

Predicting Pad Patents

Tzuo Shuin Yew

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-66

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-66.html>

May 11, 2017



Copyright © 2017, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

University of California, Berkeley College of Engineering

MASTER OF ENGINEERING - SPRING 2017

Electrical Engineering and Computer Sciences

Data Science and Systems

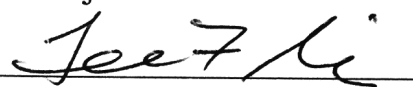
Predicting Bad Patents: Employing Machine Learning to Predict Post-Grant Review Outcomes for US Patents

Tzuo Shuin Yew

This **Masters Project Paper** fulfills the Master of Engineering degree requirement.

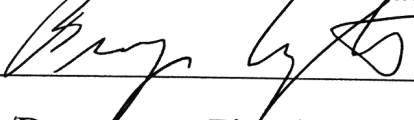
Approved by:

1. Capstone Project Advisor: Lee Fleming

Signature:  Date 5/8/17

Print Name/Department: Industrial Engineering & Operations Research

2. Faculty Committee Member #2: Vladimir Stojanovic

Signature:  Date 5/4/17

Print Name/Department: Electrical Engineering & Computer Sciences

Predicting Bad Patents

FINAL CAPSTONE REPORT

With the collaboration of:

- Dany Srage – IEBOR
- Joong Hwa Lee – EECS
- William Ho – EECS
- David Winer - EECS

Contents

- Executive Summary..... 2
- Chapter 1: Technical Contributions Introduction and Problem Overview 3
 - 1. Establishing a Connected Architecture 4
 - 2. Designing the Relational Database 6
 - 3. Planning an Automated Downloader..... 8
 - 4. Developing the Back End of Full-Stack Web Application 9
 - 5. Designing the Front End of a Full-Stack Web Application 11
 - 6. Optimizing the Front and Back End Relationship..... 13
 - 7. Conclusion 14
- Chapter 2: Engineering Leadership..... 15
 - 1. Introduction and Overview of Technology 15
 - 2. Marketing Strategy 16
 - 3. Competition 16
 - 4. Machine Learning Technology Trends and Strategy..... 19
 - 5. Ethical Challenges 19
 - 6. Conclusion..... 20
- Chapter 3: Overall Conclusions..... 21
- References 21

Executive Summary

Our capstone project sees us utilizing machine learning to address an issue of “bad patents”, whereby new patent filings with a high level of similarity to older filings cause disputes which in turn cause significant wastage of time and money on litigation in court. If every new patent filing could be compared against an entire database of older patents before a decision is made, that would solve the issue, but the sheer number of patents in existence make this unfeasible if using humans. Machine learning, on the other hand, allows such actions to be fully automated, and thus presents a viable solution to the problem. Over the past nine months, we developed a multi-part software solution involving large-scale data retrieval and analysis, the implementation and training of a support vector machine learning model, and the creation of an interactive graphical user interface.

In this paper, I discuss in detail my specific contributions, which include creating a connected infrastructure and writing the graphical user interface using web tools. My work in those areas enhanced team collaboration, improved operating efficiency, and provided our target audience with an interactive portal for actually utilizing our work. Other critical technical work, such as the model training and data parsing, is analyzed inside the papers written by my teammates David Winer, Joong Hwa Lee, Dany Srage, and William Ho. I then follow up with an analysis of how our project incorporates key facets of engineering leadership, such as marketing strategy, competitive analysis, and ethical forethought. Our combined efforts resulted in a novel and robust software solution which we believe satisfies the needs of our target audience. In the long run, we hope our work will help steer the United States patent system back towards its original purpose of fostering innovation, while also contributing to increased public interest in machine learning as a tool to solve broad problems (Yew, 2016).

Chapter 1: Technical Contributions

Introduction and Problem Overview

Our capstone project targets an issue within the United States patent system, where patent litigation often results in significant wasted time and money. The litigation is caused by accusations of infringement after a newly filed patent is seen to be similar to an existing patent, leading to subsequent court battles (Winer, 2016 p. 3-4). In recent years, a surge in technology-centric patent filings, thanks to related economic trends has made the problem worse (Winer, 2016 p. 2). To tackle this issue, we are developing a software application capable of comparing a patent's text against a database of older patents, outputting a predicted probability of invalidation, and highlighting which aspects of said patent would be responsible for the invalidation (Yew, 2016). Supervised machine learning techniques will be at the core of this project, especially the prediction. Our stakeholders include the United States Patent and Trademark Office, prospective filers of new patents, holders of existing patents, and law firms (Srage, 2016 p. 3-5). As a team, our goal is to deliver an effective, efficient software solution which not only satisfies our stakeholders, but is also easily transferrable to future teams with minimal overhead, since we expect teams in later years to take over and further improve upon this project.

In this chapter, I will discuss my technical contributions to our project, which include establishing a connected architecture, laying the groundwork for an automated downloader, and developing a graphical user interface through a web application. My teammates Joong Hwa Lee, William Ho, David Winer, and Dany Srage will focus on their respective responsibilities with the machine learning algorithms and data analysis. The work breakdown structure below (Figure 1), which David Winer created, summarizes the way we organized our tasks to achieve the final outcome.

Predicting Bad Patents

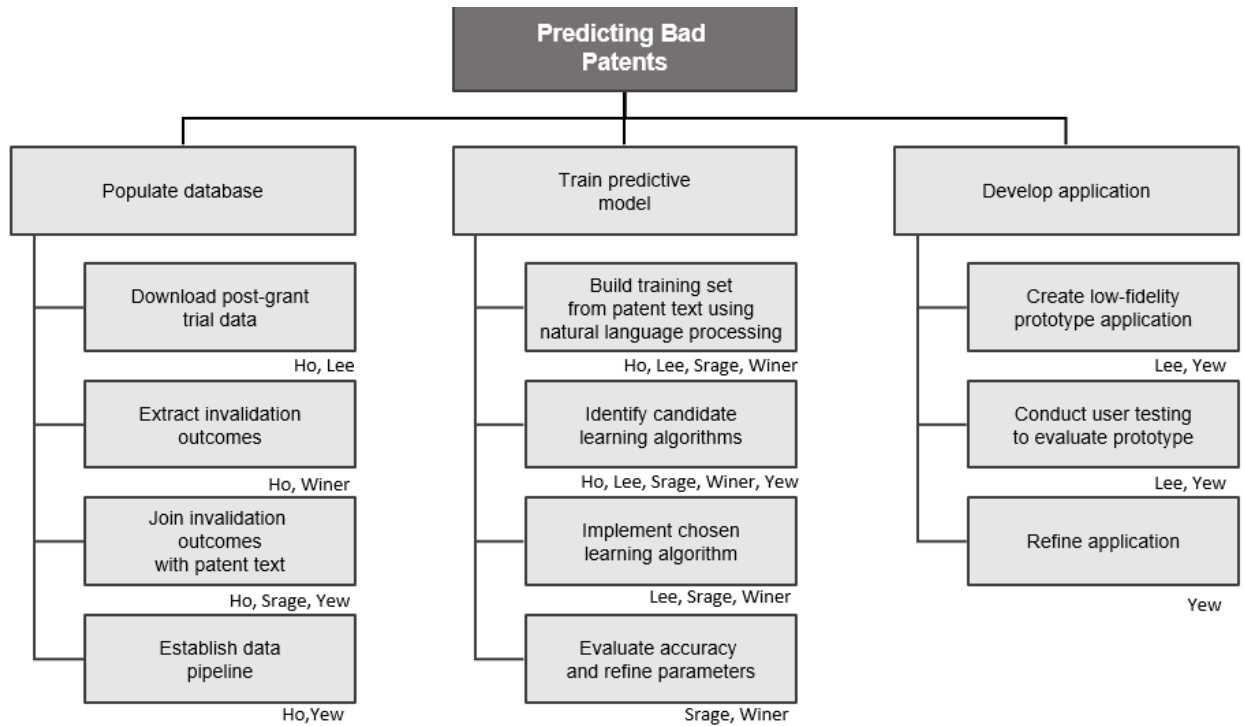


Figure 1: Work Breakdown Structure (Winer, 2017)

1. Establishing a Connected Architecture

Our project requires us to run computationally intense machine learning algorithms and parsing scripts for many hours without interruption, creating large output files in the process. If we were to utilize our personal computers for this work, our resources would be strained and it would compromise our productivity for work unrelated to the project. We would also run the risk of team members executing out-of-sync versions of the code base, which would produce incorrect results. Realizing the need for an environment more conducive to such responsibilities, I took on the role of chief architect and established a connected infrastructure for the team's use.

The first major component of this architecture is a Unix server. The server plays a critical role as an always-available computing environment, accessible to all team members from anywhere with an Internet connection. It is therefore the ideal place to store and execute all code for the project. We also make our work easily portable by consolidating all progress in this server, which helps future parties

Predicting Bad Patents

who may want to transfer our work to a new environment for expansion after we conclude our roles on the project.

A Unix server was chosen over a Windows server for both technical and economic reasons. The Unix command line provides a large number of functions not present on the Windows DOS command line, such as the ability to extract text from PDF files (elaborated upon in later sections), the ability to run optical character recognition on PDF files, the ability to parse text files using regular expressions, and more (Boston Computing Network, 2004). The Unix command line also benefits from a robust developer community, which maintains an extensive repository of third-party functions which can be installed on a Unix server in a single command. From a cost perspective, there was no question about whether or not to go with Unix, thanks to free trials available from Amazon Web Services.

I initially utilized a free tier account from Amazon Web Services, providing us with twenty gigabytes of server storage space. I later discovered that our code's output files would exceed this space before we were even halfway through the project. I then requested access to a higher-capacity server from the Fung Institute at the University of California, Berkeley and was able to shift our work onto a more spacious environment.

The Unix server's most valuable asset ultimately proved to be its raw computing power, thanks to a multiprocessor system with 256 GB of RAM. Figure 2 displays a chart comparing the performance of a TFIDF (Term Frequency Inverse Document Frequency) machine learning algorithm on this server to its performance on a standard consumer laptop with a dual-core processor and 8 GB of RAM. Execution completed on the server in roughly a tenth of the time it took on the laptop, with the added benefit of being able to run remotely at off-peak hours without interrupting unrelated work. The latter function was invaluable for training our support vector classifier algorithm, which Dany Srage elaborates upon.

Predicting Bad Patents

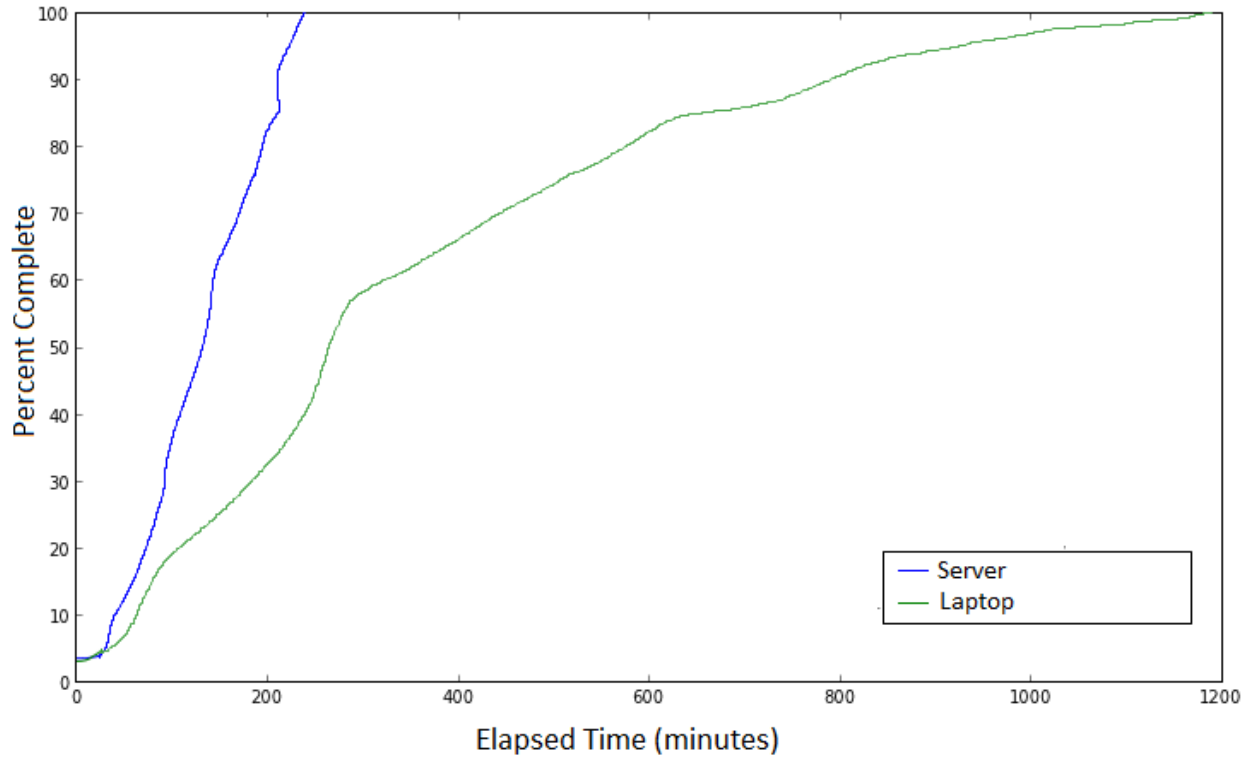


Figure 2: Comparing the task completion times of a server and a laptop

2. Designing the Relational Database

The second major component of this architecture is a relational database. Powered by MySQL, it provides a structured and organized container for storing the critical information parsed out of the patent text files, which facilitates future retrieval and analysis. Our architectural decisions mean that all future work will continue to be performed in this remote environment, and all future data will be stored in this relational database (Yew, 2016).

The alternative to a relational database was to leave all patent information inside text files and write scripts to sort and search those text files. We decided against this method because doing so would leave us with only one filter to search on – the name of each text file. If we wanted to find only a subset of patents fulfilling one particular characteristic, we would need to search each and every text file – a computationally expensive procedure. Instead, a relational database would allow us to link patent

Predicting Bad Patents

numbers with multiple filters of key information like decision outcomes and patent class, and index the records for instant lookups, allowing for much faster search queries.

I designed the schema for our relational database with normalization principles in mind. I aimed to have all tables conform to BCNF (Boyce-Codd Normal Form) to minimize data redundancy and optimize data retrieval performance. I chose BCNF because it is the strictest of all normalization techniques, having the most scalability and consistency, which aligns with one of our goals of making our work easily portable and transferrable to future teams (Ledoux et al, 1982). To align with BCNF, I split the schema into separate relations for patent information, patent decisions, and PTAB hearings, and placed a single superkey in every relation. Figure 3 displays the BCNF-aligned schema below.

Table: <u>ptab_cases</u>		Table: <u>patents_decision</u>		Table: <u>patent_info</u>	
Columns:		Columns:		Columns:	
<u>case_id</u>	varchar(15) PK	<u>patent_id</u>	varchar(255) PK	<u>patent_id</u>	varchar(15) PK
patent_id	varchar(15)	invalidated	tinyint(4)	filing_date	date
invalidated	tinyint(1)	claims_text	mediumblob	issue_date	date
denied	tinyint(1)			art_unit	varchar(7)
filing_date	date			examiner_name	varchar(127)
decision_date	date				
patent_owner	varchar(255)				
petitioner_name	varchar(255)				

Figure 3: Our BCNF-compliant table schema

To perform its functions, the database needs to be backed up by a database management systems. Since I was operating on a Unix server, my options were either MySQL or PostgreSQL, which both accept the same querying language but differ on internal workings such as query optimization strategies and crash recovery procedures. While PostgreSQL offered superior reliability and data integrity, I ultimately selected MySQL because it is the only management system for which an open-source Python library exists (Tezer, 2014). The Python code written by my teammates David J. Winer and Dany Srage requires frequent querying of the relational database, and the ability to integrate the open-source MySQL access library for Python is invaluable to our progress.

3. Planning an Automated Downloader

The first phase of our project required us to download tens of thousands of files from eFOIA, a website which hosts patent claim data in PDF format. Performing this task manually was out of the question, since it would take many hours and no member of the team could spare that much personal time. Additionally, we needed to extract the raw text from those PDF files to programmatically analyze them, since our tools were unable to comb through stock PDF files. I took responsibility for designing a new downloading and parsing toolset that would fully automate both tasks. My teammates Joong Hwa Lee and William Ho performed the actual implementation after I laid the groundwork.

I researched several methods of combing a website and extracting download URLs to pass to a Unix downloader function. I had the option of using Perl with the HTML library, Python with the BeautifulSoup library, or Python with the Scrapy crawler. While all three options were highly capable, I chose the Python with BeautifulSoup as my base for several reasons. Firstly, the group was already using Python for machine learning, so conforming to the same language would prevent confusion and improve productivity. Secondly, the setup procedure for this method was less complicated than it was with the other two methods (Hietaniemi, n.d.), especially on Windows machines, which would facilitate collaboration among the team. Thirdly, we were only operating on one website, so I felt that Scrapy, which also crawls embedded links on the page, was overkill for our project scope (Lookfwd, 2016). Figure 4 below displays part of our crawling code, specifically the functions which extract the contents and parse out the relevant file links.

Predicting Bad Patents

```
# Downloads case information from the USPTO's PTAB API
# First downloads metadata, then attempts to find and download the final
# decision documents for further processing

ptab_url_tmpl = "https://ptabdata.uspto.gov/ptab-api/trials?limit=100&offset={}"
delay_mean = 1.0
delay_stddev = delay_mean * 0.07

# Requests and returns the contents located at the given HTTPS-protected URL
def get_raw_contents_at(url):
    ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
    response = urllib2.urlopen(url, context=ssl_context)
    contents = response.read()
    return contents

# Convenience method for JSON-decoding content at a given HTTPS-protected URL
def get_json_at(url):
    return json.loads(get_raw_contents_at(url))

# Get the total number of cases available from the API
def get_num_cases():
    ptab_url = ptab_url_tmpl.format(0)
    response = get_json_at(ptab_url)
    return response["metadata"]["count"]
```

Figure 4: Snippet of crawler code (Yew et al, 2016)

Besides scraping and downloading, we needed a script that would extract raw text from the downloaded PDF files. From my research, I found only one possible way to do this, which was by using the “pdftotext” command line tool on Unix-based operating systems. Once we had all the PDF files, a short bash script to run this command in a loop on every single file took care of the rest.

After Joong Hwa and William completed the implementation according to my plan, I verified the correctness of their work by querying for the total number of PDF files on the eFOIA website, and comparing that against the output of the scripts that my teammates wrote.

An added benefit of this scripting project was that the scripts could be used by future teams to re-acquire the data in case our downloaded files were lost for any reason. This allowed us to maintain our goal of ensuring easily portable and transferrable work.

4. Developing the Back End of Full-Stack Web Application

Our work would be for nothing if we did not provide our target users with a method of executing our code and viewing our results. I decided to create a web application for this purpose. Our application

Predicting Bad Patents

allows users to input patent text and receive a prediction, view statistics about past predictions, and search for existing predictions by patent number.

I had several established design standards to choose from when developing the back end of our web application, such as model-view-controller, model-view-view-model, and state-action model. I selected the model-view-controller standard after determining that our relational database schema formed an effective model, and that our machine learning algorithms were appropriate to be given controller responsibilities. While the other standards were also valid for similar reasons, I personally had industry experience with this standard, so choosing model-view-controller allowed me to leverage my past experience for more rapid development.

The model-view controller standard abstracts the data into objects, in this case our patents. The object attributes, copied from our database tables, would be properties like the claim type, PTAB decision outcome, and filing date. By structuring the model to conform with our database schema, I could achieve instantaneous CRUD (create, retrieve, update, delete) operations on our patent data in just a few lines of code. This is because the CRUD operations can be implemented in a single file and abstracted into simple functions which can later be called by the controller (Daling, 2009).

The controller functions as the “brains” of the web application, its core functionality being the ability to process requests and deliver responses (Daling, 2009). It also determines the flow between different pages of the web application, redirecting the user to the appropriate page whenever the user clicks on a button or hyperlink with that capability. An example of the controller’s responsibility is when a user requests an invalidation prediction on a new patent. The controller will wait for the user to submit the patent text, create a new object with the patent’s attributes, and insert the object into the database. The controller will then call our machine learning library and perform the appropriate calculations, before returning a numeric prediction to the user. Another example is when a user requests the result of a previous prediction on a patent. The user will supply the patent number to the

Predicting Bad Patents

controller, and the controller will query the database to extract the matching model, and from that model, the recorded prediction, before returning the result to the user. Figure 5 below displays a sample of how these interactions play out in the application's Python back-end code.

```
def predict(request, *args, **kwargs):
    if request.method == 'POST':
        patent = ""
        # try getting patent text from file
        patent += request.POST.get('textfield', None)
        # get outcome type
        outcome = request.POST.get('outcome', None)

        #calculate probability
        probability = case_prediction.predict_probability(patent,outcome)
        color = ""
        if probability < .25:
            color = "green"
        elif probability < .75:
            color = "yellow"
        else:
            color = "red"
        response = "There is a <span style='color:"+color+"'>{0:.1f}%</span>".format(probability*100)
        response += " chance of "+outcome+" . <br/>"

        # get words
        words = case_prediction.get_top_keywords(patent,outcome)
        response += "The words that are most indicative of case "+outcome+" are:<br/> <ul> "
        for word in words:
            response += "<li>"+word+"</li>"
        response += "</ul>"

    return HttpResponse(json.dumps({'response': response}), content_type="application/json")
```

Figure 5: Integrating the machine learning model into a Django application (Yew, 2017)

5. Designing the Front End of a Full-Stack Web Application

The views act as the “eyes” and “ears” of the web application, since they constitute the front end which users actually interact with. The views implement elements like text entry boxes, submission buttons, navigation bars, and others. These elements are used by users to make requests to our controllers and to navigate between different pages of the web application (Daling, 2009). Figure 6 displays examples of typical components that go into the views of model-view-controller applications.

Predicting Bad Patents

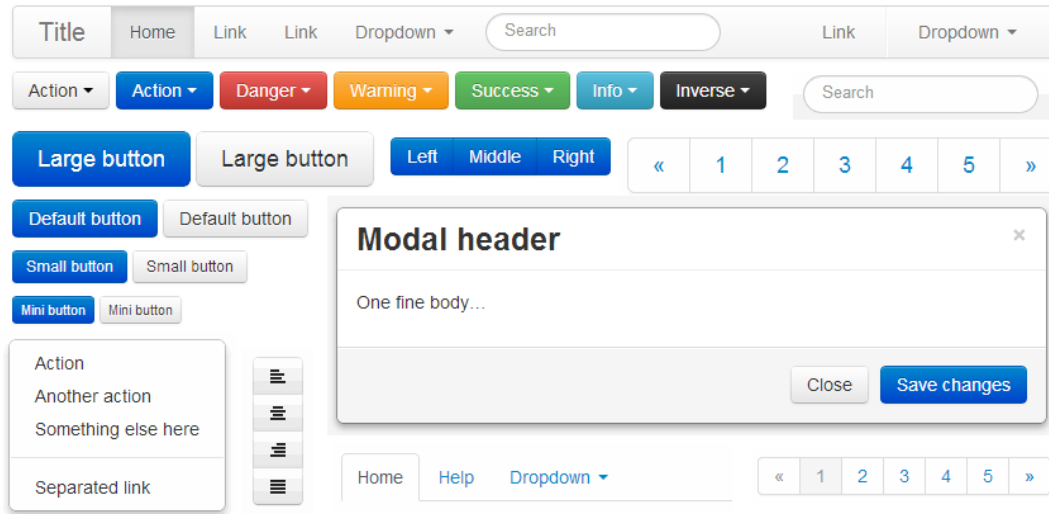


Figure 6: Front-end webpage components (Bootstrap Components, 2014)

While not a central priority for the project, the styling of views is nevertheless important, and care must be taken to ensure that poor styling decisions do not detract from the user experience. Many open-source front-end web tools exist to address this need, such as Bootstrap, jQuery UI, and React.js. These tools provide style-sheet code which can be integrated into the web application, and ensure a uniform, aesthetically pleasing style for components like navigation bars, text boxes, buttons, cursor icons, and others. I chose Bootstrap for its robust suite of features and ease of installation, as well as my own industry experience with the standard. Bootstrap's appearance is visible in Figure 6 above. My teammate Joong Hwa Lee chose specific styling attributes such as colors and fonts.

Besides styling, user experience is also impacted by the decisions over what sort of data to display on which page, and the methods of transitioning between each page. Following industry best practices, I planned out a flowchart for the web application, which is displayed in Figure 4 below. Since the main focus of our project is on predictions, I decided to feature a prediction form on the main index page. Links and a search box in the navigation bar would allow users to access the other two core pages – one for data visualization and one for searching past predictions. The navigation bar's contents remain static no matter which page the user is on, ensuring that the user does not get confused or lost and is able to

Predicting Bad Patents

easily return to pages of interest. My teammate Joong Hwa Lee was responsible for arranging the overall layout of these views. Figure 7 below displays the final result.

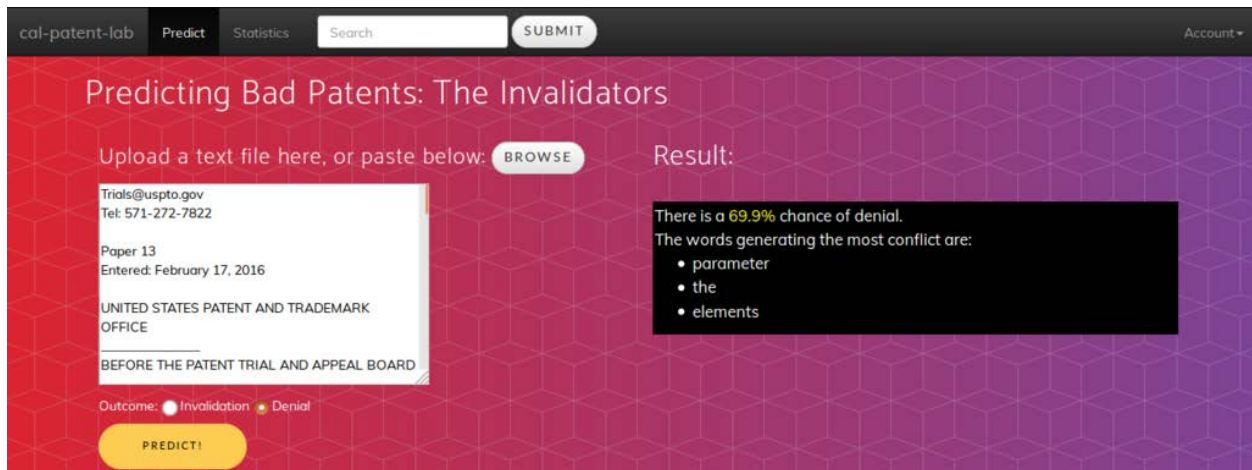


Figure 7: The final appearance of our graphical user interface (Yew, 2017)

6. Optimizing the Front and Back End Relationship

With the front and back end infrastructure established, the final piece of the puzzle was the method of communicating between the two sides. Specifically, how would my web application direct its responses to user requests? The simplest and most straightforward method is to have a separate Django view specifically to display the response, and the controller would redirect the user to that view upon submission of the request. While serviceable, this approach is not ideal because it constrains the web app's ability to handle erroneous input, and there is a risk of the response page crashing as a result, which would confuse the user. Instead, I utilized AJAX (Asynchronous JavaScript and XML) functions to asynchronously send requests and responses without having to navigate to a new page (Marsh, 2014).

The diagram in Figure 8 below demonstrates how a typical AJAX communication plays out between the front and back ends. In this case, the back end is the Django controller and the front end is the Django view, specifically the prediction form on the index page. The user would paste the patent text into the form, and click the "Predict" button. A small viewport on the right side of the page would display the response from the controller. If the input contained any errors, a warning would be

displayed in the viewport, so that the user would be aware and thus able to resubmit the form with the correct input. Keeping the user on the same page also improves the application's overall performance as no time is wasted on redirection to a new page.

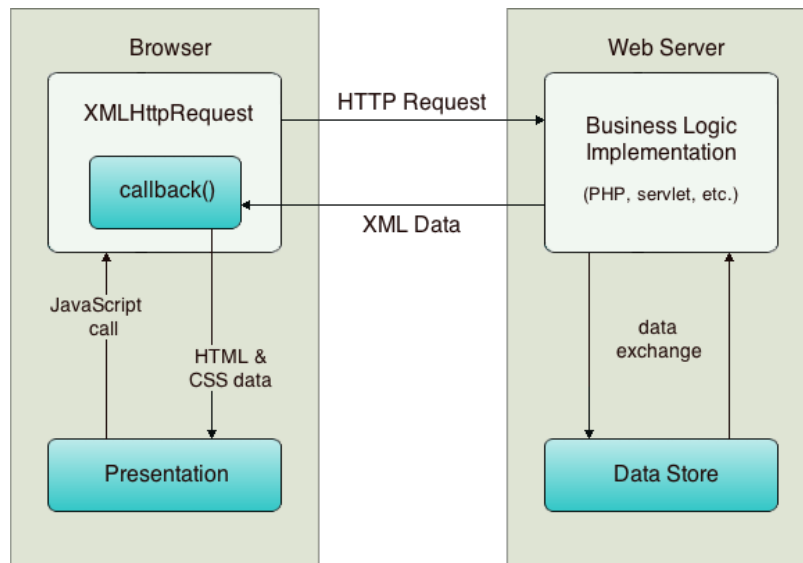


Figure 8: How AJAX communication works (Netbeans Community, 2016)

AJAX also plays a pivotal role for the search box used in my navigation bar. When the user begins typing the patent number, AJAX functions immediately call the controller and pass it the partial patent number, and repeat this request for every new character the user inputs. A list of potential matches is returned, and the list rapidly shrinks as the user continues to input more characters. If the user sees the matching patent inside the drop-down list, then the user can immediately click on the match and be redirected to the result without having to type the entire patent number. This is a standard design feature of countless modern web applications, and contributes to a positive user experience.

7. Conclusion

My technical contributions all played a key role in helping us achieve our goals, allowing us to reach all our milestones ahead of schedule, as shown in Figure 9. The infrastructure I established was

instrumental to every team member's coding responsibilities. The relational database was integrated into almost every parser and machine learning function we wrote, and the content we stored in that database will be a useful reference for future teams. My web application ultimately allows our target audience to interact with and appreciate our work, and acquire statistics for their own use.

Chapter 2: Engineering Leadership

1. Introduction and Overview of Technology

As technological innovation has accelerated over the last two decades, the number of patents has also grown substantially. 300,000 patents were filed with the US Patent and Trademark Office (USPTO) in 2014, which shattered the previous record for the number of patents in a year (U.S. Patent Statistics Chart, 2016). This growth in applications, however, also led to more resources being spent on patent litigation; for example, more than \$20 billion was spent on patent litigation in the last two years. We aim to build a machine learning tool for patent applicants and examiners to check the likelihood that their patents will be invalidated before even filing them. By providing this information before patents are approved, we hope to limit the cost that bad patents impose on the legal system and the economy—estimated to be up to \$26 billion annually (Beard, p. 268). In this paper, we will examine the industry context and business considerations associated with building this machine learning tool. First, we will discuss the current patent landscape and how it informs the marketing strategy for potential customers. Second, we will analyze the different competitors in the legal services space and define how our tool differs from existing offerings. Third, we will discuss the current state and trends in the machine learning field today and how they can be applied to our tool. Fourth and finally, we will close with an ethics section that will examine the ethical issues that we must consider in building the product.

Predicting Bad Patents

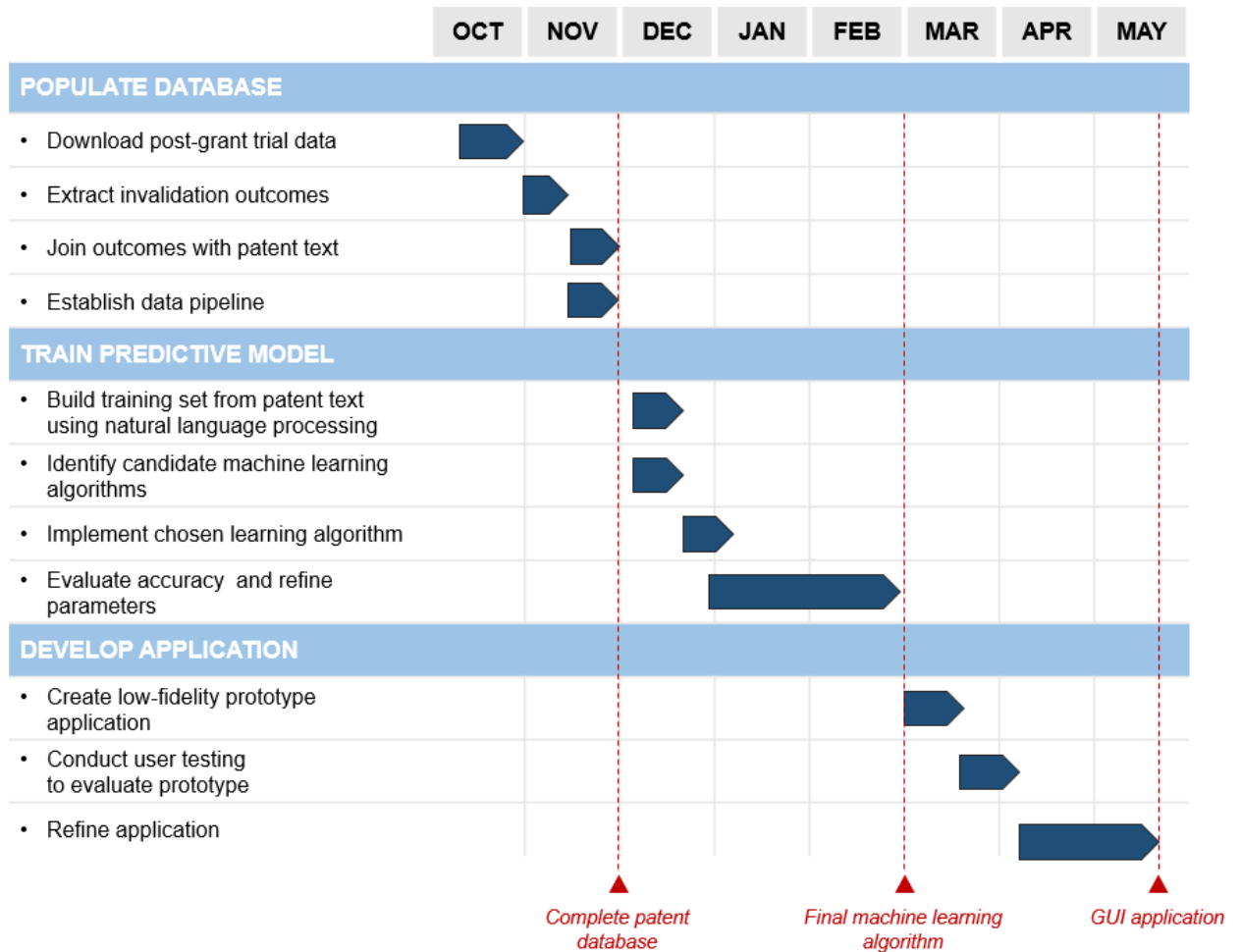


Figure 9: Milestone timeline (Winer et al, 2017)

2. Marketing Strategy

It is becoming increasingly challenging for research-oriented firms and their attorneys to navigate the intellectual property landscape in the United States. In addition to the increase in the sheer number of patents, recent changes in US law have made it significantly easier for members of the public to challenge existing patents. In 2012, the US federal government enacted the Leahy-Smith America Invents Act (AIA). This legislation substantially expanded the role of the US Patent and Trademark Office in post-grant patent reexamination (Love, 2014, Background). The AIA opened the gates of post-grant patent opposition to members of the public by providing a much less costly and more streamlined

Predicting Bad Patents

avenue for post-grant opposition through the Patent Office's Patent Trial and Appeals Board (PTAB). Any member of the public could challenge an existing patent for only a few thousand dollars—relatively inexpensive compared to litigation (Marco, 2016).

Accordingly, the patent application process is under two types of strain: it is resource constrained—since there are more and more patents being filed every year—and it is coming under more scrutiny due to the America Invents Act. There are two main sets of stakeholders that have an interest in improving the current application process: (1) the USPTO and (2) patents filers and their lawyers.

Firstly, because “IP-intensive industries accounted for about [...] 34.8 percent of U.S. gross domestic product [...] in 2010” (Economics and Statistics Administration and United States Patent and Trademark Office, March 2012, p. vii), reducing the time it takes to effectively examine a patent—perhaps through assistance from a computerized algorithm—is a critical priority for the USPTO (appendix A). Indeed, helping patent examiners reduce the time they spend on each patent (while still maintaining the quality of examinations) would mean reducing the cost and time associated with filing patents, proving economically accretive and reflecting well on the US Patent and Trademark Office. In fact, the USPTO has expressed interest in a predictive service in the past and has conducted its own research into predicting invalidation (US Patent and Trademark Office, 2015, p. 38).

Secondly, when applying for patents, patent filers and their attorneys have a strong interest in preempting potential litigation through effective framing and wording of their patents. Patent litigation is becoming more and more common as evidenced by IBISWorld: “Demand for litigation services has boomed over the past five years” (Carter, 2015, p. 4). Therefore, a tool that could help patent filers prevent litigation would be valuable during the application process. One industry that may be especially interested in this sort of tool is Business Analytics and Enterprise Software. In the past several years, the

Predicting Bad Patents

costs associated with protecting “a portfolio of patents” have disproportionately increased in this industry (Blau, 2016, p. 22).

3. Competition

Patent validity is a major concern for the \$40.5 billion intellectual property licensing industry, whose players often must decide whether to license a patent or challenge its validity (Rivera, 2016, p. 7). These decisions are currently made through manual analysis conducted by highly-paid lawyers (Morea, 2016, p. 7). Because of the cost of these searches, data analytics firms such as Juristat, Lex Machina, and Innography have created services to help lawyers perform analyses more effectively.

One common service is semantic search for prior art and similar patents, where queries take the form of natural language instead of mere keywords. Other services include statistics about law firms, judges, and the patent-granting process. These services build their own high-quality databases by crawling court records and other public data sources, correcting or removing incomplete records, and adding custom attributes to enable such search patterns and reports. Their prevalence reflects the trend towards data analysis as a service, since law firms are not in the data-analysis business (Diment, 2016).

The above services lie outside the scope of predicting invalidation from patent text and metadata but become relevant when discussing commercialization because high-quality data improves model accuracy and enables techniques like concept analysis that are difficult or impossible with raw unlabeled datasets. As such, partnering with existing firms that provide clean datasets or otherwise cross-licensing our technologies may be advantageous.

While these services help lawyers make manual decisions with historical statistics, we have found no service that attempts to predict invalidation for individual patents. Juristat is the only major service we found that performs predictions on user-provided patent applications. Specifically, Juristat predicts how the patent office will classify a given application and highlights key words contributing to

Predicting Bad Patents

that classification, with the goal of helping inventors avoid technology centers in the patent office with high rejection rates (Juristat, n.d.).

Our project, if successful, can become a Juristat-like service for predicting post-grant invalidation. While we cannot speculate on existing firms' development efforts, the lack of similar services on the market suggests a business opportunity. Whereas existing services target law firms and in-house IP lawyers, our project aims to help the USPTO evaluate post-grant petitions, which are brought forth by parties attempting to invalidate a patent.

4. Machine Learning Technology Trends and Strategy

This work has been enabled by many recent advances in the application of machine learning to large data problems. Even though machine learning has been around for several decades, it took off within the past decade as a popular way of handling computer vision, speech recognition, robotic control, and other applications. By mining large datasets using machine learning, one can "improve services and productivity" (Jordan & Mitchell, 2015 p. 255-256), for example by using historical traffic data in the design of congestion control systems, or by using historical medical data to predict the future of one's health (Jordan & Mitchell, 2015 p. 255-256). For this project, we had access to a large dataset of historical patent filings since 1976, for which recently developed machine learning techniques proved especially useful.

Machine learning algorithms generally fall into one of two categories: supervised and unsupervised (Jordan & Mitchell, 2015 p. 256). Supervised learning algorithms need to be run on training data sets where the correct output is already known. Once the algorithm is able to generate the correct output, it can then be used for regression or clustering. In contrast, unsupervised learning algorithms use data sets without any advance knowledge of the output, and perform clustering to try to find relationships between variables which a human eye might not notice. Recent trends indicate that

Predicting Bad Patents

supervised learning algorithms are far more widely used (Jordan & Mitchell, 2015 p. 257-260). Our historical data set indicated whether or not patents were invalidated/denied during past disputes, which made a supervised learning algorithm the appropriate choice.

5. Ethical Challenges

As with all engineering projects, we anticipated the possibility of running into potential ethical conflicts. We used the Code of Ethics, written by the NSPE (National Society of Professional Engineers) (NSPE, 2017), as a guideline for our planning. We identified two components of the Code of Ethics, which our project could potentially violate if left unchecked.

The first is Rule of Practice 2: “Engineers shall perform services only in the areas of their competence” (NSPE, 2017). One of our potential target customers is the United States Patent and Trademark Office, who would ideally use our project to aid with their patent approval decisions. If our project were seen to be an automated replacement, rather than a complement, for trained patent examiners or attorneys, that may be considered an attempt to perform services outside of our “areas of competence.” While we cannot dictate how our customers ultimately utilize our product, we can mitigate the issue through thorough written recommendations in our documentation to hopefully encourage responsible usage.

The second ethical consideration is Professional Obligation 3: “Engineers shall avoid all conduct or practice that deceives the public”. While we fully intend our project to be used in service to the public, we recognize the possibility of bias in our supervised machine learning algorithm (Reese, 2016), with the resulting output capable of unfairly swinging the outcome of a patent decision. Unlike the prior ethical issue, we have more control in this situation, since we do not have a viable product without a sufficiently trained algorithm. By verifying our datasets to ensure equal representation and objective input, we can avoid inserting biases and thus maintain ethical integrity.

6. Conclusion

Collectively, recent economic and regulatory trends have made now an exciting but uncertain time for inventors, attorneys, and the US Patent and Trademark Office. Thoughtful applications of machine learning and statistics can make sense of these recent changes and assist stakeholders in truly understanding what drives patent invalidation. As we pursue this technology, our understanding of the industry landscape of potential customers/competitors, leading trends in machine learning research, and the ethical considerations associated with our technology will drive our research. Ultimately, we hope that our technology contributes to the continued development of a patent ecosystem that enables inventors to do what they do best: developing novel and socially valuable inventions.

Chapter 3: Overall Conclusions

Our combined efforts resulted in a novel and robust software solution which we believe satisfies the needs of our target audience. In the long run, we hope our work will help steer the United States patent system back towards its original purpose of fostering innovation, while also contributing to increased public interest in machine learning as a tool to solve broad problems (Yew, 2016).

References

- Yew, Tzuo Shuin, Ho, William, Lee, Joong Hwa (2016, August) PTAB API Downloader. Retrieved April 25, 2017 from https://github.com/davidjwiner/cal-patent-lab/blob/master/lib/ptab_api_downloader.py
- Yew, Tzuo Shuin (2017, April) Django Back End Controller. Retrieved April 25, 2017 from <https://github.com/davidjwiner/cal-patent-lab/blob/master/patentGUI/invalidators/views.py>
- Netbeans Community. (2016, December 15). Introduction to Ajax for Java Web Applications [Ajax Tutorial]. Retrieved March 12, 2017, from <https://netbeans.org/kb/docs/web/ajax-quickstart.html>
- Bootstrap Components [Typical front end components]. (2014). Retrieved March 12, 2017, from <http://www.boss-development.biz/taxonomy/term/4>

Predicting Bad Patents

Tezer, O. (2014, 2). SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean. Retrieved March 11, 2017, from <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>

Boston Computing Network. (2004, April 17). UNIX vs. Windows Hosting. Retrieved March 11, 2017, from <https://www.bostoncomputing.net/webhosting/unix-vs-windows/>

Patent Technology Monitoring Team. (2016, October 16). U.S. Patent Statistics Chart. Retrieved from USPTO: https://www.uspto.gov/web/offices/ac/ido/oeip/taf/us_stat.htm

Winer, David J. (2017, 4). "Predicting Bad Patents: Final Capstone Report"

Yew, Tzuo Shuin (2016, 11). "Predicting Bad Patents: Project Status"

Yew, Tzuo Shuin (2016, 10). "Predicting Bad Patents: Technological Trends"

Srage, Dany (2016, 10). "Capstone Project Context: Customers"

Winer, David J. (2016, 10). "Predicting Bad Patents: Analyzing the Economic and Regulatory Landscape"

Ho, William (2016, 10). "Project Context: The Competitive Landscape"

Lee, Joong Hwa (2016, 10). "Project Context: User Interface Design"

Winer, David J. (2016, 11). "Predicting Bad Patents: Machine Learning Status Update"

Lee, Joong Hwa (2016, 11). "Predicting Bad Patents: Data Wrangling"

Srage, Dany (2016, 11). "Predicting Bad Patents: Project Status" Tezer, O. (2014, February 21). SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean. Retrieved March 11, 2017, from <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>

Boston Computing Network. (2004, April 17). UNIX vs. Windows Hosting. Retrieved March 11, 2017, from <https://www.bostoncomputing.net/webhosting/unix-vs-windows/>

Dalling, T. (2009, May 31). Model View Controller Explained. Retrieved March 11, 2017, from <http://www.tomdalling.com/blog/software-design/model-view-controller-explained/>

LeDoux, C. H., & Parker, D. S., Jr. (1982, September). Reflections on Boyce-Codd Normal Form [Scholarly project]. In Very Large Databases Endowment Inc. Retrieved March 11, 2017, from <http://www.vldb.org/conf/1982/P131.PDF>

Hietaniemi, J. (n.d.). Comprehensive Perl Archive Network. Retrieved March 11, 2017, from <http://www.cpan.org/modules/INSTALL.html>

Predicting Bad Patents

- Lookfwd. (2016, January 29). Scrapy VS. urllib2, requests, BeautifulSoup and lxml. Retrieved March 11, 2017, from <http://www.scrapinginsider.com/2016/01/scrapy-urllib2-requests-beautifulsoup-lxml.html>
- Marsh, J. (2014, April 7). Ajax Submit: Submitting Forms Asynchronously. Retrieved March 12, 2017, from <https://blog.udemy.com/ajax-submit/>
- Beard, T.R., Ford, G.S., Koutsky, T.M., & Spiwak, L.J. (2010). Quantifying the cost of substandard patents: some preliminary evidence. *Yale Journal of Law and Technology*, 12(1): 240-268.
- Blau, G. (2016). IBISWorld Industry Report 51121c Business Analytics & Enterprise Software Publishing in the US. IBISWorld. Retrieved October 16, 2016
- Carter, B. (2015). IBISWorld Industry Report OD4809 Trademark & Patent Lawyers & Attorneys in the US. IBISWorld. Retrieved October 15, 2016
- Diment, Dmitry (2016, March). IBISWorld Industry Report 51821. Data Processing & Hosting Services in the US. Retrieved October 11, 2016 from IBISWorld database.
- Economics and Statistics Administration and United States Patent and Trademark Office. (March 2012). Intellectual Property And The U.S. Economy: Industries in Focus. U.S. Department of Commerce. Retrieved October 15, 2016
- Jordan, M. I., & Mitchell, T. M. (2015, 07). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255-260. doi:10.1126/science.aaa8415
- Juristat - Patent Analytics. (n.d.). Retrieved October 13, 2016, from <https://juristat.com/#etro-1>
- Love, B.J., & Ambwani, S. (2014). Inter partes review: an early look at the numbers. *University of Chicago Law Review*, 81(93). Retrieved October 14, 2016 from: <https://lawreview.uchicago.edu/page/inter-partes-review-early-look-numbers>
- Marco, A. (2016, October 5). Phone interview with USPTO Chief Economist.
- Morea, S. (2016). IBISWorld Industry Report 54111 Law Firms in the US. IBISWorld. Retrieved October 16, 2016
- NSPE. (n.d.). Code of Ethics. Retrieved February 03, 2017, from <https://www.nspe.org/resources/ethics/code-ethics>
- Patent Trial and Appeal Board Statistics [PDF Document]. Retrieved October 15, 2016 from United States Patent and Trademark Office web site: <https://www.uspto.gov/sites/default/files/documents/2016-08-31%20PTAB.pdf>
- Reese, H. (2016, November 18). Bias in machine learning, and how to stop it. Retrieved February 04, 2017, from <http://www.techrepublic.com/article/bias-in-machine-learning-and-how-to-stop-it/>
- U.S. Patent Statistics Chart [PDF Document]. Retrieved from United States Patent and Trademark Office web site: https://www.uspto.gov/web/offices/ac/ido/oeip/taf/us_stat.htm
- US Patent and Trademark Office (2015, January). Patent Litigation and USPTO Trials: Implications for Patent Examination Quality. Retrieved October 9, 2016 from

Predicting Bad Patents

<https://www.uspto.gov/sites/default/files/documents/Patent%20litigation%20and%20USPTO%20trials%2020150130.pdf>

Walker, J. & Copeland, R. (2015, April 7). New hedge fund strategy: dispute the patent, short the stock. The Wall Street Journal. Retrieved October 15, 2016 from: <http://www.wsj.com/articles/hedge-fund-manager-kyle-bass-challenges-jazz-pharmaceuticals-patent-1428417408>