Conditional Semantic Similarity with Fine Tuning and Prompt Learning: a case study on US Patents Phrase to Phrase Matching

Alessandro Dipalma, Marco Petix

November 2022

Abstract

This case study aims to find a solution for the U.S. Patent Phrase to Phrase Matching competition, that challenges to build a model able to evaluate semantic similarity between two phrases given the patent categorization in which they are found. Two solutions are proposed: one based on pretrained transformers finetuning, and another based on the Pattern Exploiting Training strategy. The few shot learning capabilities of the PET strategy and efficiency are analyzed, showing that PET is more time efficient w.r.t. finetuning. The best performances on the internal test set are achieved training Electra-large with PET (86.92% Pearson correlation).

1 Introduction

1.1 The competition

The competition that inspired this project is the **U.S. Patent Phrase to Phrase Matching** hosted on the Kaggle platform. The objective is the development of a model providing the semantic similarity between pairs of key phrases extracted from U.S. patent descriptions.

The U.S. patent archives stands as a rare combination of data volume, quality, and diversity, also due to its two centuries of history. The challenge aims at the development of a model to ease the intensive vetting process prior to a patent grant by determining if an invention has been described before.

This task is further conditioned by the inclusion, as an additional feature, of the Cooperative Patent Classification (CPC) as a technical domain context to help disambiguate situations involving domain-dependent terms. Example: the term "resistant material" could refer to various concepts (cement, steel, denim) depending on the context of reference (construction, electronics, tailoring).

1.2 Data exploration

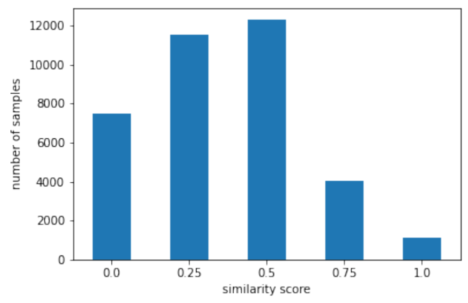
The dataset featured pairs of sentences, an **anchor** sentence and a **target** one, along with a code representing the **CPC_code** of the patent according to the Cooperative Patent Classification (CPC), thereby indicating the subject matter to which the patent relates. The semantic similarity of the two phrases, given the context provided, is represented in the dataset by a **score** value from 0 (not similar at all) to 1 (identical in meaning). Table 1 provides an overview of the aforementioned columns of the dataset.

The dataset accounts for 36473 sentence pairs. The study of the distribution of similarity scores associated with each of these revealed a low population of semantically identical, or almost so, pairs

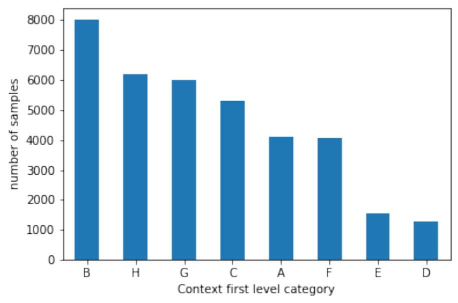
Column	Description
Id	Unique identifier for a pair of phrases
Anchor	The first phrase
Target	The second phrase
CPC_code	The CPC classification code indicating the field of knowledge within which the similarity has to be scored.
Score	Semantic similarity, conditioned on the **CPC_code**

Table 1: Overview of the Dataset columns

(score 0 and 0.25). On the other hand, the study of the distribution of the contexts associated with the sentences revealed a slight imbalance against a few of the covered contexts. Both distributions are represented in Figure 1.



(a) Similarity Score distribution



(b) Context Domain distribution

Figure 1: Similarity Score and Context Domain distributions

2 Related works

2.1 Sentence similarity and patent similarity

Generally, different semantic similarity approaches can be categorized into four groups:

- **Keyword-based methods** are based on keyword frequency and co-occurrence measures.
- Analysis of the **SAO structure**, where a triple structure is extracted from a text corpus that includes Subjects, Objects, and Actions.
- **Ontology-based analysis approach**, where the concepts and related relations are defined for a specific domain.
- **Machine-learning-based approaches** for text classification are able to map the complex relationships of texts and provide high accuracy compared to other approaches.

According to the importance of the machine learning approach, lately considerable scientific studies are being conducted to develop these methods to determine how to estimate semantic similarity and patent classification. Most of the studies on patent classification are based on Deep learning and Transformers, but due to the lack of access to the standard datasets and the same evaluation structure, a fair review of the literature is very difficult. Whereas, the existence of different approaches in the classification like IPC and CPC and diversity of information, both structured and unstructured such as Date, Claim, Abstract, and Description have added to this difficulty [11].

Chen and Chang (2012) proposed the hybrid approach as a three-phase categorization method that includes SVM, K-means, and KNN algorithms [2]. Tran and Kavuluru (2017) explore text data and machine learning-based classification in the context of the CPC system [16]. Grawe et al. (2017) introduce deep learning and word embedding methods for text-based patent classification [6].

Hepburn (2018) proposed the idea of using the transformers model for patent classification, using a Support Vector Machine (SVM) and Universal Language model with Fine-tuning [9]. Li et al. (2018) proposed DeepPatent as a deep learning algorithm for patent classification based on convolutional neural networks (CNN) and word vector embedding of patent title and abstract [11].

Lee and Hsiang (2020) proposed PatentBERT as a transformers model based on the BERT-Base pre-trained model for patent classification. They focused on fine-tuning the BERT model for patent classification and used a large dataset USPTO of over two million patent claims at the CPC subclass level [10]. Although in this approach they proposed a model with appropriate accuracy, they did not provide similarity measures between patents or suggest a workflow of how to do so.
Bekamiri et al. introduced a version of SBERT (Siamese BERT) trained on CPC texts. Their model, PatentSBERTa, has interesting performances in patent to patent similarity and CPC code prediction [1].

2.2 Few shot learning and prompt learning

Prompt learning is a new method of training models in AI, and in NLP in particular, to perform specific downstream tasks [5]. It involves using textual prompts or demonstrations to stimulate pre-trained language models (PLMs) without introducing additional parameters or task-specific objectives. Prompt learning can be effective in low-data scenarios and can adapt PLMs to various NLP tasks such as cloze-style prediction, autoregressive modeling, or sequence to sequence generation.

Prompt learning revolves around a template and a verbalizer, where a template is used to process the original text with some extra tokens, and a verbalizer projects the original labels to words in the vocabulary for the final prediction.
For example, for sentiment classification, the template could be:

$$\text{“}\langle\text{TEXT}\rangle\text{ It is }\langle\text{MASK}\rangle\text{”},$$

where the token ⟨TEXT⟩ stands for the original text, and the verbalizer could be:

$$\{\text{“positive”:“great”, “negative”:“terrible”}\}.$$

In this first dive in the prompt learning environment, we decided to follow the Pattern Exploiting Training (PET) method introduced in [14]. This technique aims at performing additional training to adapt the weights of Pretrained Language Models (PLMs) to a specific task. This semi-supervised training procedure reformulates input examples as cloze-style phrases to help language models understand a given task. These phrases are then used to assign soft labels to a large set of unlabeled examples. Finally, standard supervised training is performed on the resulting training set. This technique, and its iterative variant iPET, shows to be effective in low resources settings [15]. An example of the PET pipeline for the sentiment classification task is displayed on Figure 2.

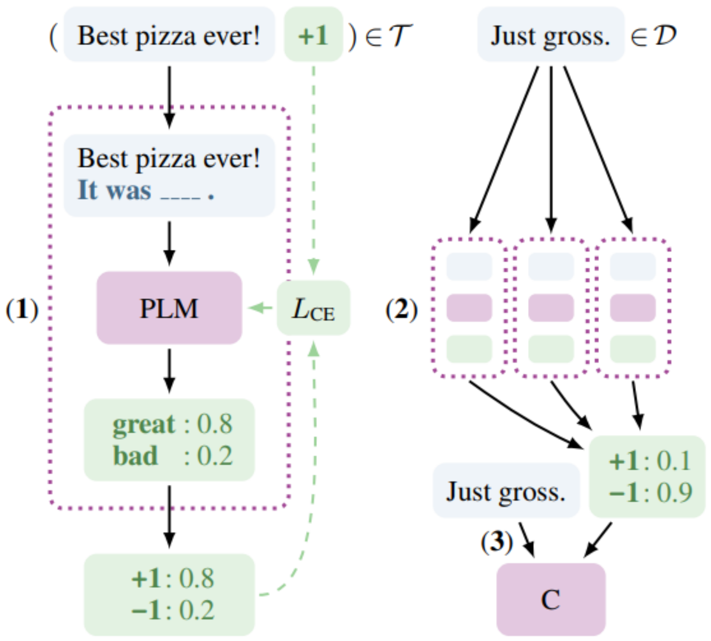


Figure 2: PET for sentiment classification. (1) A number of patterns encoding some form of task description are created to convert training examples to cloze questions; for each pattern, a pretrained language model is finetuned. (2) The ensemble of trained models annotates unlabeled data. (3) A classifier is trained on the resulting soft-labeled dataset. Courtesy from [14].

3 Proposed solution(s)

In this case study, we follow two significantly different approaches to solve the task prompted by the challenge. Both approaches rely on the use of a PLM. The first approach is a classical finetuning, in which we adapt the weights of a PLM to solve our task by training on the dataset via k-fold cross validation. Secondly, we experiment the application of prompt learning to this semantic similarity prediction task. For both techniques, in order to compare them properly, we use the same set of transformers and the same set of "context-features". For "context-feature" we mean the additional context that we concatenate to the phrases pair to produce the embedding for the finetuning and the prompt for the prompt learning.
These common aspects are described in the following two subsections.
All the experiments were run on a single machine with a Nvidia P100 GPU having 16 gb of dedicated memory, an 80 core intel Xeon CPU and 500gb of ram.

3.1 Context Features

In order to make further use the information provided by the competition dataset, the attributes described in Table 1 have been involved in the definition of some additional context-features. The following paragraphs provide a detailed description of such features but a brief overview is presented in Table 2.

The **CPC** context-feature maps the Cooperative Patent Classification code contained in the **CPC_code** attribute to its natural language description for the sake of comprehensibility and a

Features	Description
CPC	Natural language description of the context described by the **CPC_code** feature
	e.g. from A41 to *human necessities*
SACT	List of **target** values paired up with both the same **anchor** and **context**
	e.g. abatement[SEP]act of abating[SEP]active catalyst, eliminating process, forest region...
SACST	Like **SACT** but with **target** values from pairs with similarity **score** greater or equal to 0.75
	e.g. abatement[SEP]act of abating[SEP]active catalyst, eliminating process...
CPC+SACST	Concatenation of **CPC** with the list of **target** values from **SACST**
	e.g. abatement[SEP]act of abating[SEP]human necessities[SEP]active catalyst, eliminating process...

Table 2: Overview of derived features.

more meaningful embedding.
The next three features aim to exploit the bond between each single anchor-target pair and other similar pairs in the dataset. These are:

- **same_anchor_context_targets** (**SACT**), is a concatenation of all the **target** values from pairs for the same **anchor** and **context** values;
- **same_anchor_context_similar_targets** (**SACST**), mimics the **SACT** feature but excludes from the **target** values list the ones from pairs with low similarity (**score** < 0.75);
- **CPCdescription_same_anchor_context_similar_targets** (**CPC+SACST**), is a concatenations of the **CPC** and **SACST** features.

Other than the semantics of each feature, it is important to mark out the length that each context-feature has, since a long input text will cause a longer training phase. Figure 3 shows the distributions of the word count for each one. It can be observed that **SACT** is by far the longest context-feature, with an average length of 60 word and a maximum of 300. The **CPC** one, instead is the shortest. We also noticed that the **CPC**, **SACT** and **SACST** features can also produce texts that are very short or even empty. This doesn't happen with the **CPC+SACST** feature that guarantees a non-empty context for every sample.

3.2 Transformers

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based neural network architecture that was introduced by Google in 2018. It uses a bidirectional transformer encoder to pre-train deep bidirectional representations from unlabeled text. The vanilla BERT was the starting point of our research. To search for the model that would better perform for the given task, we experimented with transformers architectures of different sizes and different pretraining techniques. The pretrained transformers involved in our analysis, accessed via the huggingface transformers library, are:

- `BERT-base-uncased` [4].

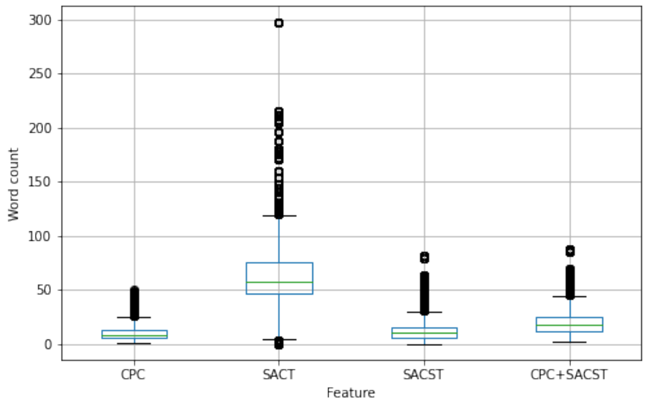


Figure 3: Word count for each context-feature built.

- `distilBERT-base-uncased`. DistilBERT is a version of BERT that uses knowledge distillation to reduce its size by 40% while retaining 97% of its capabilities. It uses a similar general architecture as BERT but with fewer encoder blocks (6 blocks compared to 12 blocks of BERT and 24 blocks of RoBERTa) [13].

- `ahotrod/electra_large_discriminator_squad2_512`. ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) is another transformer-based neural network architecture that was introduced by Google in 2020 [3]. It uses a generator-discriminator approach to pre-train deep bidirectional representations from unlabeled text. Since we perform a comparison between finetuning approach and few shot learning, in which we ask the model to answer a question, we expected a model pretrained on a question answering benchmark to suit our needs.

- `microsoft/deBERTa-v3-large`. DeBERTa is an extension of BERT that uses an ELECTRA-style pre-training method [8]. It uses a two-stream attention mechanism to capture both local and global dependencies. DeBERTaV3 is an improved version of DeBERTa that uses a similar structure as ELECTRA [7]. This is the largest model we use.

- `anferico/BERT-for-patents`. This is a BERT large model pretrained on a large patents corpora. The pretraining dataset was made not only from US patents. We expect such a model to be quite competitive with the others, despite the lower model capacity given by the standard BERT architecture w.r.t. more parameters-efficient model such as ELECTRA and DeBERTa.

- `AI-Growth-Lab/PatentSBERTa`. A version of SBERT (Siamese BERT) trained on CPC texts. In their work, they show interesting performances in patent to patent similarity and CPC code prediction. [1]

3.3 Transformers Finetuning

The first methodology that we explored has been the finetuning of pretrained transformers. Despite the dataset being divided in 5 classes, the problem was managed as a regression task. A continuous prediction has two advantages: it can be used as a fine rank for many applications, such as synonym

Transformer	Short name	#parameters
DistilBERT-base-uncased	DistilBERT	66362880
BERT-base-uncased	BERT	109486464
AI-Growth-Lab/PatentSBERTa	PatentSBERTa	344702976
ahotrod/electra-large-discriminator-squad2-512	Electra-l	334092288
Yanhao/simcse-BERT-for-patent	BERT-for-patents	355359744
microsoft/deBERTa-v3-large	DeBERTa-l	434012160

Table 3: Overview of the Transformers

retrieval, and from the competition viewpoint, can mitigate inexact predictions.
For each experiment, the models were trained via a 4-fold stratified cross validation, with a maximums of 8 epochs. Given the involvement of large transformers, an exhaustive search was impractical with our means. Preliminary trials to decide the parameters space to explore were run using only DistilBERT, given the smaller amount of time required to finetune it.

3.3.1 Transformers Pretrained Performance

Before starting the grid-search, we checked how the selected PLM's would perform without finetuning. The results of this analysis are presented in Table 4.

Transformer	Pearson's Correlation Score
distilBERT-base-uncased	-0.020384
ahotrod/electra-large-discriminator-squad2-512	-0.019712
BERT-based-uncased	0.027264
Yanhao/simcse-BERT-for-patent	0.012302
microsoft/deBERTa-v3-large	-0.019823

Table 4: Transformers performance before fine tuning on the **SACST** feature.

3.3.2 Search space

The following sections discuss the various terms of comparison regarding the experiments we performed: batch size, loss, model architecture, stratification strategy, features, PLMs, learning rate and warmup steps. Table 5 provides an overview of the comparison.

Batch Size For the sake of reducing the cumulative training time required by the many comparisons, we used the highest batch size enabled by our GPUs for each of the transformers involved in the analysis.

Loss Since the competition score is the Pearson correlation coefficient, which is a differentiable function, we attempted to use the competition score as loss function, instead of the Binary Cross Entropy (BCE) loss, a more standard choice for regression tasks. This comparison was run over DistilBERT and involved all the context-features described previously. The warmup steps were fixed to 0, the minimum learning rate to 2e-5 and an ADAM optimizer was used. Table 6 clearly

parameter	**values**
Batch size	The largest possible
Loss	BCE, Pearson
Model Architecture (readout type)	linear, attention, lstm
Stratification strategy	by score, by score and context
Feature	CPC, SACT, SACST, CPC+SACT
PLM	DistilBERT, BERT, BERT-for-patents, Electra-l, DeBERTa-l
Learning rate (starting)	1e-4, 2e-5, 1e-6
Warmup steps (percentage)	0, 0.01, 0.1, 0.5, 1

Table 5: parameters search space.

shows that BCE performs better (+2.5% on validation) independently from the additional context that we use.

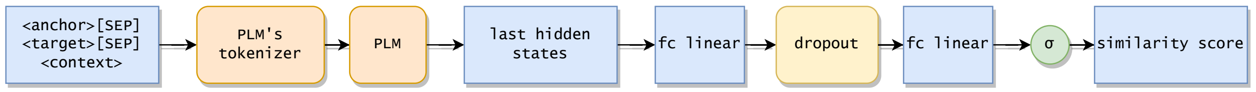
	validation score (%)	
feature	**with BCE loss**	**with Pearson loss**
CPC	76.70	73.60
SACT	79.27	77.73
SACST	76.63	74.15
CPC+SACST	78.24	74.81
mean	**77.71**	**75.07**

Table 6: Validation score using different losses with DistilBERT.

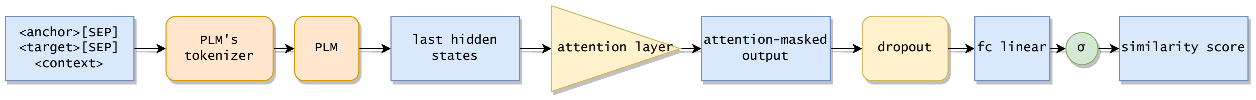
Model Architecture We restricted the choice of the network that elaborated the PLM embeddings to three architectures: linear, attention layer, LSTM layer, all followed by a dropout and a sigmoid activation function. The choice was performed by running some configurations using DistilBERT as PLM. Putting an attention layer gave better results, and was also much more feasible w.r.t. the LSTM layer, that, especially on the longest features, took far too long to train. Figure 4 presents the architecture of the model used for the finetuning. Table 7 reports a numerical comparison over the readout architectures. Putting an attention layer on top of the PLM gives +2.7% performances, requiring a +9.3% time for each epoch, w.r.t. the linear readout. From the variance in the table and the learning curves in Figure 5 we can also observe that with the attention layer the model learns much faster and the performances are more consistent across folds. Thus, we choose to keep the attention layer since it gives a performance boost and converges in less epochs.

Stratification Strategy As mentioned in Section 3.3, the experiment implied the application of a 4-Fold Stratified Cross Validation. For the sake of better balanced fold populations, we compared different stratification strategies. The main two strategies compared in such way are:

- **Stratification by score**: based on the **score** feature, it divides the samples in 5 bins ([0-4]) and aims to balance the fold population w.r.t. to the degrees of similarity between the **anchor-target** pairs.
- **Stratification by score and CPC code**: while still accounting for the **score** feature values, it also consider the initial character from the **context** feature so to encourage more balanced



(a) linear readout



(b) attention readout

Figure 4: Model architectures for finetuning. Fc stands for Fully Connected layer.

	validation score mean		**validation score variance**		**epoch ETA (s)**	
feature	**attention**	**linear**	**attention**	**linear**	**attention**	**linear**
CPC	76.70	73.53	1.30×10^{-6}	4.62×10^{-5}	1.03×10^{2}	9.01×10^{1}
CPC+SACST	78.24	76.13	2.24×10^{-5}	1.21×10^{-5}	1.70×10^{2}	1.55×10^{2}
SACST	76.63	72.78	4.79×10^{-7}	6.16×10^{-5}	1.39×10^{2}	1.28×10^{2}
SACT	79.27	77.63	3.98×10^{-6}	5.07×10^{-5}	7.10×10^{2}	6.51×10^{2}
mean	77.71	75.02	7.03×10^{-6}	4.26×10^{-5}	2.80×10^{2}	2.56×10^{2}

Table 7: Validation score with different readout architectures. The fixed params are: PLM: DistilBERT, warmup=0, stratification=by score and **CPC** code, max epochs=8. min lr=1e-6.

folds w.r.t. to both the similarity and associated context of the **anchor-target** pairs.

The estimate of comparison between the two strategies was computed by setting up four pairs of runs to train different instances of the DistilBERT transformer. Each pair involved a different context-feature from the four described in Section 3.1 and employ either one of the stratification strategies. The results of such comparison are presented in Table 8 and also displayed, averaged w.r.t. the features, in Figure 6. Ultimately both them point out the slightly better results implied by the **Stratification by score and CPC code** strategy.

Features and PLMs The four context-features described in Table 2 were used as inputs for the different PLMs in order to evaluate and compare their effectiveness. The results of such comparison, w.r.t. both the model trained and the feature used as embedding, are shown in Figure 7. What follows is a discussion based on said results.

As mentioned in Section 3.1, the **CPC** feature, relying only on information associated with the

	validation score (%)	
features	**stratification by score**	**stratification by score and CPC code**
CPC+SACST	77.10	78.24
CPC	76.36	76.57
SACST	76.25	76.63
SACT	78.81	79.22
mean	**77.13**	**77.67**

Table 8: Comparison of validation scores over the different features.

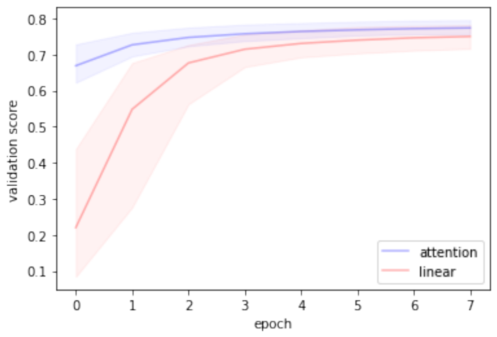


Figure 5: Learning curves with different readout architectures. Shaded regions surrounding the lines depict the fold variance.

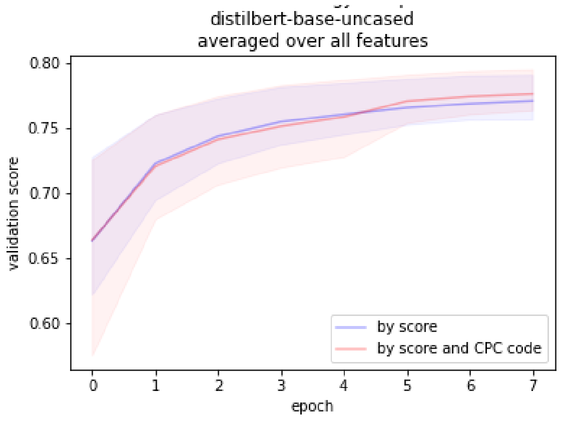


Figure 6: Comparison between the Stratification Strategies. The transparent band shows the min and max over the different folds.

single sample, provided the simplest and shortest input text of the set. It fared best with models on the larger side of the spectrum, displaying optimal results with both the BERT-for-patent, the Electra and the DeBERTa transformers. Smaller transformers like BERT and DistilBERT instead did not benefit as much from the single-sample information producing inferior results compared to more complex features.

The employment of the **SACT** feature, as the one resulting with the longest input text of the set, has consistently provided the best results for each transformer. Regarding the interaction of the feature with the largest transformers, obtaining such promising results has required a detailed search for a suitable learning rate that characterized the latest stages of this project development.

The runs employing the **SACST** feature provided slightly worse results w.r.t. to the ones provided both by the **CPC** and **SACT** features. This last comparison interestingly shows the degrading effect of excluding the information associated with **anchor-target** pairs with low similarity **score**.

Ultimately, the employment of the concatenation-based feature **CPC+SACT** has provided second-to-top results for every transformer.

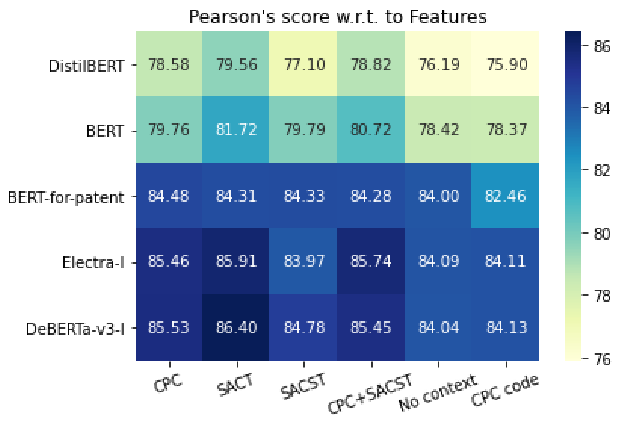
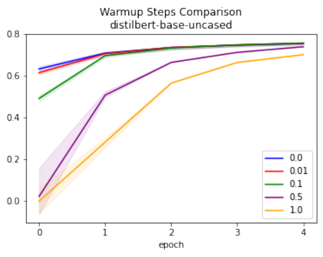
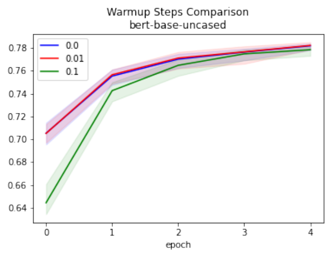


Figure 7: Heatmap of model-feature performance

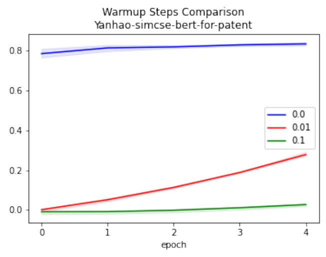
Learning rate and warmup steps Many configurations, especially on large transformers, have shown great instability when trained with larger input text probably due to their inherent susceptibility to relatively high learning rates. This was also accompanied by some instances of overfitting. We therefore tested two workarounds: adding some warmup steps and lowering the learning rate.



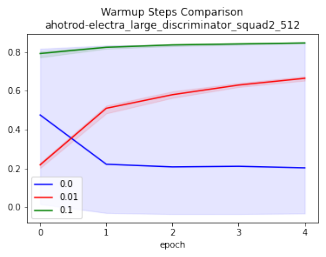
(a) DistilBERT



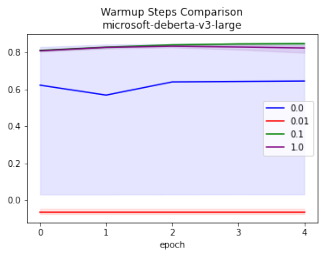
(b) BERT



(c) BERT-for-patents



(d) Electra-l



(e) DeBERTa-v3-l

Figure 8: The effect of warmup steps when training over the CPC feature. Only first 5 epochs are shown. A similar behavior was observed on all other features.

We expected the aforementioned instability to benefit greatly from the addition of warmup steps and we performed a deep analysis regarding their impact w.r.t. to the amount of steps added on top of the runs. Figure 8 presents the learning curves computed from runs for each transformer with increasing percentages of warmup steps w.r.t. the total number of training steps. Analyzing the impact of this strategy on the DistilBERT-based runs, as clearly portrayed in Figure 8(a), the increasing amount of warmup steps only delays the convergence of the small transformer. Figure

8(e) and Figure 8(d) instead shows how the DeBERTa- and Electra-based runs with few, or none, warmup steps display a low average score w.r.t. to the task in analysis. The graph also show how increasing the amount of warmup steps has ultimately led to higher results on average.

Lowering the learning rate down to $1e-6$ also led to very positive performances when dealing with larger transformers, ultimately resulting in our best finetuning configuration overall, which is the one of DeBERTa trained on the **SACT** feature.

3.3.3 Results

The following observations can be made.

- The overall analysis has shown how performances in general tend to scale with the size of the models involved in the task at hand.
- The DistilBERT-based runs have shown to reach about 97% of the performances achieved by the BERT-based ones, this is perfectly in line with the key claims regarding DistilBERT.
- BERT-for-patents, despite being specifically pretrained on the patent domain, never managed to reach the performance of the Electra and DeBERTa transformers. The performance displayed by these models, having slightly more parameters than BERT-for-patent, prove their flexibility and efficient architecture.
- Larger models usually reach the plateau in 3 to 5 epochs, then starting to overfit and causing the validation performances to incrementally drop. BERT and DistilBERT, instead, stabilize around the 8th epoch.

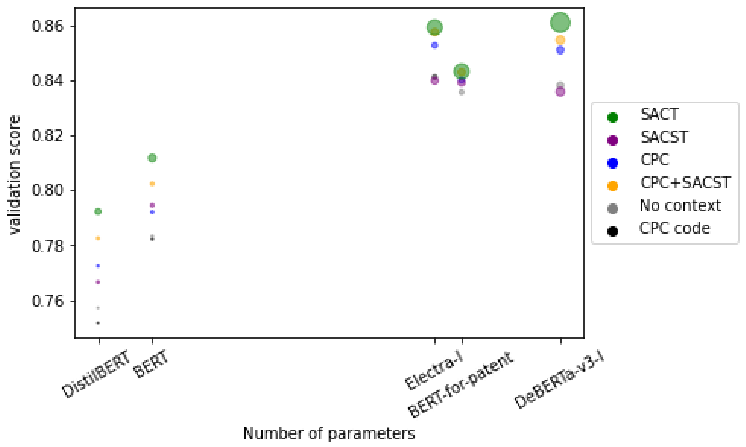


Figure 9: Performances w.r.t. PLM size and feature. The size of the points is proportional to the training time needed to reach that performances.

Figure 9 represents the results obtained for each transformer and feature involved in the runs we analyzed. The plot highlights the differences in terms of complexity and performance of each model while also displaying the training time, for a single epoch, by the size of each run-representing point. The runs employing the Electra transformer, especially the one based on **SACT** feature, distinguish themselves from the ones employing the slightly bigger BERT-for-patent transformer for their better results and relatively faster training times.

parameter	**values**
PLM	DeBERTa-l
Feature	SACT
Model Architecture (readout type)	attention-based
Loss	BCE
Learning rate (starting)	1e-6
Warmup steps (percentage)	10%

Table 9: Run configuration for the competition submission

From the same figure, it can be observed a relatively small training time gap between the runs of BERT with the **SACT** feature and the larger transformers trained on the **CPC** or the **SACST** ones; training larger models on shorter inputs ultimately shows to be more time efficient.

Final training The configuration elected for the competition submission is described in Table 9. The model was trained for 5 epochs and resulted in a 85.71% on the internal test set.

3.4 Prompt-based learning: PET

As stated in section 2.2, we employ the PET technique. We divided the dataset into four different sets: training, dev, test and unlabeled. The latter containing the samples involved in the unsupervised part of the PET formula. Figure 10 shows how data is used in the PET pipeline.

Prompt engineering We then defined the templates to be filled in order to build the prompts. These are:

p0 ”<anchor>” and ”<target>” Are related if we are talking of <CPC>? [MASK]

p1 ”<anchor>” and ”<target>” Are relatable in the same context of <SACT>? [MASK]

p2 ”<anchor>” and ”<target>” <CPC> Are they similar? [MASK]

p3 ”<anchor>” and ”<target>” Have these phrases a semantic correlation? [MASK]

The choice of using only the **CPC** and **SACT** context-features came from the fact that in the previous proposed solution they both proved to be the most effective to learn from. We use the PLMs to predict the probability of the words in the [MASK] position.

Verbalizers The pair (label, word) is called Pattern Verbalizer Pair (PVP) and the verbalizers we considered are the following:

- No/Yes: the samples with similarity **score** $\in [0.0, 0.25]$ are mapped to the No class, those with **score** $= 0.5$ are excluded and, those with **score** $\in [0.75, 1.0]$ are mapped to the Yes class.
- No/Quite/Yes: the samples with similarity **score** $= 0$ are mapped to the No class, those with **score** $\in [0.25, 0.75]$ are mapped to the Quite class and, those with **score** $= 1.0$ are mapped to the Yes class.

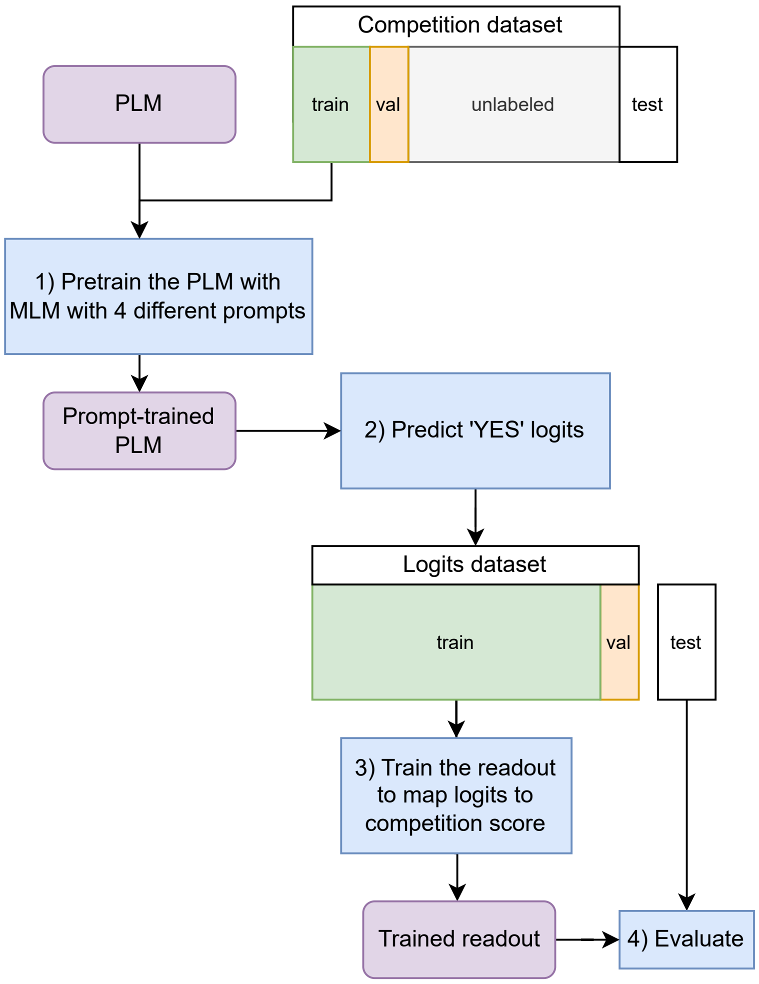


Figure 10: Few shot learning pipeline.

- All (No/Poor/Maybe/Probably/Yes): all the similarity **score** unique values from the training dataset, considering the values from 0 to 1 with steps of 0.25, are mapped to the 5 categories.

Some attempts were also made involving a mapping that included the samples for similarity **score** = 0.5 in the No class but with no meaningful results. The most probable label is given by a softmax layer over the logits produced by the model.

Readout The PET pipeline is designed for classification tasks, but our task is a regression one: we used a regressor to map the PET output logits into the semantic similarity score. This regressor was required due to our use of 2 to 3 classes which couldn't be mapped into the 5 similarity score bins from the competition dataset. The training of this regressor was performed by feeding it the logits from the data samples belonging to the unlabeled portion of the dataset. This technique is supposed to increase the model robustness to noisy prediction in a way not unlike the readout training performed in the Echo State Networks.

We use the same PLMs used in the finetuning adding the GPT2 [12] model to the comparison. Obviously, the former wasn't included in the previous proposed solution due to its generative nature.

We then built a new simple dataset to be used in training the regressor. The dataset attributes are the logits predicted for all the verbalizer labels, while the target is the similarity score associated with each sample. We want to highlight that the readout training could even be useless, since the competition is evaluated via Pearson correlation. Due to the few features that constitute this new dataset, the training time to train the regressor is considered to be negligible.

Different types of regressor were used (Multi Layer Perceptron (MLP), Support Vector Regressor (SVR), Logistic Regressor. We do not delve in the details of the model selection of this regressor, since many configurations were able to reach the same results. We attempted to train the regressor both using only the YES logits and the whole set of logits. Using all of the logits performed better for the All verbalizer, the one accounting for 5 classes, while for the other two did not provide much improvements.

Few-shot and Many-shot approaches The described pipeline was used in two ways:

- to perform few-shot learning, using about 5 to 10% of the dataset to perform the pretraining step and the rest left for the readout training and final.
- to train the model on the whole dataset, swapping the proportions of the pretraining and unlabeled sets.

3.4.1 PET results

In the following analysis we point out the differences in performances among the transformers and PVPs involved in our experiments. As stated before, PET proved to be useful when dealing with a modest amount of data but also for models with a modest amount of parameters.

In Table 10 we report the best performing run for each transformer used in the PET pipeline. With **No-Readout Training** we refer to the Pearson's Correlation Coefficient between the logits for the *Yes* class and the similarity score of the samples in analysis. **Train** and **Val** instead refer to the Pearson's Correlation Coefficient between the same similarity scores and the output of the regression readout layer.

Few-shot results In Figure 12 we report the performance of each run for every transformer involved in the analysis and for any of the employed PVPs. We can see that GPT-2 is surpassed even by DistilBERT in our experiment, which is an outcome that deserve further analysis. A first explanation can be found in the fact that GPT models are known to be more prone to sintactic correctness w.r.t. semantic one. We can observe that the performances scales with the model size, with DistilBERT reaching 90% of the BERT performances. Differently from the finetuning, BERT-for-patents here outperforms DeBERTa-v3-l: this can be due the importance of having a model that has already been pretrained to the patents corpus. Also notice that BERT-for-patent reaches this result with the prompt *p3*, that does not include any context feature, hence requiring half the DeBERTa-v3-l training time. Electra-l stands on the top of the performances rank; on the few shot learning, Electra-l shows to be the most effective PLM (see Table 11). The *All* split with the *p1* and *p0* prompts shows to be the most effective PVPs. As stated earlier, we expected this version of Electra-l to be effective since it is pretrained for question answering.

			Score			
PLM	**Pattern**	**Verbalizer**	**No-Readout Training**	**Train**	**Val**	**Train Time**
GPT-2	p3	All	22.73 20.47 25.63 26.54 27.05	42.12	40.15	902.49
DistilBERT	p3	No/Quite/Yes	-43.54 14.21 42.22]	49.79	47.13	499.92
BERT	p1	No/Quite/Yes	-50.06 6.32 46.42	54.70	52.36	840.70
DeBERTa-v3-l	p1	No/Quite/Yes	-66.33 20.16 61.37	71.68	70.09	5037.75
BERT-for-patent	p3	All	-65.55 -22.42 48.85 64.83 54.84	72.38	71.09	2483.27
Electra-l	p1	All	-70.19 -35.54 58.22 69.13 49.11	75.30	73.96	2497.30

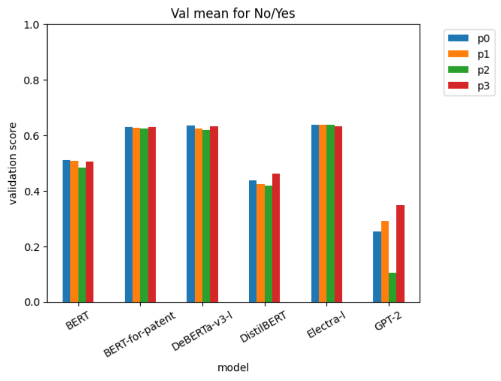
Table 10: Top performing PVPs for each PLM trained on the 10% of the competition dataset.

			Score (%)			
PLM	**Pattern**	**Verbalizer**	**No-Readout Training**	**Train**	**Val**	**Train Time**
Electra-l	p1	All	[-70.19 -35.54 58.22 69.13 49.11]	75.30	73.96	2497.30
Electra-l	p0	All	[-70.75 -30.05 59.99 69.05 47.87]	75.19	73.95	2522.55
Electra-l	p3	All	[-70.51 -40.13 60.15 67.68 49.54]	74.16	72.63	2496.35
Electra-l	p1	No/Quite/Yes	[-69.13 42.05 69.48]	73.36	71.87	2539.22
BERT-for-patent	p3	All	[-65.55 -22.42 48.85 64.83 54.84]	72.38	71.09	2483.27
Electra-l	p0	No/Quite/Yes	[-65.42 44.95 67.29]	72.50	71.09	2557.04
DeBERTa-v3-l	p1	No/Quite/Yes	[-66.33 20.16 61.37]	71.68	70.09	5037.75
BERT-for-patent	p1	All	[-65.47 -21.42 56. 65.18 59.25]	71.53	70.09	2474.37
Electra-l	p2	No/Quite/Yes	[-68.69 25.09 68.18]	71.85	70.09	2548.95

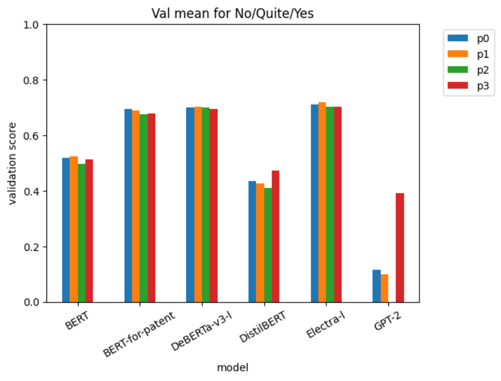
Table 11: Top performing configurations for PET few shot runs.

Notice that on the *All* split, training the readout is fundamental to get a good Pearson correlation score, because the similarity information is not condensed only in the *Yes* logit, while for the *No/Quite/Yes* and *No/Yes* split the *Yes* logit already has an high correlation with the similarity.

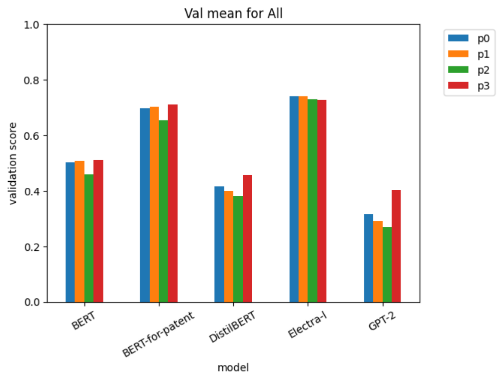
Many-shot results Training the PLMs with the PET pipeline using the whole competition dataset led to higher performances w.r.t. the fine-tuning. The results are reported in Table 12 and figure 11(e). We can see that: training on the whole dataset makes the models match the fine-tuning performance in about the same amount of time for DistilBERT and BERT. Electra, instead is able to get up to 88% validation score with a training phase that requires about 10% of the time needed for the fine-tuning. It can also be noticed that the discrepancy in performances among patterns shrinks when training on larger amount of data.



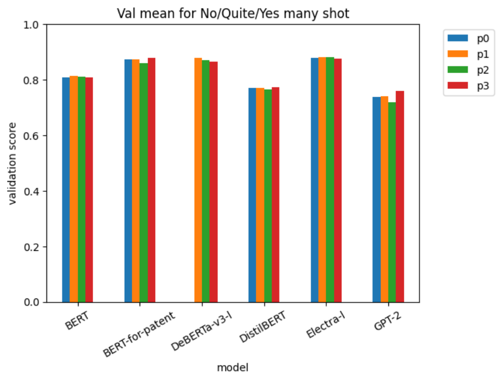
(a) No/Yes few shot



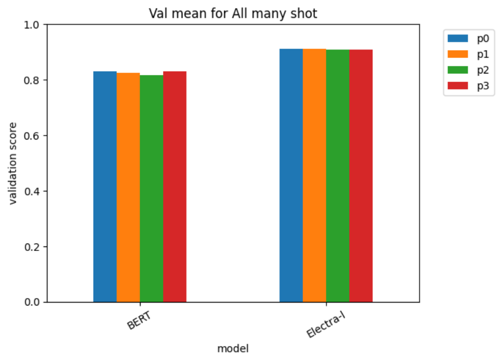
(b) No/Quite/Yes few-shot



(c) *All* few-shot



(d) No/Quite/Yes many-shot



(e) *All* many-shot

Figure 11: PET performances for each PVP and PLM.

			Score (%)			
PLM	**Pattern**	**Verbalizer**	**No-Readout Training**	**Train**	**Val**	**Train Time**
Electra-l	p1	All many shot	[-72.37 -61.35 26.65 85.64 70.73]	91.06	91.26	10 863.52
Electra-l	p0	All many shot	[-64.4 -60.3 30.69 86. 76.76]	91.26	91.20	10 819.28
Electra-l	p2	All many shot	[-70.16 -51.73 32.9 87.18 78.33]	90.98	90.93	207 176.07
Electra-l	p3	All many shot	[-70.96 -64.86 33.23 83.63 73.38]	90.92	90.91	10 878.56
Electra-l	p2	No/Quite/Yes many shot	[-81.98 60.66 84.95]	87.14	88.01	10 554.44
Electra-l	p1	No/Quite/Yes many shot	[-86.38 26.14 84.97]	86.73	87.94	10 581.94
Electra-l	p0	No/Quite/Yes many shot	[-86.66 10.94 83.69]	86.57	87.86	10 543.72
BERT-for-patent	p3	No/Quite/Yes many shot	[-88.5 7.44 87.96]	86.65	87.85	204 153.65
DeBERTa-v3-l	p1	No/Quite/Yes many shot	[-87.69 -2.19 82.37]	85.77	87.74	20 060.53

Table 12: Top performing configurations for PET many shot runs.

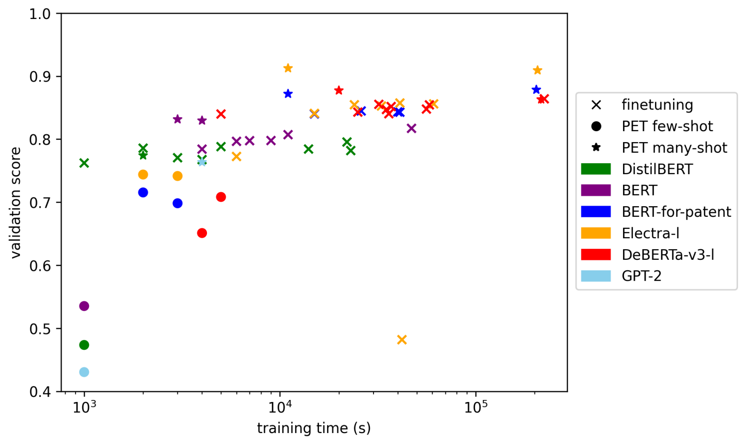


Figure 12: Performance comparison between PET and fine-tuning.

4 Finetuning VS PET

Figure 12 shows the differences in performance for each PLM w.r.t. time and training strategy. We can see that we were able to get better performances with BERT and Electra using PET, in less time. As we can see, the largest models are much more suitable for prompt based learning w.r.t. the smaller ones. What really surprised us about the PET pipeline was its leaning toward a more ”plug and play” approach w.r.t. the traditional fine-tuning of a PLM. We did not have the chance to neither try different combinations of hyperparameters nor to define any new learning objective. This means that there could still be a large amount of improvement over the current solution.

5 Internal test and Competition submission results

Table 13 shows the results of the chosen configurations for fine-tuning and PET on our internal test set and on the competition blind test set, aside the competition ranking.

		Score (%)	
Method	Model	Internal test set	Competition blind test set
Fine-tuning	DeBERTa SACT	85.7	84.67
PET-few-shot	Electra-l (p2, All)	73.13	79.31
PET-many-shot	Electra-l (p2, All)	86.32	83.59

Table 13: Results our top performing models on the competition blind test set.

6 Conclusions and future works

In this case study we proposed two different solutions to the US Patent Phrase to Phrase Matching competition. To get results that were competitive with the leaderboard, it was necessary to enrich the competition datasets adding to each phrase a context built with intra- and/or extra-dataset information. First, we built a model using a PLM embedding and fine-tuned its parameters to fit the competition task. This kind of model shows performances that overall scale with the model size, although we observed that some tricks in the PLM architecture can make a general purpose model to perform significantly better even if it is not pretrained on the specific domain. We compared BERT-for-patent, which is a version of BERT-large pretrained on the US patent corpora, with Electra-l and DeBERTa-v3-l that are instead trained on general purpose corpora.
While our dataset is not that small, we aim to investigate the potential benefits of trying alternative training processes and smaller training time for bigger models like the DeBERTa-large transformer. We consider being able to obtain similar performances using only a small fraction of the available data to be an interesting and useful achievement, since it means that the approach work also for categories having a smaller amount of patents. Interesting results were obtained using the PET strategy: when trained on only the 10% of the available data, Electra-l reached 75% validation score, about the same performances reached with BERT-base trained on the whole dataset.

The PET adoption performed in this case study is certainly not optimal, since the hyperparameters space was not explored and 4 handcrafted prompts were used. Performing autonomous prompt search[5], while out of our current scope, stands as a meaningful potential addition to the current pipeline.

References

[1] Hamid Bekamiri, Daniel S. Hain, and Roman Jurowetzki. Patentsberta: A deep nlp based hybrid model for patent distance and classification using augmented sbert, 2021.

[2] Yen-Liang Chen and Yuan-Che Chang. A three-phase method for patent classification. *Information Processing Management*, 48(6):1017–1030, 2012.

[3] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. Openprompt: An open-source framework for prompt-learning, 2021.

[6] Mattyws F Grawe, Claudia A Martins, and Andreia G Bonfante. Automated patent classification using word embedding. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 408–411. IEEE, 2017.

[7] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023.

[8] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.

[9] Jason Hepburn. Universal language model fine-tuning for patent classification. In *Proceedings of the Australasian Language Technology Association Workshop 2018*, pages 93–96, 2018.

[10] Jieh-Sheng Lee and Jieh Hsiang. Patentbert: Patent classification with fine-tuning a pre-trained bert model. *arXiv preprint arXiv:1906.02124*, 2019.

[11] Shaobo Li, Jie Hu, Yuxin Cui, and Jianjun Hu. Deeppatent: patent classification with convolutional neural networks and word embedding. *Scientometrics*, 117:721–744, 2018.

[12] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[14] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference, 2021.

[15] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners, 2021.

[16] Tung Tran and Ramakanth Kavuluru. Supervised approaches to assign cooperative patent classification (cpc) codes to patents. In *Mining Intelligence and Knowledge Exploration: 5th International Conference, MIKE 2017, Hyderabad, India, December 13–15, 2017, Proceedings 5*, pages 22–34. Springer, 2017.