Semi-Supervised Learning Methods for Patent Classification Using Search-Optimized Graph-Based Representations

Ekaterina Kotliarova[1], Sebastian Björkqvist[1]

[1]IPRally Technologies Oy, Helsinki, Finland

Abstract

Classifying patent documents into categories is an important step in many decision-making processes like competitor monitoring, patent landscaping, and portfolio management. Performing the classification task manually is time-consuming, so the amount of training data available to train machine learning models for classification is often limited. The lack of training data is thus frequently the limiting factor in the performance of the classification model. In this work, we share semi-supervised learning methods used to alleviate the issue of the lack of training data. We utilize pre-trained search-optimized representations to automatically label additional training data by performing label propagation using nearest neighbor searches in the vast space of patent documents and demonstrate that the classifiers' performance improves compared to using the smaller, original training data sets and to naive oversampling methods.

Keywords

classification, label propagation, patents, document embeddings, patent search

1. Introduction

The classification of patent documents plays an important role in strategic decision-making. Supervised machine learning models require labeled datasets for predicting new input data, yet the manual labeling of patents is labor-intensive and often leads to limited or imbalanced training data. This highlights a crucial difference between academic datasets and real-world scenarios in patent classification. In academic settings, researchers typically work with well-curated datasets like CIFAR-10 and IMAGENET, focusing on optimizing models for these datasets [1, 2]. In contrast, when working on real-world patent classification, the data are not fixed. Users can continuously add new patents to the dataset or remove some, change labels, etc. Moreover, in some cases only a few manually labeled data points are provided, while unlabeled data may be abundant. In this work, we cover this specific setting: the user might provide labels for only a small set of documents, and the database could have millions of structured, but unclassified patent documents. Therefore, to improve the classifier performance, we are free not only to modify the customers' datasets but even to collect additional data.

Our previous work [3] showed 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. As a logical continuation of the aforementioned work, this paper presents an algorithm already used in our commercial product [4], that utilizes knowledge about the embeddings' structure in the vector space to enlarge customer datasets using label propagation techniques and k-nearest neighbors algorithm.

2. Literature survey

In machine learning, we primarily discuss two key concepts: supervised and unsupervised learning [5]. Supervised learning involves having examples with known inputs and corresponding outputs. The goal is to build a model that can predict outputs for new inputs accurately. On the other hand, unsupervised learning deals with datasets that contain only inputs. Unsupervised learning models aim at uncovering patterns or clusters within these inputs. Finally, semi-supervised learning combines aspects of both supervised and unsupervised learning. For instance, in classification tasks, we may utilize data without labels to enhance the performance of our model [6].

In the last twenty years, many different semi-supervised classification methods have been proposed. These methods vary in their conceptual approach to semi-supervised learning, their use of unlabeled data, and their integration with standard supervised algorithms. The survey [6] suggests the following taxonomy to distinguish between them. At the basic level, there are two types of methods: *inductive* and *transductive*. Firstly, they have different goals: *inductive* methods aim at creating a classification model, while *transductive* methods focus only on label prediction for the unlabeled data points. *Inductive* methods have clear *train* and *test* phases, while

5th Workshop on Patent Text Mining and Semantic Technologies (PatentSemTech) 2024

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

transductive algorithms are provided with labeled data (X_L, y_L) and unlabeled data X_U, and output exclusively predictions y_U for the unlabeled data. Since the method we analyze in this work is *transductive*, the literature survey below focuses mainly on the latter.

The transductive methods typically define a graph over all data points, both labeled and unlabeled, encoding the pairwise similarity of data points with possibly weighted edges [7]. Graph-based semi-supervised learning methods generally involve three separate steps: *graph creation*, *graph weighting* (these two steps could also be described as one *graph construction phase*), and *inference* [8, 9].

After the first two steps, we have a graph consisting of a set of nodes corresponding to the data points, and a weight matrix, containing the edge weights for all pairs of nodes. Once the graph is constructed, it is used to obtain predictions for the unlabeled data points.

Three common methods for graph creation include ϵ-neighbourhood, k-neighbourhood, and b-matching [6]. The ϵ-neighbourhood method connects nodes within a distance of ϵ. The k-neighbourhood method links each node to its closest k neighbors. The b-matching method is a post-processing step used with k-neighbourhood to ensure the graph's regularity, checking that the nodes have the same number of neighbors and a specified number of edges. Regarding the graph weighting step, in many cases, graph weights reflect the similarity measure employed for the edge construction, as in the case of the method we are about to introduce.

Concerning the inference stage, the literature survey [10] suggests that the graph-based methods predominantly can be viewed as estimating a function f on the graph. One wants f to satisfy two conditions at the same time: 1) it should be close to the given labels y_L on the labeled nodes, and 2) it should be smooth on the whole graph. In other words, this can be explained as a loss function and a regularizer. Examples of different loss functions and regularizers used in graph methods are described in [11, 12, 10].

A similar approach to our work was used in [11], who experimented with graph construction using a k-nearest neighbors algorithm and the ϵ-neighbourhood (connecting pairs of data points with distance smaller than ϵ). They kept the edge weights fixed and uniform but experimented with changing the weight of edges between unlabeled data points relative to the other edges [6].

3. Methodology

Numerous manipulations can be conducted on datasets to enhance metric performance, achieve greater result stability, and facilitate improved model generalization. In this work, our main focus is to investigate the usability of search-optimized graph-based embeddings for the datasets enlarging and fine-tuning and therefore better classification metrics scores. We compare our suggested semi-supervised method to up-sampling, wherein existing samples within the dataset are duplicated to address class imbalance.

Dataset	Labels	Full train size	Test size
Qubit [13]	1	985	421
Cannabinoid [14]	1	1,117	478
Mechanical eng.	10	3,295	1412
Chemical	5	887	380

Table 1
Statistics for the datasets used for training and evaluation. In all datasets only one document per patent family is preserved to avoid overrepresenting certain families.

3.1. Datasets

Four datasets were chosen for this study, comprising two binary datasets and two multi-label datasets. The binary datasets consist of the well-established Qubit dataset [13] and the Cannabinoid patent dataset [14]. The multi-label datasets are proprietary and originate from distinct domains: one from the mechanical engineering patent domain and the other from the chemistry field (refer to Table 1 for specific dataset details).

3.2. Experiment setup

Each model is trained using a distinct training set extracted from the complete dataset. The models take document embeddings as input and generate probabilities for each label as output. For binary datasets, a single classifier is trained. Conversely, for multi-label datasets, a binary classifier is trained for each class using the one-versus-all approach, resulting in a set of m individual binary classifiers.

For the experiments on the subsets of data, the training set was partitioned based on predetermined percentages (i.e. we randomly sample p percent of data points with p varying from 0.5 to 100). The criterion for a successful sampling attempt is the presence of at least one positive and one negative sample in each class of the dataset. We would call this subset A of training data D with at least one positive and one negative sample per class an *original* subset, $A \subseteq D$.

Following this, to enlarge the training data subset A and have higher test scores, we search for k additional samples for every sample $a \in A$, $k \in \{3, 5, 7, 10\}$. The algorithm for creating an additional dataset and propagating labels to it is detailed in Section 4.3. Here we would only mention that the additional samples are strictly not from the same patent families as the sampled training data and the test data. Therefore, in the final training dataset, where the additional and original training data are merged, all samples belong to different patent families.

To provide a comparison, we perform an up-sampling procedure for the same *original* subset. We copy each sample $a \in A$ k times, where $k \in \{3, 5, 7, 10\}$. Then we merge copied samples with the *original* dataset A, as we did with the semi-supervised additional samples. The copied samples have the same labels and embedding vectors as the original ones.

Hence, each model discussed in the corresponding section (see Section 3.3) is trained using two training datasets. One dataset involves *up-sampling*, where copies of data points are included, while the other is a *semi-supervised* dataset enriched with *additional* data. It's crucial to emphasize that for both datasets the original samples $a \in A$ are the same and only the enlarged part changes. Therefore the models' training results obtained from these two datasets can be directly compared to each other.

When training on a subset, we repeat the random sampling and training processes n times to reduce the amount of noise caused by a poor train-validation split, where n varies from 2 for the largest subsets to 10 for the smallest subsets.

This methodological approach was designed to maintain a result comparability across iterations. By utilizing the same test set and keeping the same training set for different models, the obtained metrics are comparable. Metrics from multiple sampling iterations for the same percentage were then averaged.

Since the models' outputs are probabilities, it's crucial to determine an optimal cut-off threshold that maximizes the F1 score. This threshold is determined through a stratified 5-fold cross-validation. In both binary and multi-label scenarios, only a single (global) threshold is selected. In the multi-label case, we choose the threshold that maximizes the micro-averaged F1 score across all classifiers.

3.3. Choice of classification models

Building upon our previous work [3], we have decided to continue using the same classification models for this study. Our selection criteria emphasize minimizing training costs and ensuring the models can provide probability estimates for sample classification. Specifically, we utilize the basic *logistic regression* and *k-nearest-neighbors* classifiers with default parameters, implemented using the *scikit-learn* library [15] among with *XGBoost* classifier using library [16].

3.4. Model evaluation

Evaluations for all subsets of data were done using a separate holdout test set independent of the training data. For demonstration purposes, we calculate the standard F1 scores. In the context of the multi-label dataset, micro-averaging is applied. To transform predicted probabilities into binary predictions, we employ an optimal threshold determined during the training phase.

4. Transductive semi-supervised method for dataset enlarging

As highlighted in the literature survey, the proposed semi-supervised method has a *transductive* nature, because it has the following three stages: graph creation, graph weighting, and inference.

In the first stage, we create embeddings for patents. This involves parsing patent document text to construct a *patent document graph* and then using a neural network to embed this graph into a vector space. We then utilize the embeddings for each patent document as the foundation when creating the *search space graph* used to perform the label propagation. It's crucial to differentiate between the concepts of the patent document graph and the search space graph. The concept of the patent document graph is specifically relevant in Section 4.1, as it pertains to the embedding creation process. Conversely, the search space graph is a recurring concept throughout the paper.

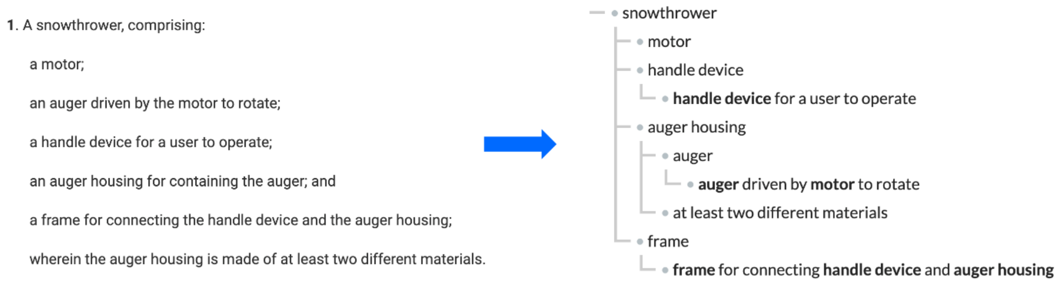
The second phase is the search space graph creation and weighting using the nearest neighbors algorithm. Finally, the third phase is the inference step, where we find possible new members of the dataset and propagate labels for them.

4.1. Graph-based representations optimized for patent search

To incorporate the patent domain-specific information into the embeddings used for classification, we utilize a two-stage process:

1. The text of a patent document is parsed into a patent document graph using a specialized parser. The result is a collection of nodes and edges describing the invention found in the original text.
2. A graph neural network is used to embed the graph into a vector space, where the network is trained to perform prior art searches in the patent domain.

The graph of a patent document contains all relevant features of the invention described in the original text, and also the relationships between them. An example of



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: A patent claim describing a snowthrower and the graph conversion of that claim. The classification is performed on the vector created from this graph by a graph neural network model.

a patent claim converted to a graph can be seen in Figure 1.

The specialized parser works as follows: First, a linguistic analysis of the text is done using the spaCy [17] library, and uses this information to detect all nouns and noun chunks in the text. The detected nouns and noun chunks are the features of the invention, and they become nodes in the graph. In Figure 1, examples of nouns and noun chunks turned into nodes are *snowthrower*, *motor*, and *handle device*.

After the features have been detected, the parser extracts relationships between the features utilizing the output of the linguistic analysis combined with hand-crafted rules that recognize common phrases used in patent texts. The outcome of this process is a set of edges connecting the pairs of feature nodes. For instance, in Figure 1, the parser will recognize the term *comprising*, and it will, among others, result in an edge between *snowthrower* and *motor*.

Due to the removal of duplication and some legal jargon, the graphs are smaller than the original texts and thus faster to process, while also allowing downstream models to learn faster since the relationships between features are explicitly encoded. More details on the patent document graphs and how they are created can be found in [18].

The graph neural network is then given the parsed graphs as input and trained to perform patent searching in a supervised manner by using patent office examiner citations. The result is a model that embeds documents with similar technical content near each other in the embedding space, regardless of the exact words used. Further information on the training of the graph neural network is found in [18].

The embeddings created by the graph neural network can then be used for patent classification purposes by training a lightweight classification model on top of the embeddings [3].

4.2. Search space graph creation for label propagation

To create the search space graph to use for the label propagation, we perform for each existing sample in the training data set a nearest neighbor search in the embedding space of patent documents, where the embeddings are created using the method described in section 4.1. The total size of the search space is over 100 million documents, reflecting the large number of available patent documents. The nearest neighbor searches are done using the Annoy library [19].

When creating the graph we exclude from the nearest neighbor search results any documents that are already a part of the dataset, to guarantee that the label propagation results in completely new documents being used for the training.

4.3. Inference stage

We are given a set L of labeled samples, and a set U of unlabeled samples. Since in our case, the search space graph G has millions of patents, U is exceedingly large. After the construction of the graph $G = (V, E)$, weights are assigned to the edges between all vertices (i.e. patent documents' embeddings) based on a cosine distance between them. To enlarge the set L and propagate labels for the subset of the set U, we perform the next steps:

1. For a sample $l \in L$ we find k neighbors among $u \in U$, using the Annoy library [19]. Let's mark these neighbors as V_{-}. Any found neighbor $v_{-} \in V_{-}$ shouldn't belong to the same patent family as any $l \in L$ sample. If the neighbor v_{-} shares the same patent family as the sample l, then we skip

Qubit Dataset											
		XGBoost					Logistic Regression				
		Semi-supervised		Up-sampling			Semi-supervised		Up-sampling		
%	N_{train}	k=5	k=10	k=5	k=10	Baseline	k=5	k=10	k=5	k=10	Baseline
0.5	4	**0.47**	0.45	0.41	0.41	0.26	**0.68**	0.54	0.33	0.31	0.37
1	9	**0.56**	0.53	0.46	0.46	0.44	**0.64**	0.6	0.46	0.43	0.54
3	29	0.7	**0.73**	0.49	0.51	0.55	**0.81**	0.8	0.75	0.73	0.79
5	49	**0.78**	**0.78**	0.56	0.6	0.71	**0.84**	0.83	0.83	0.81	0.83
10	98	0.81	**0.82**	0.68	0.69	0.75	0.85	**0.87**	0.86	0.85	0.86
15	147	0.81	**0.82**	0.68	0.69	0.78	0.85	0.85	0.85	0.85	**0.86**
20	197	**0.85**	**0.85**	0.73	0.75	0.81	**0.87**	**0.87**	0.86	0.86	0.85
25	246	0.83	**0.85**	0.72	0.73	0.80	0.87	**0.88**	**0.88**	0.87	0.87
30	295	0.84	**0.86**	0.75	0.76	0.85	0.87	0.87	**0.88**	0.87	0.87

Table 2

F1-scores computed for the Qubit dataset [13]. % symbol means the percentage of the training dataset utilized during the training, while N_{train} represents the count of the original samples (i.e. without additional data of any kind) used for the training. "Semi-supervised" refers to the approach we propose, whereas "up-sampling" denotes a technique wherein existing samples within the dataset are duplicated. The variable k signifies the number of neighbors added for each original sample (or the number of made sample copies if the "up-sampling" method was used). "Baseline" refers to the F1-score computed using only the original data ($k = 0$).

it and move to the next, more distant neighbor. Therefore, we always find k-neighbors for sample l, but they might be not the closest ones.

2. We assign the same labels as the sample l has to all $v_- \in V_-$. Consequently, each neighbor $v_- \in V_-$ becomes $v_+ \in V_+$.
3. Since all $v_+ \in V_+$ samples have labels now, we merge them into the L set.
4. We repeat this procedure for all samples $l \in L$.

The motivation for this algorithm is that if edges between samples that are similar to each other are given a high weight, then it's probable that they share the same labels since this conforms with the basic assumption of many learning algorithms that similar examples should be classified similarly.

5. Results and discussions

Results are presented in Figure 2 and Table 2. The semi-supervised method showed improvements in metrics for XGBoost and logistic regression classifiers when using small subsets of training data ($p \leq 25\%$). However, for larger subsets, enlarging the dataset did not lead to further improvement in the metrics. This observation was expected, especially for the binary datasets, as lightweight models like logistic regression typically do not require many samples per class to achieve an F1-score plateau.

For future work, it would be beneficial to have an understanding of why the k-nearest neighbors classification algorithm's metrics are barely affected by additional data.

Moreover, exploring the influence of varying the parameter k in the k-nearest neighbors search algorithm and its impact on the F1-score is worthwhile. Our experiments involved training models with additional data where parameter k varies in $\{3, 5, 7, 10\}$ set. Although the semi-supervised method generally yielded the best results for $k = 10$, the differences in metrics across different k parameters were relatively modest. Notably, we observed a noticeable increase in time consumption and memory usage with higher k values, indicating a trade-off between the model performance and resource utilization.

There is also a potential to modify the algorithm to fix the class imbalance more effectively rather than simply appending as much data as possible.

6. Conclusions

In conclusion, our study highlights the effectiveness of graph embeddings optimized for a search task in the context of semi-supervised learning. Our evaluation focused on two binary and two multi-label datasets from different patent domains. The methods used in the work are also utilized in our commercial product. The results show that our semi-supervised approach for dataset enlargement outperforms the naive up-sampling method. We observed improvements in metrics for XGBoost and logistic regression classifiers when using small subsets of training data, indicating the potential of our method for handling limited data scenarios.

Overall, our work contributes to advancing semi-supervised learning algorithms and their applicability in real-world scenarios with limited labeled data.

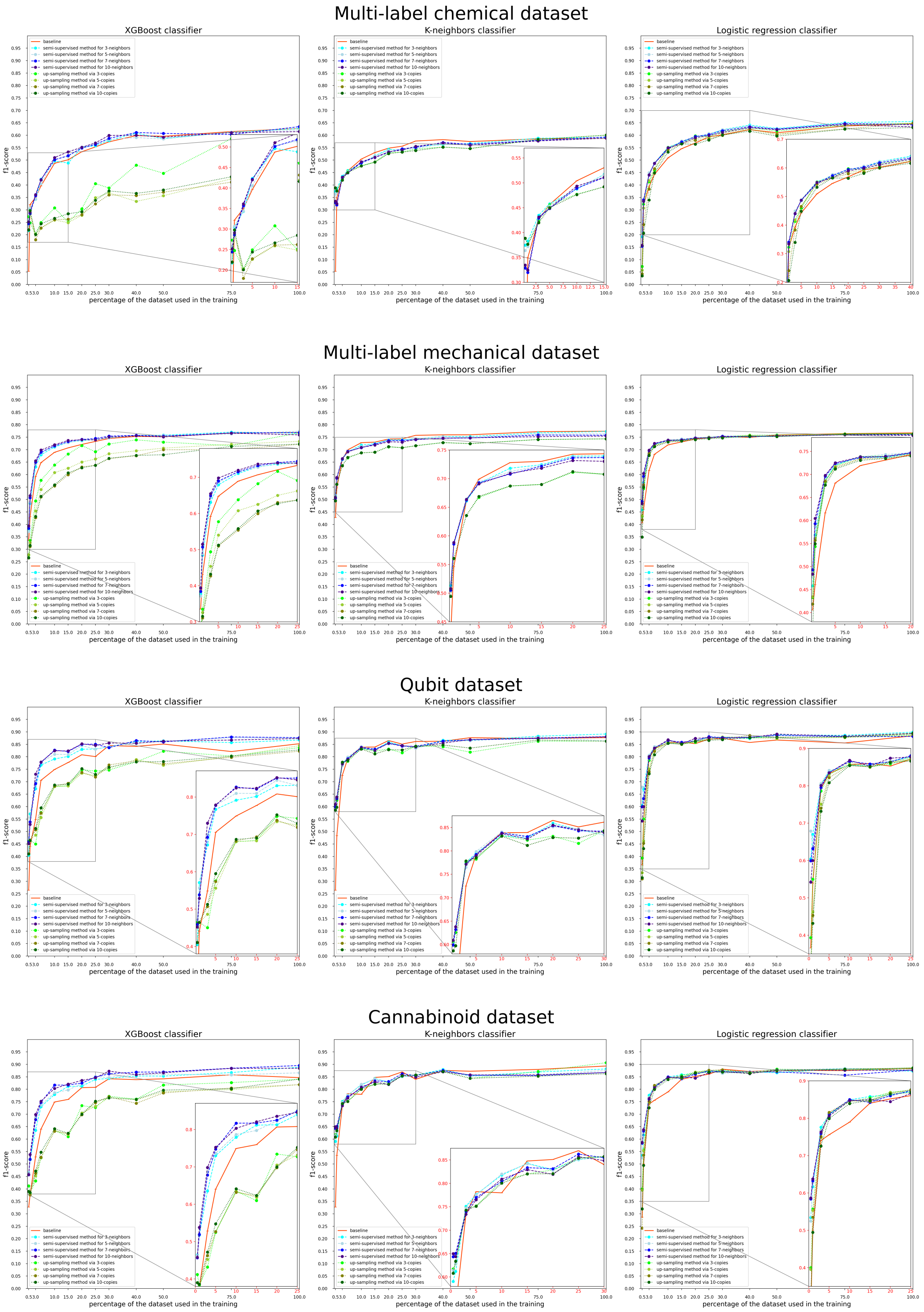


Figure 2: Evaluations from two multi-label and two binary datasets, where the X-axis denotes the percentage of the training dataset utilized during training, and the Y-axis reflects the F1-score calculated for the test set. Information about datasets can be found in Table 1. As observed from the results, among the three models utilized, the suggested semi-supervised method consistently outperformed up-sampling when applied to the XGBoost classifier, exhibiting score differences of up to 0.15 points for corresponding percentages. The semi-supervised method demonstrated efficiency for $p \in \{0.5, 1, 3, 5, 10, 15, 20, 25, 30\}$ in the context of logistic regression. However, when considering various datasets on average, up-sampling did not perform much worse (see Table 2 for the specific numerical values). The efficiency of both methods for the k-neighbors classifier was noticeable only for the lowest percentages $p \in \{0.5, 1, 3, 5\}$, while for higher percentages, they either maintained the F1-score at baseline levels or even yielded worse results.

References

[1] A. Krizhevsky, Learning multiple layers of features from tiny images, null (2009) 32–33. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[3] J. Lagus, E. Kotliarova, S. Björkqvist, Patent classification on search-optimized graph-based representations, in: Proceedings of the 4th Workshop on Patent Text Mining and Semantic Technologies, PatentSemTech'23, 2023, pp. 33–38. URL: https://ceur-ws.org/Vol-3604/paper2.pdf.

[4] IPRally: Custom patent classifier, 2024. URL: https://www.iprally.com/custom-classifier.

[5] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[6] J. E. van Engelen, A survey on semi-supervised learning, Machine Learning (2020).

[7] X. Zhu, Semi-Supervised Learning With Graphs, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 2005.

[8] T. Jebara, J. Wang, S.-F. Chang, Graph construction and b-matching for semi-supervised learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 441–448. URL: https://doi.org/10.1145/1553374.1553432. doi:10.1145/1553374.1553432.

[9] W. Liu, J. Wang, S.-F. Chang, Robust and scalable graph-based semisupervised learning, in: Proceedings of the IEEE, volume 100, 2012, pp. 2624–2638. doi:10.1109/JPROC.2012.2197809.

[10] X. Zhu, Semi-Supervised Learning Literature Survey, Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. URL: http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.

[11] A. Blum, S. Chawla, Learning from labeled and unlabeled data using graph mincuts, in: International Conference on Machine Learning, 2001, pp. 19–26. URL: https://api.semanticscholar.org/CorpusID:5892518.

[12] M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs, in: Learning Theory, volume 3120, 2004, pp. 624–638. doi:10.1007/978-3-540-27819-1_43.

[13] 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.

[14] S. Harris, Gold standard for the evaluation of machine classification of patent data, 2019. URL: https://github.com/swh/classification-gold-standard/tree/master.

[15] 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.

[16] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 785–794. URL: http://doi.acm.org/10.1145/2939672.2939785. doi:10.1145/2939672.2939785.

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

[18] S. Björkqvist, J. Kallio, Building a graph-based patent search engine, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 3300–3304. URL: https://doi.org/10.1145/3539618.3591842. doi:10.1145/3539618.3591842.

[19] E. Bernhardsson, Annoy: Approximate nearest neighbors in c++/python, 2023. URL: https://pypi.org/project/annoy/, Python package version 1.17.2.