Patent Classification on Search-Optimized Graph-Based Representations

Jarkko Lagus[1], Ekaterina Kotliarova[1] and Sebastian Björkqvist[1]

[1]*IPRally Technologies Oy, Helsinki, Finland*

Abstract

Patent documents can be effectively represented using embeddings derived from graphs. These graph-based representations capture the intricate relationships and contextual information within the documents. By leveraging the power of graph embeddings, we can create rich document representations that can be further fine-tuned to enhance their performance for specific tasks.

In this paper, we aim to address the fundamental question if search-optimized graph-based document embeddings can be directly used for classification. Traditionally, different training pipelines and storage mechanisms were required for each distinct task, resulting in increased complexity and resource consumption. However, by establishing whether the same representations can be effectively used for both search and classification, we can streamline the process and eliminate the need for maintaining multiple sets of embeddings.

Our results provide evidence that embeddings optimized for search tasks can be directly employed to perform classification tasks, offering a promising solution that significantly improves efficiency and resource utilization. By repurposing the same set of optimized embeddings for both search and classification, we not only achieve data efficiency but also reduce computational overhead. This approach allows us to leverage the benefits of existing search-optimized embeddings without sacrificing the accuracy or effectiveness of classification tasks. As a result, we present a novel and efficient classification method that reduces the complexity of maintaining separate training pipelines and storing multiple representations.

Keywords

classification, patents, document embeddings, patent search

1. Introduction

The categorization of patent documents plays a crucial role in various aspects of strategic decision-making, such as competitor monitoring, portfolio management, and patent landscaping. Document classification itself is a foundational task in natural language processing and a vast amount of research has been done both using traditional machine learning approaches and deep learning-based approaches [1]. Specifically, in the domain of patent classification, approaches using methods such as convolutional neural networks [2] and transformers [3, 4, 5] have been used. Performing document classification for patents manually can be very time-consuming and often requires domain expertise. This means that the amount of labeled data available for training may be small.

As patent documents are already categorized by the patent offices using the International or Cooperative Patent Classification (IPC/CPC) standards, it may be tempting to try mapping these classes directly to the classification task of interest. These classes however rarely correlate with the actual business tasks, so simple mapping from these classes to classes of interest does not often work [6].

Due to the discrete nature of certain metrics, direct optimization becomes challenging. Consequently, in various machine learning tasks, a common approach is to solve the main objective by optimizing a substitute target instead. An illustrative case of such a task is ranking, where instead of directly solving the discrete ranking problem, we turn it into a problem of optimizing pairwise distances. Inspired by this concept, we investigate the approach of performing classification directly on document embeddings that have been optimized for a search task.

The work presented here is based on the hypothesis that graph-based embeddings optimized for a search task contain rich enough information to be directly applied to a classification task with no additional fine-tuning steps. Only training a lightweight classification model on top of the embeddings is needed. This approach results in a very efficient way to perform classification as such models scale well to larger datasets and in the usual problem scale can be trained in a few seconds. This enables online training of new classification models on the fly, allowing for quick verification of results and multiple iterations if needed.

PatentSemTech'23: 4th Workshop on Patent Text Mining and Semantic Technologies

jarkko@iprally.com (J. Lagus); ekaterina@iprally.com (E. Kotliarova); sebastian@iprally.com (S. Björkqvist)

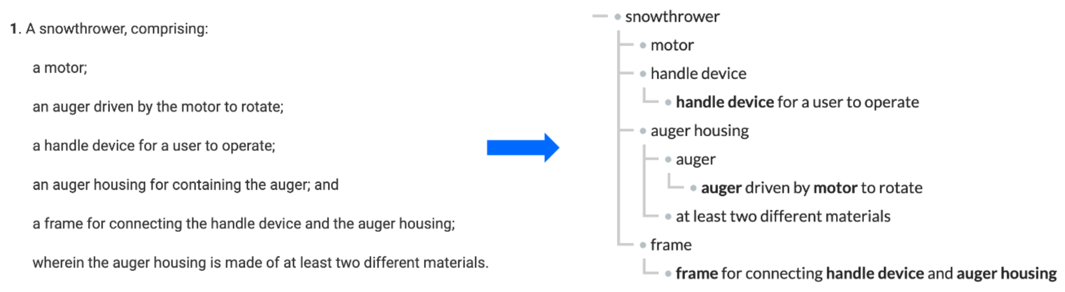

1. A snowthrower, comprising:
a motor;
an auger driven by the motor to rotate;
a handle device for a user to operate;
an auger housing for containing the auger; and
a frame for connecting the handle device and the auger housing;
wherein the auger housing is made of at least two different materials.
snowthrower
motor
handle device
handle device for a user to operate
auger housing
auger
auger driven by motor to rotate
at least two different materials
frame
frame for connecting handle device and auger housing

Figure 1: An example of a patent claim describing a snowthrower and the corresponding graph created from the claim. When used in the downstream classification task, these graphs are further encoded as d-dimensional vectors using a graph neural network model.

2. Methodology

Representations for text documents can be made in various ways. In this work, our main focus is to investigate the usability of a search-optimized graph-based embedding method, where the patent document is first parsed into an intermediate graph representation that is then turned into an embedding. This is then compared to other common document embedding methods.

2.1. Graph-based representations optimized for search

In contrast to traditional methods, such as word embedding or transformer-based approaches, where the whole document is directly encoded into a vector format without task-specific regularization, the graph format adds additional prior information about the relations between the elements in the document. The idea of the graph is to describe all the relevant technical features of a patent in a concise form that is easily understandable by humans and efficient to process by machines. An example of a patent claim converted to a graph can be seen in Figure 1.

The details of how the graphs and embeddings are created are described in [7]. In short, the process is the following:

1. Turn the text of a patent document into a graph using a specialized parser, resulting in a collection of nodes and edges.
2. Embed the graph into a vector space using a graph neural network model trained to perform prior art searches for patents.

The parser that converts text to graphs uses the spaCy [8] library to do a linguistic analysis of the text and to detect all nouns and noun chunks in the text. The nouns in the text describe the features of the invention and become the nodes of the graph. In Figure 1, examples of nouns and noun chunks are *snowthrower*, *motor*, and *handle device*. After this, the parser detects, using hand-crafted rules, words indicating relationships between the features of the invention (e.g. comprising, having, containing). These words will create edges to the graph. The endpoints of the edges are found using the output of the linguistic analysis done previously. For instance, in Figure 1 the term *comprising* will result in, among others, an edge between *snowthrower* and *motor*.

The graph neural network model is trained in a supervised manner using citations reported by patent office examiners, resulting in documents having similar technical content being placed close to each other in the embedding space. The model is trained using triplet loss, where a patent application acts as the anchor and a patent cited by the application is the positive sample. The negative sample is chosen to be some other patent document that is not cited by the application. This results in a model that is useful for searching for prior art for new inventions. The embeddings used for the later classification stage are created from the description graph of the patent document, the description graph including both the claims and the description text of the document.

2.2. Other document embedding models

In order to comparatively measure the effectiveness of our embedding method, we conduct the experiments using a few additional models to provide meaningful baselines. For the baseline evaluations, we create document embeddings using five different methods: TF-IDF embeddings, two different GloVe [9] embeddings and two different BERT-based [10, 4, 11] embeddings (see Table

Embedding model	Dimensionality
Ours [7]	150
TF-IDF	$\approx 33,000$
GloVe (Stanford) [9]	300
GloVe (patents)	300
BERT (base uncased) [10]	768
BERT (patents) [11, 4]	1,024

Table 1
The set of different embeddings used in the experiments. *BERT (patents)* is the large BERT and *GloVe (patents)* is the standard GloVe model trained with patent data.

1 for more details). All the embedding models chosen represent conceptually different ways of forming the document embeddings. The embeddings are created using the full text of the patent document, including both the claims and the description of the document.

For TF-IDF embeddings we use `scikit-learn` [12] library. To form the document embeddings out of the GloVe embeddings, we use the `spaCy` [8] library. For the BERT models we use `HuggingFace` [13] library. Because of the limitations of input layer size and the length of patent documents, to form the BERT-based document embeddings, we split the documents into chunks of 100 tokens, and embed each chunk individually. After this, we extract all the separate embeddings and form a mean vector representation out of these.

2.3. Classification models

As one of the goals is to minimize the training cost for the classification model, we employ simple classification models instead of heavy deep learning models. The only requirement we impose on the model is the ability to output a probability estimate for the input sample belonging to a specific class. For the classification, we use ready-made implementations from the `scikit-learn` [12] library. The specific models chosen are the basic *logistic regression* and *k-nearest-neighbors* classifiers using the default parameters.

2.3.1. Model training

We train each model using a training set separated from the full dataset. The input for the models is the document embedding and the output is the probability for each label. In the case of the binary dataset, we train one classifier. In the case of the multi-label dataset, we train one binary classifier for each class following the one-versus-rest strategy leading into a collection of m separate binary classifiers.

For the experiments with full data, we train one classifier for each dataset-model pair. As the outputs are probabilities, we need to find the optimal cut-off threshold that maximizes the F1 score. This threshold is selected using stratified 5-fold cross-validation. In both, binary and multi-label cases, only one threshold is selected. For the multi-label case, the threshold that maximizes the micro-averaged F1 score of all classifiers is chosen.

For the experiments where we limit the data amount, we first randomly sample p percent of data points (with p varying from 0.5 to 75) and then follow the same procedure as with the full data case. The sampling is done so that all the models are trained using the same fixed subset. When training on a subset of data, we repeat the training process n times in order to reduce the amount of noise caused by poor train-validation split, where n varies from 2 for the largest subsets to 10 for the smallest subsets.

Dataset	Labels	Train size	Test size
Qubit [6]	2	1,124	282
Mechanical eng.	10	3,768	943

Table 2
Dataset statistics for the datasets used for training and evaluation. In both datasets only one document per patent family is preserved to avoid overrepresenting certain families.

2.3.2. Model evaluation

Evaluations are done using a separate holdout test set independent of training data. For evaluation, we calculate the standard F1 scores. In the case of the multi-label dataset, micro averaging is used. We conduct the evaluation on two different datasets, one binary, and one multi-label. The binary dataset is the gold-standard Qubit patent dataset [6] and the multi-label dataset is a proprietary dataset from a mechanical engineering patent domain (see Table 2 for details).

The same holdout test set is used for all evaluations, both for the experiments with the full data and the experiments with subsets of the data. To convert the predicted probabilities to binary predictions we use the optimal threshold selected using the training phase.

3. Experiments

We experiment with how different choices of embedding the documents (see Table 1 for the list of methods) affect the performance. Our main interests are classification accuracy (measured using the F1 score), sample efficiency, and training time. When measuring the training time, we do not consider the time required to create the document embeddings or include the hyperparameter search but assume that the embeddings are readily available and the optimal hyperparameters are known.

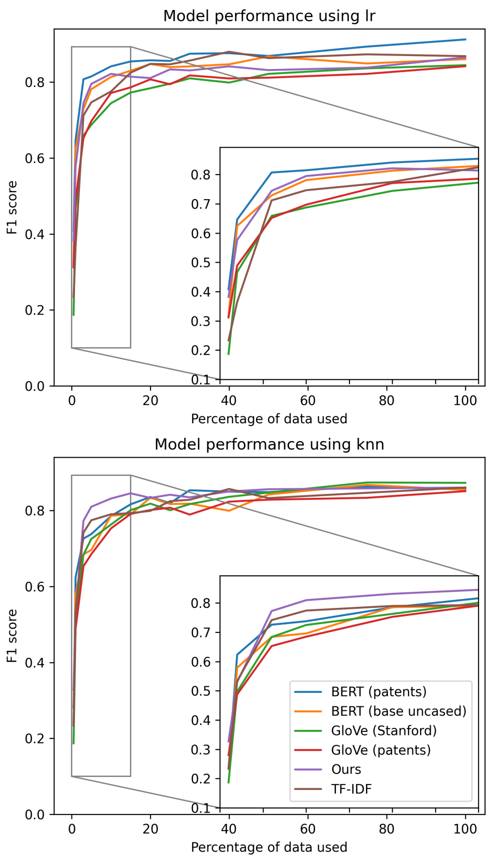


Figure 2: The effects of the amount of training data on the Qubit dataset on model performance over different embeddings.

Qubit dataset (binary)

Embedding type	**Model**	**F1**	**Time (s)**
BERT (base uncased)	knn	0.854	0.049
BERT (patents)	knn	0.856	0.065
GloVe (Stanford)	knn	**0.873**	0.003
GloVe (patents)	knn	0.851	0.003
Ours	knn	0.860	0.011
TF-IDF	knn	0.860	1.216
BERT (base uncased)	lr	0.860	0.135
BERT (patents)	lr	**0.912**	0.184
GloVe (Stanford)	lr	0.844	0.035
GloVe (patents)	lr	0.842	0.021
Ours	lr	0.865	0.021
TF-IDF	lr	0.868	1.792

Mechanical engineering dataset (multi-label)

Embedding type	**Model**	**F1**	**Time (s)**
BERT (base uncased)	knn	0.655	1.215
BERT (patents)	knn	0.664	1.561
GloVe (Stanford)	knn	0.691	0.041
GloVe (patents)	knn	0.645	0.040
Ours	knn	**0.775**	0.261
TF-IDF	knn	0.698	53.909
BERT (base uncased)	lr	0.719	4.448
BERT (patents)	lr	**0.770**	5.734
GloVe (Stanford)	lr	0.681	0.595
GloVe (patents)	lr	0.717	0.750
Ours	lr	**0.770**	0.374
TF-IDF	lr	0.752	80.881

Table 3
Evaluation results for models trained on the full train set for the Qubit and mechanical engineering datasets for all embedding types and classification models.

3.1. Experiments on embedding performance

To measure the overall embedding performance, we look into two factors, overall accuracy measured in F1 score and required training time following the process described in Section 2.3.1. The results are summarized in Table 3.

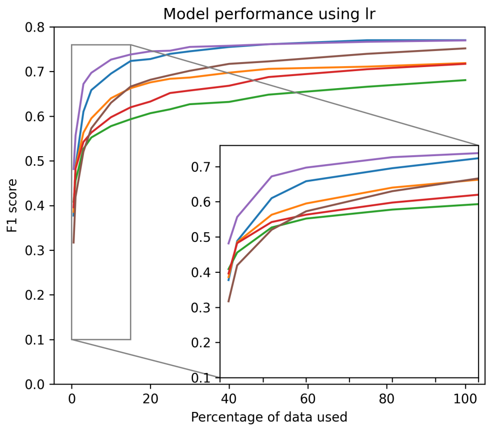
The F1 scores on the Qubit dataset show much variability between models and embedding methods: For instance, the GloVe (Stanford) embeddings perform the best when using the k-nearest-neighbors (*knn* in the figures) model but the second worst when using the logistic regression (*lr* in the figures) model. This suggests that the results on the Qubit dataset do not give much information about which model or embedding method works the best. For the multi-label dataset, however, the results are more consistent, with our approach reaching the top performance with both models.

From the training times, we can see a direct correlation between the training time and the embedding size. Especially the TF-IDF embeddings show an extreme case of this requiring over ten-fold time compared to any other embedding type. The main reason for poor training speed with TF-IDF is, however, that the models used do not support sparse training.

3.2. Experiments on sample efficiency

To experiment with how well the models perform when data is scarce, i.e. how much data is actually needed to gain reasonable performance, we limited the amount of training data to smaller subsets of specific percentages. The same test set was used here as in the previous experiment on full data.

From Figures 2 and 3, we can see that most models start to plateau already when around 30% of full data is included. On the Qubit dataset, the same effect of no clear separation seems to be present as well when using smaller subsets of the data, similar to what was seen with the full data case. The curves fluctuate over each other,



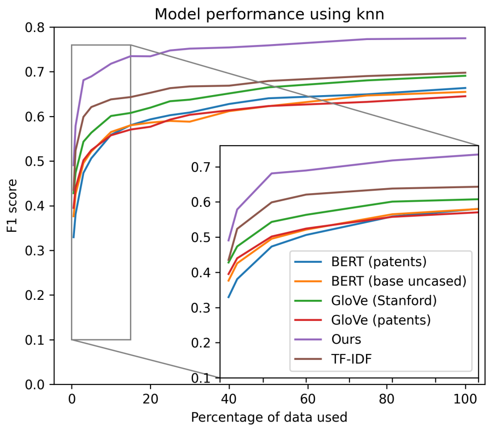


Figure 3: The effects of the amount of training data on the multi-label mechanical engineering patent dataset on model performance over different embeddings.

and no clear distinction can be seen between models.

In the multi-label case, however, clear differences show up: When using 0.5% of the data there is almost a 20-percentage-point difference between the best (Ours) and the worst (TF-IDF with *lr* and BERT (patents) for *knn*). The performance difference between the models decreases when the number of samples is increased, but the rankings of the different models mostly stay the same regardless of the amount of data used, with our method reaching the highest scores on virtually all subset sizes.

4. Conclusions

In this paper, we showed that patent classification can be done efficiently on rich graph embeddings optimized for a search task. We evaluated the performance on both a binary and a multi-label dataset and demonstrated that search-optimized embeddings work well with a very limited amount of labeled samples in the multi-label case. In the binary dataset case, the results were inconclusive. We showed that training of the classification models can be done in less than a second, enabling users to train classifiers in an online fashion. Due to the limited amount of datasets available, nothing conclusive can be said about the generalization capabilities of the method, but we believe this result generalizes to any rich-enough embeddings optimized for a search task. Further investigations are needed to say anything conclusive, however.

References

[1] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown, Text classification algorithms: A survey, Information 10 (2019) 150.

[2] S. Li, J. Hu, Y. Cui, J. Hu, Deeppatent: patent classification with convolutional neural networks and word embedding, Scientometrics 117 (2018) 721–744.

[3] J.-S. Lee, J. Hsiang, PatentBERT: Patent classification with fine-tuning a pre-trained BERT model, arXiv preprint arXiv:1906.02124 (2019).

[4] R. Srebrovic, J. Yonamine, Leveraging the BERT algorithm for Patents with TensorFlow and BigQuery, 2020. URL: https://services.google.com/fh/files/blogs/bert_for_patents_white_paper.pdf.

[5] H. Bekamiri, D. S. Hain, R. Jurowetzki, Patentsberta: a deep nlp based hybrid model for patent distance and classification using augmented sbert, arXiv preprint arXiv:2103.11933 (2021).

[6] S. Harris, A. Trippe, D. Challis, N. Swycher, Construction and evaluation of gold standards for patent classification—a case study on quantum computing, World Patent Information 61 (2020) 101961.

[7] S. Björkqvist, J. Kallio, Building a graph-based patent search engine, in: 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'23), to appear, 2023. doi:`https://doi.org/10.1145/3539618.3591842`.

[8] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spacy: Industrial-strength natural language processing in python, zenodo, 2020, 2020.

[9] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.

[10] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. `arXiv:1810.04805`.

[11] F. Cariaggi, BERT for Patents, 2023. URL: https://huggingface.co/anferico/bert-for-patents.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[13] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45.