**Finding Topical Similarity in Responsa Using Transformers**

*Jacob Khalili and Yehuda Snow*

*Advised under Prof. Jonathan Schler*

One of the cornerstones of modern halachic literature is responsa. Careful learning, analysis and understandings of these scholarly letters inform not only the recipient on how they are to act, but establish important halachic precedence and principles used to decide complex questions for future generations.

Imagine a world, where when you are learning helpful suggestions of sources to look at can help aid in your understanding; a world where it's simple and easy to find the opinions of other great poskim on similar issues and other cases where similar concepts are at play.

Using modern natural language processing (NLP) techniques, building such a tool on a large scale has only recently become possible. In 2018, Google Research published a state of the art pre-trained network for NLP, called Bidirectional Encoder Representations from Transformers, or BERT. BERT is an unsupervised machine learning model, which is used to create an encoding-vector which represents the texts. BERT’s bidirectionality is one of the key reasons for its success. When representing a word, BERT not only looks at the word itself, but the context of the word, in both directions (previous and future words).

In April of 2021, researchers at Bar Ilan University published AlephBERT, a model based on Google’s BERT, but trained on modern Hebrew literature. The model was trained on Twitter, Wikipedia, and OSCAR (a large dataset of multilingual data crawled from the internet.) The model was trained using Masked-Language Modeling (MLM), where every sentence fed to the model contained a blank word, which the model tried to predict; the weights of the model were optimized to predict the missing words correctly. This gave the model a sense of linguistic patterns of the language it was trained on.

Whether doing training or inference, the model requires a tokenizer to convert the words into a vector representation of the text. This vector representation is then fed into BERT to generate an encoding. When training, that encoding is then fed to a second half of the network, which predicts the missing word, followed by a loss calculation, and the updating of weights. When doing inference, the distance between the current paragraph’s vector representations, and all other vector representations is calculated, and the closest paragraphs are represented to the user.

After being trained in this fashion, the middle layer’s output is used as a vector encoding of the input text.

….

In this project we took AlephBERT and fine-tuned it based on the language specifically used in sheilos utshuvos. We did this fine-tuning via MLM on our dataset of responsa. Through this process, AlephBERT became accustomed to the dialect of Hebrew used specifically in responsa.

Before AlephBert was fine-tuned on responsa while it was familiar with the Hebrew language in general, it had never seen the language used in the way it was used in responsa and had not seen a lot of the words that would only come up in responsa. Therefore, the AlephBERT model had trouble generating embeddings that captured the meaning of the text, given that it was a different dialect of Hebrew. Being familiar with the dialect of responsa allowed the fine-tuned AlephBERT to generate similar embeddings for texts that were topically similar. Thus, we could compare the embeddings of one paragraph to another and would know how similar they are based on the distance between the two vectors. Now we could compare a paragraph to all the other paragraphs in responsa and find the paragraph that is the most topically similar to that paragraph.

In order to improve the results of our search to find other paragraphs similar to the original paragraph we also generated an embedding for each document. We did this by taking the first 250 tokens of the first paragraph in the document and the last 250 tokens of the last paragraph. The thought process behind this was that the main points in the document are expressed in the beginning when the question is asked and at the end in the conclusion. We were unable to include much more than 500 tokens as the max size of an embedding is 512 tokens. Then when we searched for similar paragraphs we compared both the two paragraph embeddings and the two document embeddings. We then summed the two scores together and divided by two. Our search results contained paragraphs that had the smallest average distance scores.

While our search results for topical similarity were promising, there is still more work to do. Sometimes our search results contained no paragraphs that were actually similar or just one or two similar results, even though there were actually many paragraphs that were similar. This could potentially be improved by doing more epochs of fine-tuning on our model or by doing some other combination of embeddings like how we did with making the full document embeddings.