U.S. Patent Phrase to Phrase Matching

Kowshika Sarker
ksarker2@illinois.edu
Department of Computer Science, University of Illinois at Urbana-Champaign
Urbana-Champaign, Illinois, USA

Chiranjeevi Konduru
konduru4@illinois.edu
Department of Statistics, University of Illinois at Urbana Champaign
Urbana-Champaign, Illinois, USA

Pradyumna Achyutharao Bada
pbada2@illinois.edu
Department of ISE, University of Illinois at Urbana Champaign
Urbana-Champaign, Illinois, USA

ACM Reference Format:
Kowshika Sarker, Chiranjeevi Konduru, and Pradyumna Achyutharao Bada. 2023. U.S. Patent Phrase to Phrase Matching. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We are working on identifying similarities between patent phrases to help patent professionals find relevant information from millions of patent documents by comparing key phrases and calculating the degree of similarity between them. This model will be trained on a semantic similarity dataset and will go beyond identifying paraphrases to recognize different phrases that refer to the same concept. An additional feature *context* will be used to understand the similarity across domains. The dataset has three features namely *anchor*, *target*, and *context*. The variables *anchor* and *target* have text data that we need to compare and the *context* has categorical data. This is a 5-class classification problem with classes 0 (unrelated), 0.25 (somewhat related), 0.5 (synonyms with different meanings), 0.75 (close synonym), and 1 (very close match). It gives us the similarity measure between anchor and target phrases. With the right data and model architecture, this tool can provide valuable assistance to patent professionals in navigating through patent documents.

2 DATA PREPROCESSING

2.1 Text embedding

Our task involves text data in the *anchor* and *target* input features. We use pre-trained language models to convert texts into numerical features. We initially planned to use BERT [2], a widely used transformer-based model capable of producing highly useful numerical embeddings for natural language. As different domains may have specific interpretations of words or phrases and particular patterns of sentences or words, we used a pre-trained model PatentSBERTa [1] specifically trained on patent documents fine-tuned for patent classification and distance calculation. The pre-trained model is publicly available and accessible via a Python package *sentence-transformers*. The PatentSBERTa model takes a text from the patent document as input and produces a 768-dimensional real-valued numerical vector as the semantic embedding of the text. We pass anchor and target texts separately to the PatentSBERTa model, which gives us two 768-dimensional vectors - so produces 1536 numerical features.

2.2 Categorical context

Our dataset involves a categorical variable *context* which denotes a category of the patent as per Cooperative Patent Classification. There are 106 unique contexts in our dataset. We apply one-hot-encoding on the context variable, which produces 106 binary input features.

So, in total, we have 1642 input features, of which 1536 are real-valued and 106 are binary.

2.3 Data size

The Kaggle competition provides a training dataset with 36473 samples and a test dataset with 36 samples. But the test dataset does not have ground truth and is too small in size. We divide the Kaggle train dataset into three random splits - train with 26351 samples, validation with 4651 samples, and test with 5471 samples.

3 BASELINE MODELS

We tried two baseline models on the preprocessed data. Both these models were conventional classifiers *Logistic Regression Classifier* and *Randomforest Classifier*. Initial Modeling was done by manually selecting the hyper-parameter values, later we used Bayesian optimization to get the optimal hyper-parameter values. However, these approaches didn't give the best results. The Models are discussed in detail below.

3.1 Logistic Regression

We attempted to use logistic regression as our initial baseline model on a dataset with 1642 dimensions, as it is known to perform well in high-dimensional spaces. However, after fitting the model, we found that the training error was very high at 42.87%, and the test error was even higher at 52.7%. The hyper-parameter tuning did not help much and because of high training error, we came to the conclusion that the data is not linearly separable. We felt polynomial methods would be infeasible as it increases the dimension of our data exponentially.

3.2 Random Forest

Randomforest is one of the prominently used ensemble models for classification tasks. Since the patent phrase matching is a multi-class classification problem, we tried with *"ovr" (One vs Rest)* approach.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.2.1 Random Forest without Hyper-parameter tuning: At first, we trained a random forest model with default hyperparameters and observed a training error of 0%, indicating overfitting, as the test error was 45%. To address this issue, we manually reduced the tree depth and n estimators to decrease the model complexity. However, we found that the model continued to overfit until we reached a maximum tree depth of 15, beyond which the training error increased. Unfortunately, we did not observe any improvement in the test error even after reducing the complexity of the model.

3.2.2 Random Forest with Hyper-parameter tuning: Since the results with manual tweaking of the hyper-parameters didn't improve the result much, we proceeded to perform Bayesian optimization to get the optimal values for these parameters. The reason for using this technique compared to RandpmSearch or GridSearch was the computational efficiency and also getting the best parameter values between a certain range rather than fixed hard-coded values with multiple combinations. The scoring function used to evaluate this optimization is *f1-micro*. In the multi-class case, this is the average of the F1 score of each class with weighting depending on the average parameter (*micro*). This calculates the metrics globally by counting the total true positives, false negatives, and false positives. The following are the parameters chosen for hyper-parameter tuning:

- `n_estimators`: No. of trees. Range - (low=100, high=1000, step=100). Optimal value: 1000
- `criterion`: The function to measure the quality of each split. Tested with *gini* and *entropy*. Optimal criteria: *gini*
- `max_depth`: The maximum depth of the tree. Range - (low=5, high=100, step=1). Optimal value: 52
- `min_samples_split`: The minimum number of samples required to split an internal node. Range - (low=2, high=4, step=1). Optimal value: 3
- `min_samples_leaf`: The minimum number of samples required to be at a leaf node. Range - (low=1, high=5, step=1). Optimal value: 2
- `max_features`: The number of features to consider when looking for the best split. Values evaluated : *auto* , *sqrt*, *log2*. Optimal criteria: *auto*

After getting the optimal values for the parameters, the model was used to predict the class labels on validation and Test datasets. The values of the metrics are as follows:

Table 1: Random forest performance validation metrics

Evaluation Metric	Score
`Accuracy (class - 1,2,3,4,5)`	0.85, 0.73, 0.70, 0.90, 0.97
`Precision (class - 1,2,3,4,5)`	0.80, 0.56, 0.53, 0.73, 0.78
`Recall (class - 1,2,3,4,5)`	0.37, 0.71,0.74, 0.19, 0.23
`F1-Score (class - 1,2,3,4,5)`	0.51, 0.62, 0.62, 0.30, 0.36

All the values above are computed using *One Vs Rest* strategy. Clearly, we can see that in multi-class classification accuracy is not the right measure to look at. But these per-class evaluation metric values helped us to identify the flaws in this model precisely. It is evident that *Recall* values are pretty low for 3 out 5 classes, which indicates that the model is unable to detect the true label in the majority of the cases.

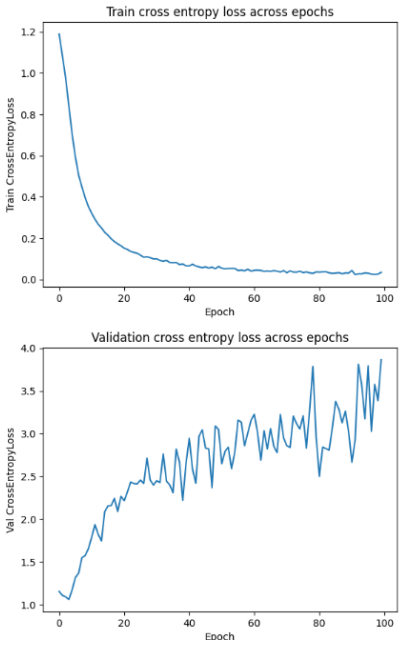

Figure 1: Training and validation cross entropy loss across training epochs of main model

4 PROPOSED MODEL

We propose to use an artificial neural network (ANN) as a five-class classifier for our task. We have experimented with two initial versions of our model so far.

4.1 Proposed model version 1

This is an ANN with 1642 input features, 500 hidden features and 5 output features, with ReLU activation function on the hidden layer. The output neurons are expected to produce logits of probabilities of a data point belonging to each of the five classes. We train the model on our train set by optimizing cross-entropy loss with Adam optimizer. The results and observations of this model were presented in the mid-term progress report.

4.2 Proposed model version 2

We take the Hadamard product of the anchor and target embeddings, which gives us 768 features instead of 1536 embedding features. The Hadamard product also captures the similarity of the embeddings. Now we use these 768 features with 106 one-hot-encoded context features to train an ANN with input, and output layers of size 874, and 5. We have added 5 hidden layers of sizes [400, 600, 800, 500, 200] respectively. Figure 1 shows the cross entropy loss of train and validation datasets across training epochs.

Figure 2 shows the evaluation metrics for train, validation, and test datasets.

Train evaluation

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5342
1	1.00	0.99	1.00	8324
2	0.99	0.99	0.99	8912
3	0.99	0.99	0.99	2912
4	0.98	0.98	0.98	861
accuracy			0.99	26351
macro avg	0.99	0.99	0.99	26351
weighted avg	0.99	0.99	0.99	26351

Validation evaluation

	precision	recall	f1-score	support
0	0.64	0.49	0.56	1005
1	0.62	0.67	0.64	1453
2	0.61	0.66	0.63	1559
3	0.47	0.44	0.45	498
4	0.55	0.57	0.56	136
accuracy			0.60	4651
macro avg	0.58	0.57	0.57	4651
weighted avg	0.60	0.60	0.60	4651

Test evaluation

	precision	recall	f1-score	support
0	0.65	0.50	0.56	1124
1	0.63	0.68	0.66	1742
2	0.62	0.67	0.64	1829
3	0.45	0.42	0.43	619
4	0.54	0.64	0.59	157
accuracy			0.61	5471
macro avg	0.58	0.58	0.58	5471
weighted avg	0.61	0.61	0.61	5471

Figure 2: Evaluation metrics of main model

4.3 Proposed model version 3

We are trying 2 things in this approach. Firstly, we are using CPC keywords instead of a categorical variable as context which could give more accurate embeddings. We mapped the keywords from CPC website (https://www.uspto.gov/web/patents/classification/cpc/html/cpc.html). Secondly, instead of getting separate embeddings for target, anchor, and context, we concatenated all three features and got embeddings in one shot. This ANN has 768 input features, 3 hidden layers of sizes [1500,1500,500] respectively. Figure 3 shows the evaluation metrics for train, validation, and test datasets.

5 ABLATION STUDY

We have performed 2 types of ablation study on the main model (ANN - 6 layers).

5.1 Ablation Study 1: Dropping the layers

In the first ablation study, the no. of layers in the network were dropped to half, which is 3 now and has only one hidden layer. We couldn't observe any major change in the validation accuracy but the test accuracy reduced a bit to 0.60. See Figure 4

Validation evaluation

	precision	recall	f1-score	support
0	0.55	0.38	0.45	1005
1	0.55	0.58	0.56	1453
2	0.50	0.59	0.54	1559
3	0.32	0.29	0.30	498
4	0.21	0.26	0.23	136
accuracy			0.50	4651
macro avg	0.43	0.42	0.42	4651
weighted avg	0.50	0.50	0.49	4651

Test evaluation

	precision	recall	f1-score	support
0	0.54	0.41	0.47	1124
1	0.56	0.57	0.57	1742
2	0.50	0.57	0.53	1829
3	0.34	0.30	0.32	619
4	0.20	0.27	0.23	157
accuracy			0.50	5471
macro avg	0.43	0.42	0.42	5471
weighted avg	0.50	0.50	0.50	5471

Figure 3: Evaluation metrics of main model version 3

Validation evaluation

	precision	recall	f1-score	support
0	0.56	0.58	0.57	1005
1	0.65	0.61	0.63	1453
2	0.59	0.65	0.62	1559
3	0.48	0.41	0.44	498
4	0.59	0.42	0.49	136
accuracy			0.59	4651
macro avg	0.57	0.53	0.55	4651
weighted avg	0.59	0.59	0.59	4651

Test evaluation

	precision	recall	f1-score	support
0	0.60	0.59	0.60	1124
1	0.64	0.61	0.62	1742
2	0.61	0.66	0.64	1829
3	0.48	0.45	0.47	619
4	0.61	0.53	0.56	157
accuracy			0.60	5471
macro avg	0.59	0.57	0.58	5471
weighted avg	0.60	0.60	0.60	5471

Figure 4: Evaluation metrics of Ablation model - 1

5.2 Ablation Study 2: RoBERTa Embeddings

In this study, we used a pre-trained RoBERTa model to create the word embeddings for the patent data. The base version of RoBERTa was used since the network was build to take input size of 874. The performance was not good compared to PatentSBERTa since, since the former model was trained on general data and not specifically for patent data. Hence the embedding vectors vary a lot. The performance of can be seen in Figure 5.

6 PARAMETER STUDY

We tuned hypermeters on our main model with different batch sizes (32, 64), learning rates (0.1, 0.01, 0.001) and optimizers (Adam and SGD) and picked the best hyperparameter setting based on validation accuracy. Following are validation accuracy with different hyperparameter settings. The best hyperparameter setting was

Validation evaluation

	precision	recall	f1-score	support
0	0.54	0.42	0.48	1005
1	0.60	0.60	0.60	1453
2	0.55	0.62	0.58	1559
3	0.45	0.44	0.44	498
4	0.45	0.61	0.52	136
accuracy			0.55	4651
macro avg	0.52	0.54	0.52	4651
weighted avg	0.55	0.55	0.55	4651

Test evaluation

	precision	recall	f1-score	support
0	0.53	0.43	0.48	1124
1	0.61	0.62	0.62	1742
2	0.56	0.61	0.58	1829
3	0.46	0.42	0.44	619
4	0.46	0.60	0.52	157
accuracy			0.56	5471
macro avg	0.52	0.54	0.53	5471
weighted avg	0.56	0.56	0.55	5471

Figure 5: Evaluation metrics of Ablation model - 2

Train evaluation

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5342
1	1.00	1.00	1.00	8324
2	1.00	1.00	1.00	8912
3	1.00	1.00	1.00	2912
4	1.00	1.00	1.00	861
accuracy			1.00	26351
macro avg	1.00	1.00	1.00	26351
weighted avg	1.00	1.00	1.00	26351

Validation evaluation

	precision	recall	f1-score	support
0	0.63	0.57	0.60	1005
1	0.64	0.67	0.66	1453
2	0.62	0.63	0.62	1559
3	0.48	0.49	0.48	498
4	0.63	0.57	0.60	136
accuracy			0.61	4651
macro avg	0.60	0.58	0.59	4651
weighted avg	0.61	0.61	0.61	4651

Test evaluation

	precision	recall	f1-score	support
0	0.63	0.60	0.62	1124
1	0.65	0.66	0.66	1742
2	0.64	0.65	0.65	1829
3	0.48	0.48	0.48	619
4	0.58	0.62	0.60	157
accuracy			0.62	5471
macro avg	0.60	0.60	0.60	5471
weighted avg	0.62	0.62	0.62	5471

Figure 6: Evaluation metrics of best hyperparameter

batch size of 64, learning rate of 0.001, and optimizer SGD. Figure 6 shows results with the best hyperparameters.

7 IMPLICATIONS

This work presented a pipeline of matching phrases found in patent documents so that new patent documents can be matched to existing resolved patents. We achieve excellent training and moderate validation and test performances with our experiments, which indicates a high degree of overfitting due to the train dataset size. The baseline random forest model produced moderately better validation performances, we conclude that this is because Bayesian optimization was used in hyperparameter space traversal of the random forest model so a systematic serach on a huge parameter space was performed which found a good parameter setting. Compared to that, the neural network model was hyperparameter tuned with fixed values of hyeprparameters, so had much lower chance of finding a well-performing parameter setting. From the ablation studies, the ablation model performed equally as the main mdoel and the RoBERTa based ablation mdoel udnerperformed than the main model. We hypothesize this is because in thhe main model the embeddings are produced from PatentSBERTa, which is fine-tuned for patent documents and so was capable to produce more informative embeddings.

8 FUTURE WORK

There was a data imbalance issue in the dataset, where the classes denoting strong similarities between anchor and target texts had very few samples. One improvement strategy can be trying different sampling schemes.nWe used simple neural network models in this study. State-of-the-art language model structures, for example, transformers can be used for the task presented in this study, and that may be able to achieve better generalizability. The embeddings can be preprocessed by dimension reduction strategies before passing to the predictive model. Instead of classification scheme, regression setup can be used in predicting the similarity scores.

9 WORK DISTRIBUTION

Until mid-term progress report, CK has performed experiments on random forests and hyper-parameter tuning of random forests. PAB has experimented with logistic regression and fixed hyperparameter settings of random forests. KS has experimented with the two versions of the proposed model. After mid-point progress report, CK has performed both ablation studies, PAB has performed the concatenation-based experiments, KS has performed hyperparameter tuning of the main model.

REFERENCES

[1] Hamid Bekamiri, Daniel S. Hain, and Roman Jurowetzki. 2021. PatentSBERTa: A Deep NLP based Hybrid Model for Patent Distance and Classification using Augmented SBERT. arXiv:2103.11933 [cs.LG]

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).