CS584: US PATENT PHRASE TO PHRASE MATCHING

Jose Amezquita[1], Jon Amitai[1]
[1]Stevens Institute of Technology
jamezqu1@stevens.edu, jamitai@stevens.edu

ABSTRACT

Patents provide a way for people or companies to own exclusive rights to their products or ideas. Patents are a pivotal part of society that provide protection to inventions and their creators. However, there are millions of patents in the US alone. People filing new patents must first make sure that their idea does not already exist so this astronomical number makes it difficult to check. A way of checking similar patents could be done by checking similarity in key phrases in the patent. However, within different contexts phrases can have different similarities. This paper aims to detect language similarities in patents from the United States Patents and Trademarks Office (USPTO). Because of the subjectivity of this task, it is reasonable to use a language model in order to determine sentence similarity. More specifically, because the output is either a similarity metric or a probability distribution of them, a sequence to vector RNN is thus used. In addition to the words themselves, it is also crucial to look at the context in which the words are used. This is because certain words or phrases become much more or less semantically similar given context. Furthermore, we will not use a pre-trained BERT model built specifically for patent similarity detection because this is already available and thus we would like to attempt new methods.

1 Introduction

The US patent system grants an individual the right to any sort of process or machine that they have invented. It allows anyone to rightfully take credit for their hard work. However, because there are over 11 million patents, it can sometimes be difficult to ensure that every one of them are unique. Because the meaning of words depends on context, we look at that too. For example, while the phrases "bird" and "Cape Cod" may have low semantic similarity in normal language, the likeness of their meaning is much closer if considered in the context of "house" [4].In the context of a house, it is common, to find birds in backyards and inside houses, and Cape Cod is a location where someone may own a house, so in this context the semantic similarity is much more considerable than in everyday use. The context can be represented as various documents, a single word, or as a tag that indicates it. These different contexts could lead to different results and methods. In Semantic analysis, methods such as word2vec or Glove use a corpus of texts to identify word vectors and use cosine similarity to calculate the similarity if words. In the case where we only have a single token as the context methods such as a Neural Networks to predict the similarity given the token. The data used in this project uses a token to describe the context and thus various neural network architectures were used to predict the similarity.

2 Background

Two parts of the patent application process are the application process, and the examination process. In the application process, the person applying for the patent must research whether or not the patent already exist. Because the USPTO has been around for over 200 years, it is extremely difficult for the applicant to accurately determine whether or not the their idea already exists. A similar issue exists for the examination, where it is almost impossible for the USPTO to figure out whether or not the idea is new.

The idea of computing sentence similarity with context is not entirely new. In January 2022, a group of Chinese researchers used both probabilistic and generative models to predict sentence similarity given context [7]. Earlier than that, in 2019, a group of authors mostly from Beijing University used the concept of SAO (Subject-Action-Object) to determine the same result [8]. In their paper, they extract an SAO structure from each patent using WordNet [5] (a syntactical analysis tool), then use that structure to produce a similarity metric. Semantic similarity tasks have been done before, such as in methods like Glove [3], however this differs from those tasks in that similarity in this case is scored within the patent's context.

Additionally, in 2018, Google published a paper about BERT, or Bidirectional Encoding Representation from Transformers. Many of these models have found use in many fields of natural language processing such as text summarization, autocorrect, and machine translation. One of the many uses of these BERT models and their analogs is sentence similarity within a given context. This model can be applied to work with phrases instead of full sentences, and therefore be used in patent similarity.

The key to BERT is the fact that it is a bidirectional encoder. In unidirectional language models, each word can only gather context from previous words. However, this can be a very big limitation on the ability of a language model to perform tasks. On the General Language Understanding Evaluation (GLUE) benchmark [1], BERT scored seven points better on average than the state of the art model at the time, which was OpenAI's GPT.[2] GLUE is a set of twelve language tasks that are used to judge a language model's overall performance. To do well on the average score, a language model must have strong general knowledge of language, as opposed to only being designed for one specific use. Additionally, these models can be used straight out of the box and only require a little bit of fine tuning in order to be used for any specific task.

3 Data Set

The dataset used in our paper is taken from Kaggle. Each row contains an anchor phrase (the first phrase to be compared), a target phrase (the second one), and the context. There are hundreds of different types of contexts, ranging from furniture to houses. The contexts are the classification codes from the USPTO website [6]. For example, context A47 is FURNITURE; DOMESTIC ARTICLES OR APPLIANCES; COFFEE MILLS; SPICE MILLS; SUCTION CLEANERS IN GENERAL. The data provided has a training and testing data set. The training data has similarity values of the anchor and target, while the test data does not.

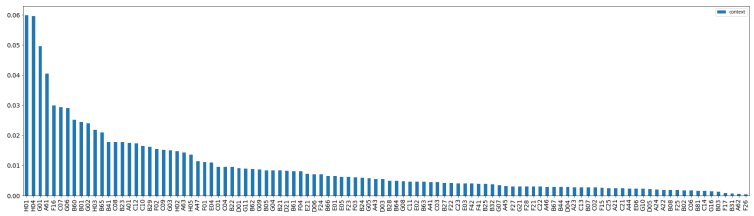
Table 1: Example Data

	Id	Anchor	Target	Context	Score
sample1	37d61fd2272659b1	abatement	abatement of pollution	A47	0.5
sample2	e955700dffd68624	acid absorption	absorption of acid	B08	1

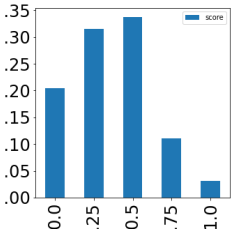
Table 1 shows some example samples of the data. Each data point has an id, anchor phrase, target phrase, context, and score. When processing the data we use the anchor, target, and context to predict the score. Futhermore, the score is either 0,0.25,0.5,0.75, or 1 so we can predict a the similarity as a class because there are only five options.

The difficulties in this dataset lie in the fact that the context is only a single token. Semantic analysis generally uses training texts to give context to a word based on the words around it, however in this case we are using just a single token for this task.

Figure 1 shows a distribution of the different contexts in the training set, providing insight into the most common and least common ones. As shown in figure 1, the most common contexts are H01 and H04 being electricity and electric communication technique. These both make up about 6 percent of the dataset each. Furthermore there are 106 unique contexts. Furthermore figure 1 also shows the distribution of the scores in the training data. Most scores have a similarity of 0 to 0.5, with very few having a similarity of 1.



(a) Distribution of the contexts in the training set



(b) Distribution of the scores in the training set

Figure 1: Training Data Distributions

4 Methods

Before even looking at the model architectures, we must first think about the output. The data is in the form of a classification problem, but we found that taking the weighted average of similarities produces a stronger correlation between predicted values and actual values as opposed to the row wise argmax and the corresponding probability to that result.

For example, if a model predicts the probabilities of each class of similarity being $[.2, .1, .3, 0, .4]$ respectively, then the weighted similarity would be computed as $[.2, .1, .3, 0, .4] * [0, .25, .5, .75.1] = .575$. However, when looking at accuracy, we will take the most likely class as predicted by the model. The correlation values using both techniques will be compared in this paper.

In order to solve the patent phrase to phrase matching problem we plan to look at a few different network architectures. First we will concatenate the anchor, target, and context into one phrase and create embeddings for each word in order to determine similarity. Additionally, we will use an encoder-decoder model, where encodings are created for each sentence and a decoder is used to predict the phrase similarity. We will also take a word frequency approach to predict similarity. This approach can do well because usually because if a word appears more than once between the anchor and the target, it is very likely that the phrases are similar. Lastly, we also have a model that in addition to looking at the anchor + context and target + context, also looks at the number of similar stems. This is a similar idea to the word frequency model, except word stems tend to catch similarities better when the words are different, but have the same root. In addition to the different models we also used an ensemble method that takes the max vote of basic model, the encoder-decoder model, and the concatenated context model in order to take the prediction that most models made.

In order to determine the best model we will use the provided test data to determine the models accuracy, as well as the correlation coefficient between the predicted similarity and the actual similarity. The output of the model will be both a continuous similarity value from 0 to 1, or one of five classes with scores from 0 to 1 in increments of $.25$.

When we were working on this task, we made the decision to not use any sort of pre-trained BERT model. This is because we would not really have much to add to them, as they are already extremely good. We wanted to create something on our own and contribute something new to this field of natural language processing.

5 Experimental Design

In order to evaluate our data we will use the Pearson correlation coefficient as outlined in the Kaggle competition. We will take the Pearson Correlation coefficient between the predicted similarity scores and the actual similarity scores. We intend to report our results from the given test set data and the Kaggle score that we get.

The Pearson Correlation Coefficient determines the correlation between two datasets. Datasets that are exaclty the same have a correlation of 1, while datasets with opposite results will have a correlation of -1. A correlation of 0 means that there is no correlation between the two datasets. The Pearson Correlation Coefficient is calculated as follows:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y}^2}$$

Furthermore we also calculated the accuracy of our models on the predictions. Accuracy is calculated as:

$$Accuracy = \frac{CorrectPredictions}{TotalPredictions}$$

For the loss function we used sparse categorical cross entropy. Cross Entropy is the standard loss used to optimize classification models. Sparse categorical cross entropy is used for models that are integer encoded. Cross Entropy is calculated as:

$$CE = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} log p(C_k|x)$$

where there are N data points, K classes, and p is the softmax probability

In total, we created and tested four different models in order to accomplish this task. They are the base word embeddings model, the BOW model, the stems model, and the encoder-decoder model.

Every single model in this paper was developed using TensorFlow 2.0, because it is quite simple to use, especially compared to the older version.

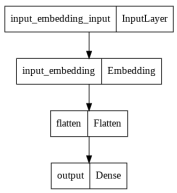


Figure 2: Base Model Architecture

The Base Model takes in the tokenized sentences of anchor + target + context, then feeds that into an embedding layer. It is then flattened, and put through a dense layer who's output represents a probability distribution over the possible similarity classes. Before the data is fed into the model, the sentences must be padded with zero tokens in order to ensure that all sentences are the same length. In order to prevent the loss of information, they are padded to the max sentence length. This can be a bit of an issue, as most sentences are less than 40 words, so many of the words in the input data are going to be mostly the $< end >$ token, which means that a lot of the data doesn't tell us anything new. The idea here is that it learns the proper word embeddings to capture the similarity between the two phrases.

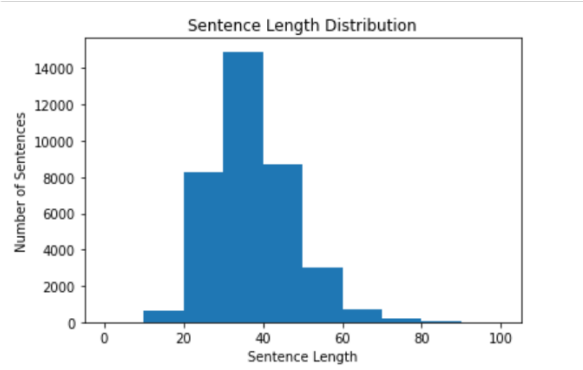


Figure 3: Sentence Length Distribution

Even with dropout, the model still heavily overfits to the training data, but it actually did better on test data without any dropout layer. As a result, none are present in it.

The word frequency model is based off the idea that if any word appears more than once in the sentence, it is very likely that there is similarity. Also, even if words used in the anchor and context aren't identical but similar, then the word frequency model could learn that certain sets of words appearing will be similar.

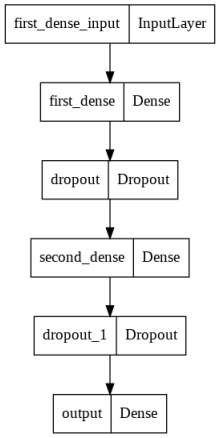


Figure 4: Bag of Words Model Architecture

The model takes in the word count array of each sentence as input, then runs it through two dense layers, two dropout layers, and then an output layer. The layers get narrower as the input feeds forward, which reduces the models size.

As it turns out, the number of parameters in this model is actually less than in the previous embedding model. As such, this model is pretty quick to train and takes up less memory.

Another advantage of this model is that the input sizes are constant for every sample. No matter whether a sentence has five words or fifty, their word frequency arrays are the same size, because their dimension will just be the vocabulary size.

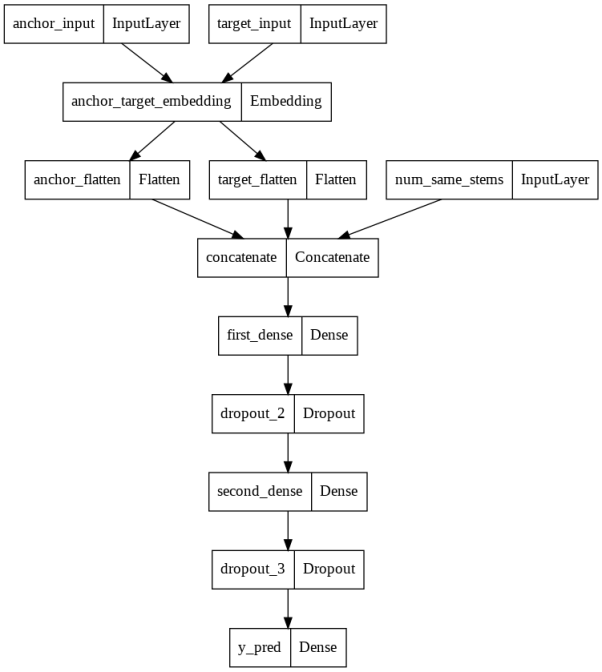


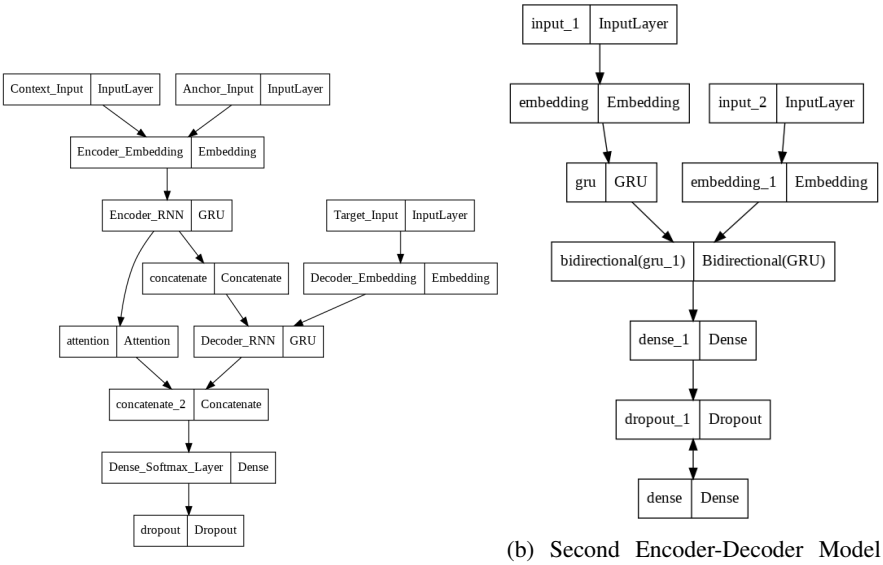
Figure 5: Number of Stems Model Architecture

One of the main issues with the BOW model is that in practice, the model fails to learn what words are similar to each other. For example, if the words ”abate” and ”abation” are in the anchor and target respectively, then the BOW model will fail to see how close the two are, even though they come from the same root. This is the idea behind the number

of stems model.

Firstly, the data that is embedded is different from the base model. Instead of anchor + target + context, there are now three inputs. The first is the anchor + context, the second is target + context, and the last is the number of duplicate stems present in the anchor and context. The idea behind splitting it into three inputs is that hopefully the anchor + context max length and the target + context max length are different. If this is the case, then there would be fewer padding zeros when preprocessed, which removes noise. Additionally, the number of duplicate stems in each example is also important as a measure of similarity because similar words tend to come from the same root, so a higher value would mean a higher similarity.

One issue with this is that it takes quite a while to train. Even for only 10 epochs, the model took about 30 minutes to train, and that is with a pretty small embedding size. Additionally, there are still many synonyms where the words do not come from the same root, and the model struggles to classify such data points.



(a) First Encoder-Decoder Model Architecture

(b) Second Encoder-Decoder Model Architecture

Figure 6: Encoder-Decoder Model Architectures

For the encoder-decoder model we based it off of the idea of seq2seq models. Seq2seq models can be thought of two models, and encoder and a decoder. The encoder takes in some data and uses and RNN layer, or variation such as LSTM or GRU, to create a context vector which is used as the initial state for the decoder. The decoder uses this context vector as the initial state to train some data on and create a sequence. For example, in machine translation the encoder encodes one language and the decoder predicts each word in the sequence for another language. For our encoder-decoder model we first tried taking in the context vector and the anchor to the encoder to create the context vector by embedding the context and anchor separately and concatenating the embeddings together. We then put them into a GRU layer to create the embedding. We also added an attention layer. The Decoder would take in the target phrase and use the context vector as the initial state for a GRU cell which would then get passed to a dense layer using the softmax function to create a probability distribution of each of the classes. We would then take the argmax of that output which would be the predicted class. This method showed poor results and overfit the data despite dropout.

Due to the poor results of the First encoder-decoer model we tried to change the architecture so that the encoder took in only the context and thus the context vector was made up of only the context token. The encoder in this case embedded the input and used a GRU layer to create the context vector. The encoder then took in the anchor and target concatenated together. The decoder used a Bidirectional GRU layer in order to more fully capture the two phrases. We did not feel the need to use attention on this model since the context vector was made up of only one token. This model performed much better than the previous architecture. Both the encoder-decoder models create a vocabulary for the data and use that to tokenize the inputs.

The ensemble method takes in predictions of the best 3 performing models and does a max vote on the three predictions, taking the prediction with the most votes. In the case of a tie, it takes the prediction from the stem model as that was the best performing one.

6 Experimental Results

Out of the five models tested, the number of stems model performed the best in both correlation between the predicted similarity and actual similarity, where the similarities are predicted using argmax or a weighted sum of similarities as outlined earlier.

In the table below, C_{max} is the correlation found by predicting the probabilities to be one of four classes, and C_{WS} is the correlation found by using a weighted sum.

Table 2: Results

	Accuracy	C_{max}	C_{WS}
Base Model	49.43%	0.39	0.46
BOW Model	46.24%	0.31	0.38
Num Stems Model	57.19%	0.52	0.58
First Encoder Model	33.12%	0.1269	0.14
Second Encoder Model	50.69%	0.42	0.44
Ensemble Method*	70.06%	0.68	N/A

With all five models that we have created, the hard argmax predictions and labels are less correlated than the predictions made with the weighted sum. This is because the hard argmax prediction method can choose an example to be a class with 30% probability, and that will on average be more wrong than taking the expected value over the models predictions.

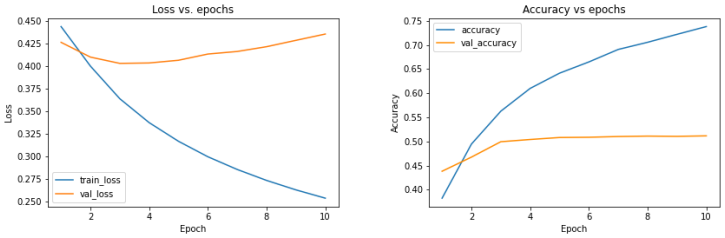


Figure 7: Base Model Graphs

The base model did fairly well, especially for how simple of an idea it is. almost 50% accuracy is great when you consider that even the most common similarity class only appears about 32% of the time. In fact, most of these models performs statistically better than picking the most common class.

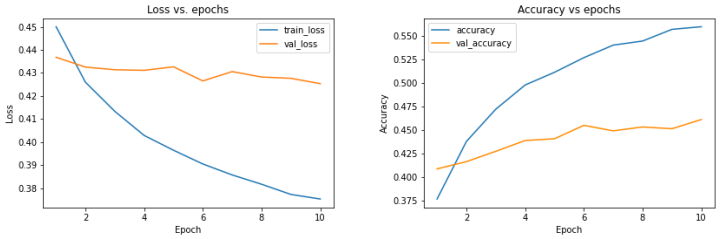


Figure 8: Bag of Words Model Graphs

It seems like the BOW model had a very difficult time learning sentence similarities. This is because the data is very sparse, as shown by the sentence distribution. Because most sentences are less than 40 words, then at most 40 elements in the vocabulary frequency array will be non-zero, which is extremely sparse. As a result, it performs quite poorly.

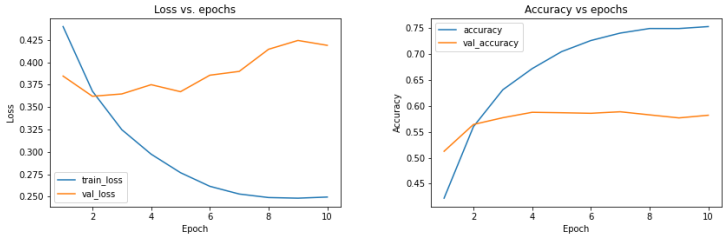


Figure 9: Num Stems Model Graphs

Out of all the models, the number of stems model performs the best on all three metrics. It would seem that looking at similar stems really captures the parts of language that account for phrase similarities. This is also the only model that we created with an accuracy above 55%.

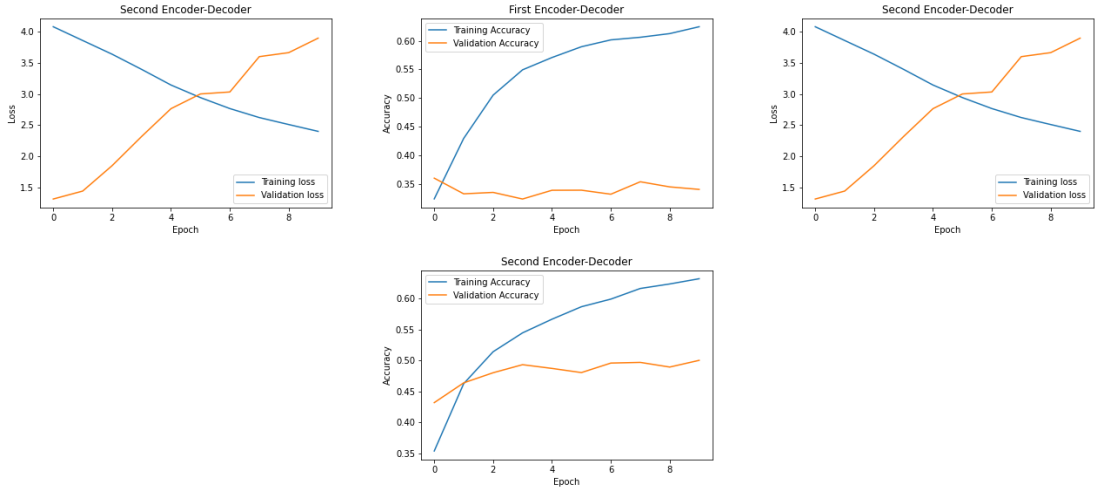


Figure 10: Encoder Decoder Model Graphs

The first encoder-decoder model performed poorly, it was the worst of all five of the models. It was having problems overfitting and never achieved an accuracy greater than around 30% on the validation data. The second encoder-decoder however performed much better. The results still left something to be desired however it was a large improvement to the first attempt. The training still had problems with the data overfitting and the validation accuracy plateauing around 49%. Both model also hat problems with the validation loss increasing throughout training instead of decreasing.

The ensemble method performs the best however it is important to note that the since we split up the work the models were trained on different subsets of the data, so this number is skewed to be higher since it is likely that some of the models were trained on some of the test data that was used at that instance.

The second encoder-decoder model performed quite well compared to the other models. It seems that using an encoder on the context created a decent initial state for training, resulting in test accuracy above 50% as well.

Submitting on kaggle using the stem model as that is our best performing one, we get a score 0.366. This is the correlation coefficient, as that is what kaggle uses to score it. This result is a dropoff from the local test data that we used. This could be due to overfitting once again.

7 Conclusion and Future Works

We were successfully able to build multiple models to classify sentence similarity within a given context, despite facing challenges that many language models do not. Although these models all performed pretty well, there are still some improvements to be made.

Every single one of the models is heavily overfitting. There are differences of up to 30% between the accuracy on training data vs testing data. Even with adding multiple dropout layers, the problem still remains. If we had more time, we would look into other methods of preprocessing and model selection in order to get a better accuracy and correlation on the test set.

The BOW model performed quite poorly, but it would be interesting to see how certain actions such as removing stopwords from the data would have on the metrics. If we had more time, we would look towards potential solutions like that to improve model results.

Even though the Number of Stems model did quite well, There could be potential for improvement if the number of similar stems was looked at only *between* the anchor and target as opposed to all the words within the two of them. Thus if the anchor has two identical words, it won't count as the same stem, because that stem does not appear in the target.

Both the second encoder-decoder model and the stem model perform with accuracy over 50%. In the future it may be worth investigating a combination of these two architectures, using an encoder and decoder that also incorporates stem counts in the decoder.

Additionally , an ensemble classifier could produce better results than any of these individual models can. If it is the case that certain models can classify certain types of phrases more accurately, then combining the strengths of the five individual models could create a classifier that has an accuracy of over 60%.

Being able to predict sentence similarity within a context will be a major help to all the future inventors who might be unable to check for the uniqueness of their ideas. It will allow people to pursue ideas without them having to worry about violating intellectual property. With the amount of data that the USPTO has, spanning over 100 years, some truly amazing things can be done to speed up the process of progress.

References

[1] Julian Michael Alex Wang Amanpreet Singh. "GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTANDING". In: (2019).

[2] Kenton Lee Jacob Devlin Ming-Wei Chang. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (2018).

[3] Christopher D. Manning Jeffrey Pennington Richard Socher. "GloVe: Global Vectors for Word Representation". In: (2014), pp. 1–12.

[4] Kaggle. *U.S. Patent Phrase to Phrase Matching.* 2022. URL: `https://www.kaggle.com/competitions/us-patent-phrase-to-phrase-matching` (visited on 04/05/2022).

[5] George A. Miller. "WordNet: A Lexical Database for English". In: *COMMUNICATIONS OF THE ACM* 38.11 (1995), pp. 39–41.

[6] USPTO. *United States Patent and Trademark Office.* 2022. URL: `https://www.uspto.gov/` (visited on 04/05/2022).

[7] Xiang Ao Xiaofei Sun Yuxian Meng. "Sentence Similarity Based on Contexts". In: (2022).

[8] Ying Huang Xuefeng Wang Yali Qiao. "Measuring patent similarity with SAO semantic analysis". In: *Scientometrics* 121.5 (2019), pp. 2–24.