EPFL

Semester project: Classification of virtual patent marking web pages using machine learning techniques

Mariko Makhmutova

Supervisors:
Gaétan de Rassenfosse
Samuel Arnod-Prin

Innovation and IP Policy Laboratory, EPFL

June 9, 2020

Contents

Abstract

Virtual patent marking (VPM) holds a great importance in modern intellectual property. It allows patent holders to easily communicate to the public that their technology is patented. Thus, the automatic detection of virtual patent marking information on the internet is crucial to building a public database that allows such information to be more easily accessible. Such automatic detection is possible using machine learning classification models. In this work, we design and implement an ensemble classification model that uses natural language processing and statistical analysis to detect whether a web page contains VPM information or not. The main purpose of this work is to use the final classifier to automate the creation of IProduct, a public VPM database built by the IIPP lab at EPFL.

1 Introduction

Virtual patent marking (VPM) is crucial in the context of intellectual property, particularly in the digital age, as it allows patent holders to communicate to the public that their product or technology is patented. Thus, having a public database that links products with their corresponding patents allows to facilitate any issues regarding patent breaches. This is exactly what IProduct, a project headed by Prof. Gaétan de Rassenfosse of the IIPP lab at EPFL aims to do.

In this work, we present the design and implementation of a machine learning classification model that determines the presence of virtual patent marking information in a web page. The classification model makes use of natural language processing, as well as statistical properties of HTML and URL properties to output the probability that a web page belongs to one of three categories, depending on the type of VPM information the web page contains.

This model will allow us to automate the detection of web pages containing VPM information, which will make the construction of the IProduct database significantly more simple and efficient.

2 Related work

Due to the massive popularity of the World Wide Web, web page classification is a topic that has been widely studied in recent years, for purposes such as ad detection, spam detection, and general topic classification. We present some works that study similar supervised web page classification problems.

Karthikeyan et al. acknowledged one of the major challenges in web page classification in their paper [1]: the lack of structure across web pages. Their work uses textual information found in the HTML of a web page for classification. We employ a similar approach in this work, by searching for category-specific keywords. However, we further extend our text-based approach by using word embeddings to get the semantic context of the text surrounding these keywords.

Joshi et al. used a similar approach [2] as the one presented in our work regarding the use of lexical features of URLs. However their goal was to use these features to detect malicious URLs, rather than for web page classification. We note that, as Karthikeyan et al. had acknowledged in their work regarding HTML, Joshi et al. also commented on the challenges posed by the lack of structure in URLs, hence focusing on the lexical features.

Another work that used a stacking framework with ensemble methods for web page classification was done by Elsalmy et al. [3]. The classification results presented in their work served as one of the motivators to use a stacking framework approach in our work.

We also recognize the importance of Albert Calvo's Master's degree thesis project [4] to the current project. Calvo's project served as a basis for our work, in particular for the definition of web page categories. We also use a subset of the data set constructed by Calvo to train and test our classification models.

3 Implementation

3.1 Data preprocessing

3.1.1 Data format

The initial data used to build our VPM web page classifier was a set of (HTML, URL, category) tuples for each web page. We limit the scope of our work to data that adheres to the following assumptions: the web pages are in English, and the HTML documents are sufficiently comprehensible when parsed. We suppose that a document is sufficiently comprehensible if the module used to detect the language of the web page has a high confidence level that the page is in English, meaning that a large portion of the page consists of English text.

These assumptions are satisfied for every sample in the data set used in this project. Nonetheless, we adapt our implementation to situations where samples may not fulfill either or both of the assumptions. In the data preprocessing phase of the classification model implementation, samples that are detected as not fulfilling either or both of the assumptions are discarded from the data set. Since the model is not designed to take into account such data, making predictions on such samples would cause unreliable results. However, taking into consideration samples that do not fulfill these criteria could be part of a further work, as mentioned in Section 4.

3.1.2 Categories

We consider three categories for web pages —simple VPM pages, hidden VPM pages and non-VPM pages. These categories can be interpreted as follows:

- **Simple VPM pages (SVPM)**: web pages whose main purpose is to present patent information about products. Such pages usually consist of lists or tables, linking products and their corresponding patent numbers.
- **Hidden VPM pages (HVPM)**: web pages with product information that contain VPM information, but the page's main purpose is different. This includes product catalogues, product user guides, press releases, e-commerce sites, as well as pages that have footers that contain patent information for the company. It is usually more difficult to link the patented item's name to the number for such pages.
- **Non-VPM pages (NVPM)**: web pages with no VPM information. We can further break this category down into the following sub-categories, although we do not make this distinction in the scope of this work:
  - news: news articles, blog posts about topics such as IP policy and patent laws
  - document: annual reports, terms and conditions pages, complaints and lawsuits
  - other: uncategorized according to above categories, pages on irrelevant topics (social media web pages, company pages with no patent information, patent attorney web pages, etc.)

3.1.3 Data collection and sources

The data used to train and test the final model was collected and categorized based on one of three techniques: manual web page collection and categorization, selection of web pages from a web crawler, and a previously categorized data set used in [4].

A web crawler was used to collect a data set of web pages containing patent information prior to the start of this project. Initially, a set of Bing search queries was generated using a list of United States Patent and Trademark Office (USPTO) applicants, combined with terms such as "patent" or "virtual patent marking". The crawler followed the results of the Bing search queries to various predefined levels of depth, and the resulting set of web pages was filtered to contain web pages that contain the word "patent" in the HTML document.

We manually categorized a subset of this data according to the categories described in Section 3.1.2. Similarly, we manually selected web pages from the internet, for instance non-VPM pages that could be mistaken for VPM pages, such as press releases. This gave us a total contribution of 263 pages.

We also worked with a subset of the data set made by Albert Calvo in [4]. The original data set contained non-English web pages, which does not adhere to the initial assumptions described in Section 3.1.1, so these pages were removed. Also, there were many repetitions of the same domain names, thus giving pages with similar HTML format. We wanted to maximize the variety of the data set format, and reduce the probability of overfitting to a specific domain, so we reduced the number of repetitions of domain names to at most three per domain name. We used 836 of the initial 1318 of Calvo's data set as part of the training data for our work.

Calvo's data set was also categorized in a different way than we categorize data in this work. The data was separated into six categories, which we map onto our three categories as precisely as possible. We use a different category assignment because as the data set is limited in size and imbalanced, we want to avoid under-fitting or over-fitting for certain categories. The category mappings can be found in Appendix A.

The final aggregated data set contains 1047 samples, consisting of 207 HVPM pages, 358 NVPM pages and 482 SVPM pages. Despite the data set not being balanced in terms of category distribution of the samples, we hope that this will not prevent the final classification model from performing well on external data, as it will be particularly well trained in distinguishing SVPM samples.

3.2 Classification model design and implementation

3.2.1 Performance metrics

Determining the performance of a classification model in our project is split into two components. The first component that we consider is the model's ability to categorize a web page as containing VPM information or not. In other words, we consider the HVPM and SVPM categories to be a single category that we call "VPM", which represents the positive class. The NVPM category thus represents the negative class. The second component used to determine a model's performance is its ability to categorize each web page as belonging to the three original

categories: HVPM, SVPM and NVPM.

We ultimately want to minimize the number of VPM false negatives that a model prediction outputs. This is defined as the number of web pages that are classified as NVPM, whereas they should have been classified as HVPM or SVPM. Minimizing the number of VPM false negatives means that we minimize our loss of potentially interesting web pages that contain VPM information.

We also want to optimize VPM recall, which gives the percentage of pages that are VPM (HVPM or SVPM) that are correctly identified as VPM pages.

The third performance metric that we try to optimize for is the macro-averaged F1-score, which is a variant of the F1-score for multi-class classification problem. In binary classification problems, the F1-score is defined as the harmonic mean of precision and recall. In the context of multi-class classification, we use the macro-averaged F1-score, or macro F1, which is the arithmetic mean of the F1-scores for each class. We do this to optimize for an overall better balance of precision and recall in each class. Mathematically, the macro-averaged F1-score is given as follows:

$$\begin{aligned}\text{macro F1} &= \frac{1}{3}\big(F1_{SVPM} + F1_{HVPM} + F1_{NVPM}\big) \\ &= \frac{1}{3}\Big(\frac{2\,P_{SVPM}\cdot R_{SVPM}}{P_{SVPM}+R_{SVPM}} + \frac{2\,P_{HVPM}\cdot R_{HVPM}}{P_{HVPM}+R_{HVPM}} + \frac{2\,P_{NVPM}\cdot R_{NVPM}}{P_{NVPM}+R_{NVPM}}\Big)\end{aligned}$$

where $P_{category}$ denotes the precision of a category and $R_{category}$ denotes the recall of a category.

The three main performance metrics taken into consideration when evaluating model performance are thus: number of VPM false negatives, VPM recall, and the macro-averaged F1-score. We also compare VPM accuracy and the multi-class accuracy to have a better understanding of how much distinguishing between the three categories impacts the model compared to simply performing a binary VPM categorization.

3.2.2 Baseline model

We designed a baseline model for the analysis using features extracted from the web page URL. The baseline model allows us to see how much a simple solution can be improved upon using natural language processing and other more complex machine learning techniques. We used a random forest classifier, which is an ensemble method that uses multiple trees in the training, thus reducing the output variance of the estimator [5]. Since it is a tree-based model, this makes it easier to evaluate which input features are more important to the model's decision making, as they are clearly assigned a score. We exploit this property when designing subsequent models.

The baseline model is thus a random forest classifier that uses the optimal parameters without any parameter testing. The input features for the baseline model are the following:

- Dummy variable representing the presence of "patent" in the URL
- Dummy variable representing the presence of "product" in the URL
- Dummy variable representing the presence of at least one of "news", "blog" or "article" in the URL

- The number of characters in the URL

We present the results of the baseline model, compared to the more advanced models described in the subsequent sections in Section 3.3.

3.2.3 Feature model

Extending from the baseline model that only takes into account URL-related features, we consider a model that takes into account a combination of features extracted from the web page URL and its corresponding HTML document. We refer to this as the feature model.

The input features used to train this model originated from the initial statistical data analysis performed on the overall data set[1]. We used recursive feature elimination and cross-validation to determine the best subset of features to use to predict a web page category. The final set of features used consists of the following:

- Dummy variable representing the presence of "patent" in the web page footer
- Dummy variable representing the presence of "patent" in the URL
- Dummy variable representing the presence of "product" in the URL
- Dummy variable representing the presence of at least one of "news", "blog" or "article" in the URL
- The number of characters in the URL
- Number of times the term "patent" (includes "patents", "patented", etc.) appears in the HTML
- Number of times 35 US Code § 287 (Limitation on damages and other remedies; marking and notice) is mentioned in the HTML
- Number of times terms related to a product description (e.g. "user guide", "product specifications") are mentioned in the HTML
- Dummy variable representing the presence of a date format detected in the HTML
- Number of times a date format is detected in the HTML

We present the results of the feature model with optimally tuned parameters in Section 3.3.

3.2.4 Vector model

We observe that each of the three categories tend to have different words around the term "patent", as well as around patent numbers. We display some excerpts, as well as their categories in Table 1. One can see from these samples that the contexts of the excerpts can vary significantly between categories. We use this observation as a basis for the design of the vector models.

The procedure to construct such a model is as follows:

[1]The reader is invited to read the Jupyter Notebooks provided alongside this report to see the complete analysis.

HTML Excerpt	Category
"Support Forum Portions licensed under the following DDB Technologies, LLC U.S. Patents 5,526,479, 5,671,347, 6,204,862, and/or 7,373,587. Other patents pending."	HVPM
"Woodguard products are protected by U.S. Patent #6,231,994 and patents pending"	HVPM
"The following list may not include all products, patents or pending applications."	SVPM
"Similac Advance 2 oz. Liquid Commodity Number: C0096; C0105;C0296 * Patent No.1: 7,829,126"	SVPM
"Willis Patent Services uses patent lawyers and firms to act as a your agents to represent you in the various international and regional patent offices"	NVPM

Table 1: HTML excerpts containing "patent" term for different categories.

1. Use a regular expression pattern to extract the surrounding terms for each pattern occurrence in each sample. We refer to these sets of surrounding terms as "documents".

2. Process each document to extract the overall context: remove stop words and terms containing non-alphabetic characters, and lemmatize the remaining terms.

3. Use a TF-IDF model to determine the importance of each term extracted over the whole data set.

4. Use pre-trained 300-dimensional GloVe vectors that represent word embeddings to calculate the overall TF-IDF weighted vector representation of each document.

5. Use the vector representation of the sum of all documents extracted for a given web page as the input to the classifier.

We build two such models. The first uses vector representations of text surrounding the term "patent" as inputs. We refer to this model as the patent vector model. The second model uses vector representations of text surrounding numbers of the form "X,XXX,XXX" or "X XXX XXX" as inputs, because this is the format in which US patent numbers are most often represented. We refer to this model as the patent number vector model.

One issue with such models is that when we match on regular expression patterns, not all samples find at least one match, if the pattern cannot be found in the HTML document. Thus, we must work with reduced data sets. More specifically, there are 763 out of 1047 web pages that contain at least one patent number match, and 1007 that contain at least one "patent" match. To be consistent, the vector models are still trained on 70% of the resulting data sets and tested on the remaining 30%, as the other models are.

We present the results of the vector input models with optimally tuned parameters in Section 3.3.

3.2.5 Stacked model

We use a stacked model to improve the performance of the feature and vector models. Stacking is an ensemble method that consists of training a new model that takes the predictions of other models as inputs. Stacking models have yielded impressive results in literature [3] and in practice, with two of the top performing algorithms in the Netflix Prize competition being forms of stacking algorithms [6].

We implement a stacked model as follows: three initial models —the feature model, the patent vector model and the patent number vector model —are trained on the available URL and HTML input data. Then, the probability predictions of each model are used as inputs to the stacked model. The probability predictions represent the probability that a web page belongs to each of the three categories according to the estimator.

We note that we use probability predictions of the initial models as inputs to the final estimator, rather than simply the category predictions. Doing so allows us to consider the confidence of the each input model for each sample, and this has been shown to be a more effective approach [7].

We tested the stacked model using three classification models —a random forest, and two implementations of gradient boosting decision trees algorithms: XGBoost [8] and LightGBM [9]. We present the best results obtained using parameter tuning for each of these models in the following section.

3.3 Results

We present the test results of the initial models, compared to the baseline URL classifier in Table 2. We train on 70% of the data set and test on the remaining 30%.

One can see that the feature model has a significant improvement on the baseline classifier, in all performance metrics. We remark that the feature model is trained in the order of seconds, so it is an inexpensive model to implement in terms of time.

Both vector models both offer a significant improvement over the baseline model in terms of VPM recall and multi-class accuracy. On their own, however, the vector models do not have results as impressive as the feature model does. Nonetheless, we observe the results of the stacked model that uses the outputs of the feature model and both vector models as input data.

We test three implementations of the stacked model, using the following ensemble classification algorithms: random forests, XGBoost and LightGBM. Each

Model	**Test size**	**VPM false negatives**	**VPM recall**	**VPM accuracy**	**Macro F1**	**Multi-class accuracy**
Baseline model	315	33	0.841	0.752	0.645	0.689
Feature model	315	18	0.913	0.835	0.766	0.794
Patent number vector model	222	18	0.881	0.838	0.665	0.716
Patent vector model	306	12	0.941	0.843	0.684	0.735

Table 2: Test results of initial models compared to the baseline model.

Model	VPM false negatives	VPM recall	VPM accuracy	Macro F1	Multi-class accuracy
Random forest	19	0.908	0.895	0.836	0.860
LightGBM	16	0.923	0.908	0.842	0.870
XGBoost	20	0.903	0.898	0.831	0.863

Table 3: Stacked model performance on test set using different ensemble algorithms.

of these algorithms have shown promising results in literature and in practice [8,9]. We present the results of each of the ensemble models, tuned to their optimal parameters, in Table 3. The optimal parameters of each algorithm can be referred to in Appendix B.

One can see that the LightGBM classification model performed better than the random forest and XGBoost, although all three models had impressive results. Each of the models was trained on 70% of the data set and tested on the remaining 30%, consisting of 315 samples. The LightGBM model outperformed all of the other models with the same test size in every performance metric. The results are particularly impressive when comparing to the baseline model. LightGBM model achieves an increase of 0.19 in the macro-averaged F1 score, an almost 15% increase in VPM accuracy, as well as an 11% increase in multi-class accuracy over the baseline model. We thus use a LightGBM stacked model in the practical implementation of our work.

4 Discussion

The primary goal of this work was to design and implement a classification model that is able to detect the presence of virtual patent marking information in a web page. We achieved this goal using various machine learning techniques in natural language processing, statistical analysis and ensemble modelling.

A major challenge in this work was information extraction from HTML. The lack of consistency in web page structure makes the use of HTML meta tags and structural elements impractical. Thus, we limited the scope of the work to textual information extraction using regular expression patterns, as well as statistical observations of URL and HTML patterns.

There are multiple further extensions to the work that can be done to improve the classification model performance:

- One could implement a similar classification model for different languages: most of the modules used in the implementation of the classification model are adaptable to other languages, so this could be an easily attainable extension of the current work.

- One could extend the patent number vector model to take into account other patent number formats, if such a detection pattern is found. One must note however, that the regular expression matching phase of the model training is the bottleneck of the training process. Thus, simply employing a more complicated regular expression pattern would not be a favourable approach in terms of time complexity.

- One could consider other regular expressions to extract text to build vector models to test whether using other expressions would enhance the classification model performance.
- One could employ keyword extraction techniques on the whole HTML document, to further observe category-based patterns and use the insights to build an improved classification model.

Given the results presented in Section 3.3, we believe that the implementation presented in this work will reliably serve as a starting point for the automation of the IProduct database creation. We hope that this work will be useful in the acceleration of the database growth, and in doing so, that it will help make virtual patent marking information more accessible to the public.

References

[1] I. D. Karthikeyan, R. R, and S. Karuppasamy, "Generating best features for web page classification," *Webology*, vol. 5, March 2008.

[2] A. Joshi, L. Lloyd, P. Westin, and S. Seethapathy, "Using lexical features for malicious URL detection - a machine learning approach," *ArXiv*, vol. abs/1910.06277, 2019.

[3] F. Elsalmy, R. Ismail, and W. Abdelmoez, "Enhancing web page classification models," in *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016*, A. E. Hassanien, K. Shaalan, T. Gaber, A. T. Azar, and M. F. Tolba, Eds. Cham: Springer International Publishing, 2017, pp. 742–750.

[4] A. C. Ibanez, "Classification of virtual patent marking web pages using machine learning techniques," Master's thesis, Swiss Federal Institute of Technology Lausanne, Switzerland, 2018.

[5] "Ensemble methods." [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html

[6] J. Sill, G. Takács, L. W. Mackey, and D. Lin, "Feature-weighted linear stacking," *CoRR*, vol. abs/0911.0460, 2009. [Online]. Available: http://arxiv.org/abs/0911.0460

[7] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *CoRR*, vol. abs/1105.5466, 2011. [Online]. Available: http://arxiv.org/abs/1105.5466

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

Appendix A Category mapping

Albert Calvo's data set consists of the following six categories: simple VPM, home VPM, other VPM, news, lawsuit and other. We invite the reader to study Calvo's report [4] to further understand the category definitions and breakdowns.

We group his six categories into the three categories presented in our work as follows:

- **SVPM**: simple VPM
- **HVPM**: home VPM, other VPM
- **NVPM**: news, lawsuit, other

Appendix B Optimal parameters

We present the optimal parameters used for the ensemble classification algorithm implementations to obtain the results presented in Section 3.3.

- Random forest: `class_weight=balanced`, `max_depth=12`, `min_samples_leaf=2`, `n_estimators=300`, `random_state=0`
- LightGBM: `class_weight = balanced`, `max_depth = 5`, `num_leaves=12`, `n_estimators=35`, `random_state=0`
- XGBoost: `max_depth=3`, `eta=0.00001`, `n_estimators=25`, `random_state=0`

We note that the optimal parameters for intermediate models are used in the practical implementation of the work by default.