as a validation set.

[2] groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

4.2 Training Setup

We set the number of neighbours $m = 2$ and the maximum depth $d = 2$, thus the sentence tree is a 2-hop 2-neighbour tree. Further we trained our model without additional knowledge, which we call Vanilla BERT, to compare the result.

We used the same hyper-parameters in Table 1 to train Multi-hop K-BERT and Vanilla BERT. Both models were trained for 10 epochs and we used learning rate decay by 0.1 every epoch. However, we trained Vanilla BERT for 5 epochs and then trained another 5 epochs from its checkpoint, hence the learning rate is set to initial learning rate at epoch 6. The models used Adam optimizer (Kingma and Ba, 2017) for optimization.

4.3 Evaluation

Our Multi-hop K-BERT achieved 90.21% of test accuracy. Compared to the Vanilla BERT which achieved 93.17%, the test accuracy of Multi-hop K-BERT is lower. This could be due to the loss of orig-

Model	Acc (%)
Multi-hop K-BERT	90.21
Vanilla BERT	93.17

Table 2: Test accuracy

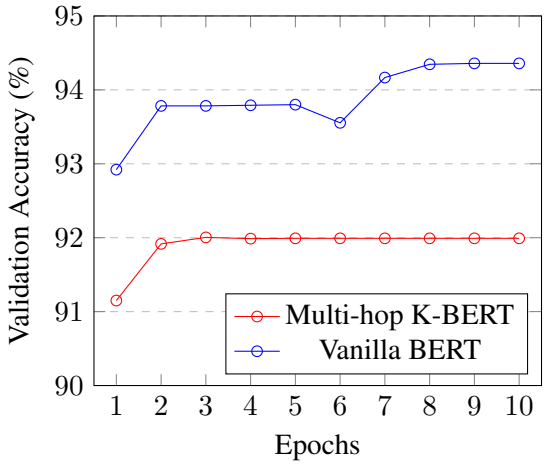

Figure 3: Validation accuracy over training epochs

inal tokens as many additional tokens were added from the knowledge graph. Moreover, there is a small difference in training setting, which might have affected the result. However, Multi-hop K-BERT still shows fairly decent performance.

5 Explainability

5.1 Explainability Framework

To explain the results of the proposed models, we use the Captum (Kokhlikyan et al., 2020), an open-source model explainability library built on Pytorch. It supports most Pytorch models. In addition to explaining the attributions, Captum also provides visualizations that help understand the explanations easier.

There are three attribution algorithm groups in Captum. Primary attribution estimates the contribution of each input feature to the model's output. Layer attribution assesses each neuron's contribution in a given layer to the model's output. Neuron attribution estimates the contribution of each input feature on the activation of a particular hidden neuron.

In this project, we use *Layer Integrated Gradient*, which falls under the layer attribution algorithms. Basically, *Integrated gradients* represent the integral of gradients with respect to inputs (Sundararajan et al., 2017). Integrated gradients along the ith dimension of input x is computed by:

$$\text{IG}_i = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

where x' is a baseline and α is the scaling coefficient. Layer integrated gradient represents the integral of gradients with respect to the layer inputs and outputs. Hence, by layer integrated gradient we can see the contribution of each bert embeddings, which are basically word tokens.

5.2 Results

Captum calculates the contribution scores of each BERT embeddings and assigns them to each token as attribution scores. Captum assigns green colors for tokens with positive attribution scores and red colors for tokens with negative attribution scores. Additionally, the more a token contributes to the prediction the darker the green and red colors are. Every word has a green or red color depending on the attribution score but not all the colors are visible because of less intensity of color shades.

Figure 4 shows an example of Captum visualization of a sci/tech news. As seen from Figure 4, the Multi-hop K-BERT model correctly predicted the class of the sentence. The result shows that words such as 'Microsoft', 'enterprise', 'operating', 'system' contribute more positively to the prediction while words such as 'sea', 'women', 'child' contribute negatively to the prediction.

In addition to Captum results, we also separated the words of the input sentence into two classes, original and knowledge graph, and further split the classes into positive and negative score words. Then the total attribution scores are calculated by the sum of attribution scores of tokens in each class. Table 3 show the total attribution scores for each class for the same sci/tech news in Figure 4. As seen, the knowledge graph data gives additional contributions to the class prediction. Table 4 shows which words have large attribution score and also whether the words are from the original sentence or from the knowledge graph.

Sentence Tree Visualization Since Captum does not give a good visualization of the sentence tree, we used NLTK.tree[3] which is a class in NLTK library that represent the hierarchical tree structures such as the syntax tree. We set positive upper threshold and negative lower threshold for the attribution scores to show tokens that contribute the

[3] www.nltk.org/_modules/nltk/tree.html

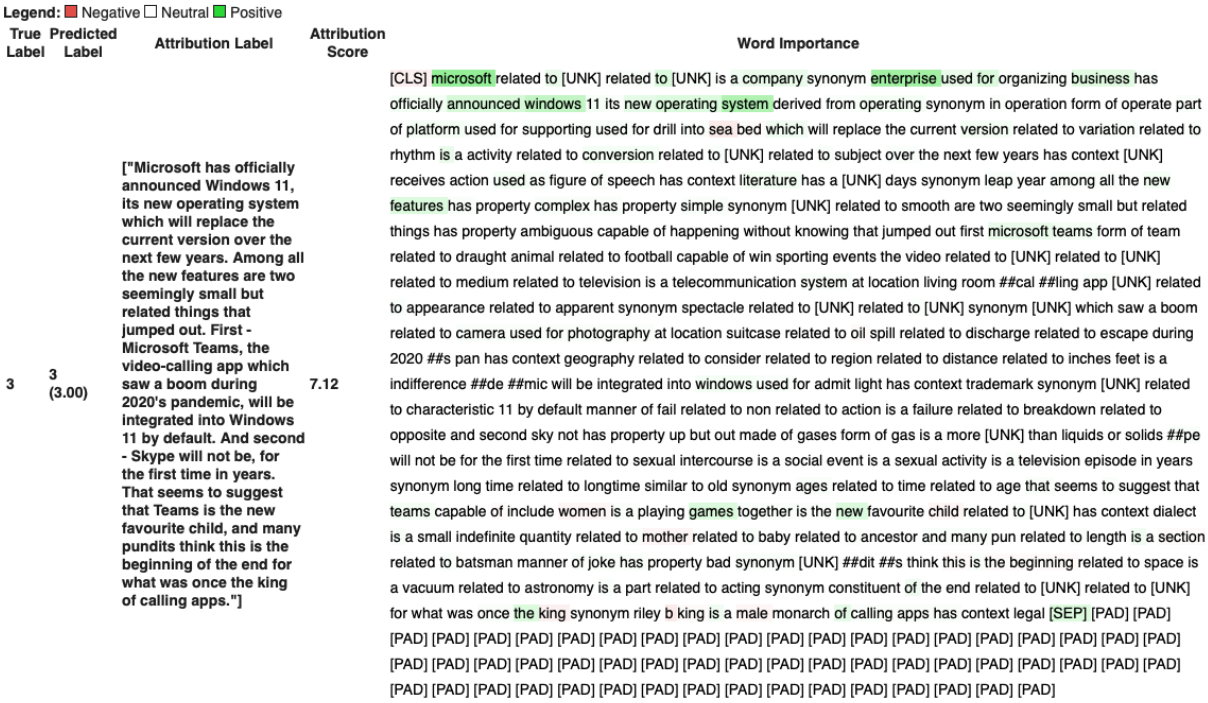

Figure 4: The example of Captum visualization for sci/tech news. The green color tokens have positive attribution scores and red color tokens have negative attribution scores. The total attribution score for the predicted class (3) is 7.12 which is higher compared to other predicted classes that are not shown here. Tokens such as Microsoft have darker green color compared to other tokens. This means that Microsoft token has a higher positive attribution score and therefore has positive influence on the class prediction.

	Original sentence	**Knowledge graph**
Positive	3.6458	4.8810
Negative	-0.4748	-0.9292

Table 3: The table shows the total word attribution score for each positive and negative tokens for both original and knowledge graph for sci/tech news in Figure 4.

	Original sentence	**Knowledge graph**
Positive	microsoft, windows, system	business, company, used
Negative	king	sea, women, mother, male

Table 4: The table shows the token distributions for original and knowledge graph for sci/tech news in Figure 4.

prediction the most. If an attribution score of a token is greater than the upper threshold, [+] sign is appended to the token and [-] sign is appended to the token that has smaller attribution score than the lower threshold. Figure 5 shows the sentence tree for the example sci/tech news. The figure does not show the entire sentence tree since it is large.

6 Conclusion & Future Work

We have introduced our new model Multi-hop K-BERT, which integrates knowledge graph to the input sentence. By adding multi-hop neighbours to the sentence it is enriched with common knowledge, thus providing better explanations of the result. Also, it is straightforward to interpret the explanation of the model since our model creates the sentence embedding after adding knowledge.

We used an open-source Pytorch explainability framework Captum to understand which tokens contribute to the class predictions. We also calculated the total positive and negative attribution scores for words from the original sentence and knowledge graph. We observed that additional information from the knowledge graph does help in understanding the prediction results of the model. Additionally, we also provide a method to visualize the constructed sentence tree.

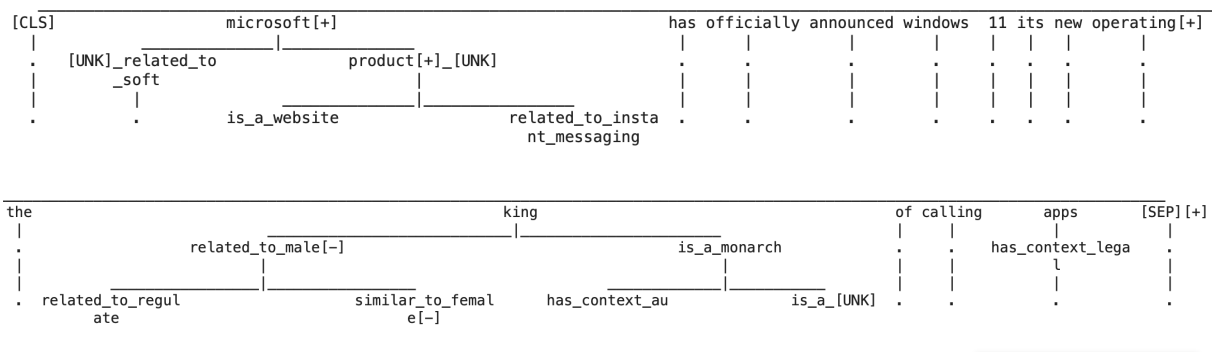

Figure 5: The figure shows a partial sentence tree of the sci/tech news in Figure 4. The structure shows sentence tree of depth 2. Tokens with [+] and [-] have positive and negative attribution scores respectively. Only the tokens with a greater or smaller than the thresholds have [+] or [-].

6.1 Limitations

There exist few limitations of our model. The first one comes from the limitation of BERT. The input size of pre-trained BERT is 512. If the input sentence is long or too many new tokens from knowledge graph are added to the sentence, the length of the sequence could exceed 512 and the tokens in the original sentence could be deleted. Thus we may lose essential information of the original sentence for the sake of common knowledge. The other limitation is in the knowledge layer. Our model randomly select neighbour tokens and add them to the sentence. Therefore, the sentence may have irrelevant knowledge to the original sentence. This is not a good behaviour if we expect better explainability of the model. This could also result in wrong prediction.

6.2 Future Work

We suggest one approach to solve the problem of knowledge layer selecting irrelevant neighbours. Instead of selecting neighbours randomly, we can use embeddings and compute cosine similarities. Given one token in the sentence we can compute all the embeddings of neighbour tokens and compute the cosine similarities of each embedding with the embedding of the original sentence or the word embedding of the given token. This approach is highly likely to find relevant neighbours of the original sentence.

However, this approach has some drawbacks as well. Since ConceptNet has many multi-word entities, we need to create multi-word embeddings. However, a pre-trained model with multi-word tokenizer and vocabulary file we used is currently not available and therefore we need to train the model from scratch.

We can also expect the training and inference of the model to be much slower. We can reduce some time by computing embeddings of all the word tokens in knowledge graph in advance and use these pre-computed embeddings for creating sentence tree, but we still need to compute sentence embeddings and their cosine similarities, which may cause overhead.

Last but not least, it is quite inefficient to discard sentence embeddings after finding relevant neighbours. To make use of embeddings we created in this step, it is better to use another approach that computes embeddings first and adds knowledge embeddings instead of K-BERT approach which creates sentence tree first and computes sentence embedding.

References

K M Annervaz, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. 2018. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, and et al. 2021. Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37.

Haozhe Ji, Pei Ke, Shaohan Huang, Furu Wei, and Minlie Huang. 2020. Generating commonsense explanation by extracting bridge concepts from reasoning paths.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2019. K-BERT: Enabling Language Representation with Knowledge Graph. *arXiv:1909.07606 [cs]*. ArXiv: 1909.07606.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions.

Mariana Y. Noguti, Eduardo Vellasques, and Luiz S. Oliveira. 2020. Legal document classification: An application to law area prediction of petitions to public prosecution service. *2020 International Joint Conference on Neural Networks (IJCNN)*.

Korbinian Poppel. 2021. Explainable natural language processing with knowledge graphs.

Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. 2020. Med-bert: pre-trained contextualized embeddings on large-scale structured electronic health records for disease prediction.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2018. ConceptNet 5.5: An open multilingual graph of general knowledge.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks.