CS230

Predicting Patent Outcomes with Text and Attributes

Ryan Lee
Department of Mechanical Engineering
Stanford University
rlee5@alumni.stanford.edu

Marcos Torres
Department of Computer Science
Stanford University
marcost7@stanford.edu

Abstract

Patents are expensive and can take years to be issued. Despite the large investments companies make producing patents, there are no predictors of patent outcomes that account for both the text of the patent and the attributes of the patent. We implement two models, one utilizing the text of the patent and one utilizing the metadata, or general attributes of the patent. We focus on optimizing our models to predict whether a patent will be issued or not. For the text-based model, we implement a convolutional neural network to obtain an F1 classification score of 0.776, and for the metadata model, we implement a 6 layer neural network with the metadata features and obtain an F1 score of 0.832. [1]

1 Introduction

Despite the steep average price of $20,000 charge per patent by a patent firm, it is often difficult for both the inventor and patent agent to know when or if a patent may be granted. We are trying to predict the outcome of a patent as either granted, delayed substantially, or abandoned. To make this prediction reliably, we are building two types of prediction models. The first has as input a set of metadata features that take into account the circumstances of the patent submission. The second considers the actual text of a patent application.

For the metadata model specifically, we use an input of 22 metadata features, including the application's examiner, the United States Patent Classification (USPC) class, USPC sublcass, and patent firm. Given these inputs we train and validate on 4,506,331 patent examples from 2000 to 2014. We then train and validate these examples on a 4 layer deep neural network. We use XGBoost, a gradient boosted tree algorithm, as our baseline for this model. For the text data, we train a convolutional neural net (CNN) on the first 500 words of the inventive claims text for 177,248 examples.

Previous work includes a course project in CS229A, Applied Machine Learning, that involved creating a set of 7 features to be trained by a logistic regression and a 1 layer neural network. The logistic regression performed on a test set with an F1 score of 0.8 and the shallow neural network performed with an F1 score of 0.78. While the same dataset, USPTO Patex, was used in both, there are different features extracted from the dataset in the current model.

2 Related work

Since disputes over intellectual property rights can make or break companies, many researchers have examined different aspects of the patent process. For example, some researchers have focused on identifying and analyzing the grant rates of individual examiners, such as the company Patent Bots [2]. Another company, Juristat, predicts the examiner that a given patent might be assigned to [3]. Others, such as the Fung Institute, have used natural language processing and machine learning

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

to predict the technology area classification of the patent before it is assigned by the USPTO[4]. Others have tried to predict post-grant review outcomes for patents that may be litigated [5]. Colleen Chien explored predictors of future patent litigation with traditional statistical analyses [6]. The most relevant research to our work is the thesis of David Winer, who like Chien tried to predict post-grant litigation and review outcomes for patents, but used support vector machines and random forests.

We found a lot of traditional data crunching, stats and machine learning, but ultimately found more articles on the future of deep learning patents than on using deep learning to learn about patents themselves. These studies also avoid the direct question of whether a patent application is likely to be granted in the first place, suggesting that our prediction task and our approach of combining text data and metadata to make a prediction is novel.

3 Dataset

The metadata dataset, as well as the labels for the text-based learning model, were extracted from USPTO Patent Examination (PatEx) datasets. We downloaded 4,506,331 examples from 2000 to 2014. For both models, the labels or result for each example was determined from the "disposal type" specified by the PatEX example set. We assign a 0 to abandoned patents (those rejected by the USPTO), a 1 for issued patents, and a 2 for pending patents that are awaiting responses.

Figure 1 shows the data flow we used to train the text-based and metadata model respectively.

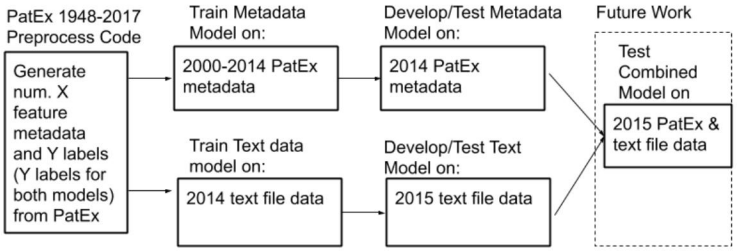

Figure 1: Overview of data transformation and data use in model training and model validation

For our text classifier, we download Public Patent Application Information Retrieval (PAIR) xml files hosted by Google and the USPTO, from which we extract claims text and application ID. We assume that claims are the most important section with respect to the likelihood of an application being approved. We use 177,248 examples from the first 6 months of 2014 and cross-list with our metadata dataset to get corresponding disposal type labels.

3.1 Metadata Features

To train a metadata supervised learning model with PatEx, a set of 22 features was created from four attributes given for each patent on public record: the given examiner, the given firm, the given USPC classification and the given USPC subclass. Shown below are a few examples of the numerical features generated:

- Feature 1: Number of patents assigned to your examiner's art unit
- Feature 2: Number of patents previously submitted for the example's USPC class
- Feature 21: Percentage of issued patents out of total patents reviewed for patent examiner
- Feature 22: Percentage of issued patents out of total patents submitted for patent firm

To prevent data leakage it is important to prevent future data from being used to predict past examples, so the examples have been assigned chronologically to training, cross-validation, and test sets. Thus, metadata feature set was split into a training, cross-validation and test set with the first 60% of examples used for training, 20% used for cross-validation, and the remaining 20% used for test performance. All the metadata features were normalized with a mean of zero and a standard deviation of one.

Figure 2 shows the distribution of labels in the 2014-2015 metadata for the training, cross-validation, and test sets. This is also representative of the label distribution for the text-based model.

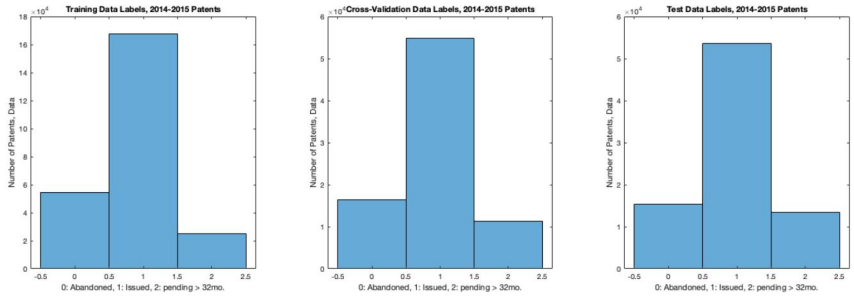

Figure 2: More issued patents than pending or abandoned in dataset

3.2 Text-based Features

To obtain a claims text dataset to train on, we downloaded publicly available patents offered from the USPTO bulk download web page[7]. These are available in XML format; we extract the claims text and unique application identifier. Using the unique application ID, we matched each patent text with the corresponding labels from the PatEx dataset, resulting in 177,248 labeled examples. Our examples are skewed in favor of issued patents, so we up-sample to balance the dataset before splitting into a 90-5-5 train-dev-test split.

We parse each example into lists of words and retain only the first 500 words to make computation feasible. Each example is truncated or padded to standardize to this length. To learn word embeddings, we extract a vocabulary set of all words in our training data. Using the vocabulary list, we map each example to a vector of word indices. This allows us to have a well-formed matrix for use with an embedding layer.

4 Methods

4.1 Metadata Neural Network

A neural network with Relu activation for hidden layers and a softmax activation for the final output layer with cross-entropy loss function was implemented for the multi-class classification and compared to the performance of a boosted tree classifier XG Boost.

A fully connected neural network trained and evaluated consists of an input layer of 22 features, a 5 hidden layers, and an output layer of size 3 for each classes. A softmax activation function was applied to the output layer acquire a probability of an example belonging to each of the 3 output classes. As shown in figure 3, a categorical regularized cost function used for forward and back-propagation to obtain weights with the neural network. An Adam Optimization algorithm with learning rate$= 0.001, \beta 1 = 0.9, \beta 2 = 0.99$ was used to learn the weights for the neural network. An Adam Optimization is a combination of a Momentum algorithm and a Root Mean Squared (RMS) Propogation, and when implemented results in an adaptive learning rate based on the update history of the parameters being trained [8].

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

Figure 3: Regularized Cost Function for One vs. All Classifier

4.2 Benchmark XGBoost Tree Algorithm

As a benchmark for performance of the neural network model, we trained and evaluated an XGBoost model. XGBoost stands for extreme gradient boosting, and is a variant of a decision tree algorithm optimized for model performance and speed. With the XGBoost decision tree algorithm, we were also able to extract a measure of feature importance to the classification task.

4.3 Text-Based CNN

We developed a simple CNN architecture for making predictions based on claimstext. This network consists of an embedding layer with embeddings of length 256 followed by 4 parallel convolution layers of kernel sizes 2,3,4 and 5. Each of these is followed by a maxpool layer, and the outputs of these are concatenated before being passed to a dense layer with hidden size 50 and ReLU activation. This is followed by a final linear layer with two outputs and softmax activation.

5 Experiments/Results/Discussion

As there are more issued patents than pending or abandoned patents in the dataset, the performance of the neural network and text-based model on the cross-validation set was not evaluated with accuracy, but with precision, recall, and the F1 score, as defined:

$$precision = \frac{tp}{tp + fp} \qquad recall = \frac{tp}{tp + fn} \qquad F1 = 2 * \frac{precision * recall}{precision + recall}$$

5.1 Metadata Neural Network Performance

The metadata model was trained and evaluated using 412,401 patent examples with the metadata features from 2014 to 2015. As shown in Figure 2, a different set of examples was used to train the neural network, here we report the precision, recall, and F1 score of the model predictions for the test set examples, in Figure 4. Additionally, performance for the multi-class classification prediction task was assessed with the Receiver Operator Characteristic Area Under the Curve and compared with a random classifier, as shown in Figure 8.

Test Performance	precision	recall	f1-score
Abandoned or Not	0.493	0.156	0.237
Issued or Not	0.748	0.937	0.832
Pending or Not	0.563	0.435	0.491
micro avg	0.709	0.709	0.709
macro avg	0.601	0.510	0.520
weighted avg	0.670	0.709	0.665

Figure 4: F1 Score of 0.832 for 2014-2015 Metadata Neural Network

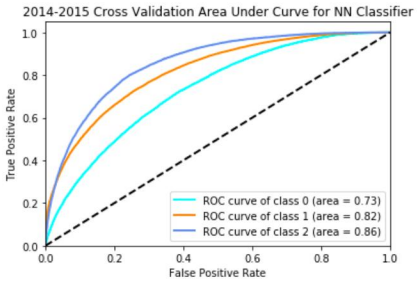

Figure 5: ROC Area Under Curve of 0.82 for 2014-2015 Metadata Neural Network

5.2 XGBoost

Our benchmark algorithm, the XGBoost tree algorithm yielded an F1 score of 0.838 for the classification of issued or not, comparable to the performance of our metadata neural network on the same classification problem.

	precision	recall	f1-score
0	0.703	0.569	0.629
1	0.789	0.870	0.828

Figure 6: F1 Score of 0.838 for 2014-2015 XGBoost Model

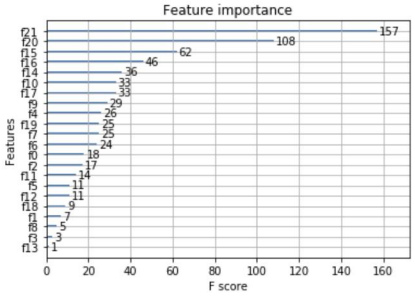

Figure 7: Firm Success Rate [f21] and Examiner Allowance Rate [f20] Most Important Features

5.3 Text-Based CNN

We trained our text-based CNN for 10 epochs, obtaining an F1 score of .776 and AUC of 0.803. We chose F1 and AUC over accuracy because we had issues with the classifier predicting only one class before balancing our dataset. We used a learning rate of 0.001 with Adam optimization, having had an exploding gradient issue with unbalanced data. We tried different filter sizes and dense layer sizes before settling on the architecture described above.

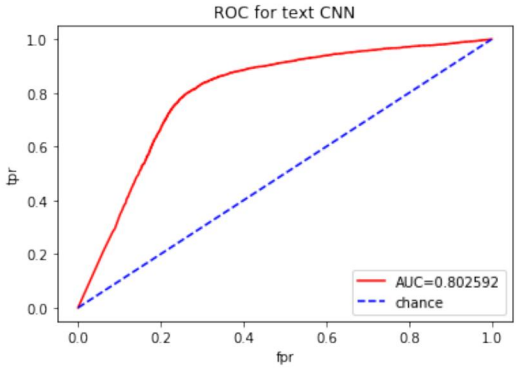

Figure 8: ROC curve for text-based CNN

6 Conclusion/Future Work

We found that the success rate of the legal firm used to submit a patent application and the allowance rate of the examiner reviewing an application are most predictive of whether a patent is issued or rejected. This makes sense; we can imagine that a successful firm behaves as a classifier in its own right - a good firm not only helps get a patent accepted but also decides when an application is even worth considering. We also found that the text of the first few claims can be predictive. We assume our model is picking up on either the technicality of the claims, the writing style, or the field of the invention.

Looking towards the future, we would like to concatenate our models to see if we can achieve better predictions by looking at both the text of a patent application and the circumstances of its submission. We would also like to explore the text of other sections within an application.

7 Contributions

Ryan was responsible for creating the metadata feature set from the USPTO public database, developing the metadata neural network model, and implementing the XGBoost algorithm for identifying feature importance and benchmarking the neural net performance. Marcos focused on developing

and training a CNN for text-based prediction of patent outcomes. This involved creating a pipeline for extracting claims text from XML files, generating a vocabulary list, standardizing text length, and mapping text to indices for embedding. We both helped debug each other's work and feel satisfied in our share of work.

For inspiration in model architectures and general NLP pipelines, we reference Marcos' prior CS224N class project with Rohan Bais and CS224N coursework. For inspiration with our metadata classifiers, we reference Ryan's previous work with patent classification in CS229A.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[2] J. O'Neill, "Predicting future patent outcomes." `https://www.ipwatchdog.com/2018/05/30/predicting-future-patent-outcomes/id=97410/`, May 2018.

[3] "About juristat." `https://www.juristat.com/about/`, June 2019. Accessed on 2019.

[4] I. Atigui, E. Atlani, D. D. Clercq, B. Tan, and L. Fliming, "Patent analysis using machine learning." `https://funginstitute.berkeley.edu/capstone-project/patent-analysis-using-ml/`.

[5] D. Winer, "Predicting bad patents: Employing machine learning to predict post-grant review outcomes for us patents," Master's thesis, EECS Department, University of California, Berkeley, May 2017.

[6] C. Chien, "Predicting patent litigation," *Texas Law Review*, vol. 90, August 2011.

[7] U. S. Patent and T. Office, "United states bulk downloads bulk downloads patent texts." `https://bulkdata.uspto.gov`, June 2019. Accessed on 2012-11-11.

[8] S. Ruder, "An overview of gradient descent optimization algorithms." `http://ruder.io/optimizing-gradient-descent/index.html?fbclid=IwAR3bdVaSebUqdHEbFVK8ESCIv34E6g5s6Dz73UcFT4d0YhDyG24J0GhbD_w#adam`, June 2019. Accessed on 2012-11-11.