

# Entrepreneurial Patent Data Analysis

*Alper Vural  
Nikhil Narayen  
Martin Gouy*



Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2016-50

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-50.html>

May 11, 2016

Copyright © 2016, 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.

# **Final Capstone Report**

May 2016

Entrepreneurial Patent Data Analysis

By: Alper Vural

## Table of Contents

Individual Technical Contributions .....	3
Abstract .....	3
Overall Project Goals and Technical Definitions .....	3
Work Breakdown and Further Technical Definitions .....	4
Constraining Resources .....	5
Purpose of This Paper .....	6
Clustering Goals .....	6
Necessity of the Sparse Matrix .....	6
Limitations of Hierarchical Clustering and K Means .....	7
Graph-Based Clustering .....	9
Effects of High Dimensionality on Clustering .....	10
Singular Value Decomposition and Lloyd’s Algorithm .....	10
Final Implementation and its Accuracy .....	12
Value of my Work to the Public .....	13
Future Work .....	14
Value of my Work to the Team, More Future Work, and Conclusion .....	14
Team Paper (everything besides intro by Alper Vural, Martin Gouy, Nikhil Narayen) .....	17
Abstract .....	17
Introduction .....	17
Industry Analysis .....	19
Go to Market Strategy .....	21
Conclusion .....	22
Appendix .....	23
Modified K Means Instructions .....	23
Modified K Means Code .....	23
Patent Feature Clustering Instructions .....	26
Patent Feature Clustering Code .....	26
References .....	28

## Individual Technical Contributions

### Abstract

The problem our capstone project seeks to solve is finding groupings of similar patents using two data sets with information on 5 million previously filed patents. The first data set has records of what previous patents each patent cited, and the second one contains text from each patent. Our solution converts these two data sets into a “samples” and “features” format, which is then usable by a machine learning technique known as clustering. Clustering takes data items in the aforementioned format and groups similar items together. Clustering is a very common practice, and a literature review revealed a plethora of approaches, including spectral clustering, K Means, and hierarchical clustering. We measured the accuracy of each approach by checking how consistent the groupings were with a third data set, a text file containing pairs of patents that blocked each other from being filed. The best approach we found ended up being a modified version of the standard K Means. Due to issues with the citation data set, as well as time and server memory constraints we achieved some success but did not reach the desired accuracy.

### Overall Project Goals and Technical Definitions

Our capstone project is a tool that primarily receives as input a legal patent and outputs previously filed legal patents that are similar enough to block the original patent. Patent blocking at one point involved a new patent being restricted by the fact that it is an improvement upon a previous patent (Adams, 2008). Adams discusses how patent blocking has been an accepted concept in the legal system for a long time, but its particular definition has changed at certain points (2008, pp.7-8). Thus, one should note that determining whether two patents conclusively block, or invalidate, each other is very difficult for someone without a legal background.

Practically speaking, an inventor or lawyer could use our tool to more easily verify that a patent to be filed is sufficiently different from those that already exist, and there are smaller achievements that we will work towards. One of these, an alternate use of our tool, is finding patents that are similar but not similar enough to block the input patent. The European Patent Office describes these similar patents, known as prior art, as “**any** evidence that your invention is already known” (2015, para. 1). When a patent is filed, it must cite similar already existing patents as prior art. The other achievement is the creation of a patent classification system by dividing the patents into similar groupings.

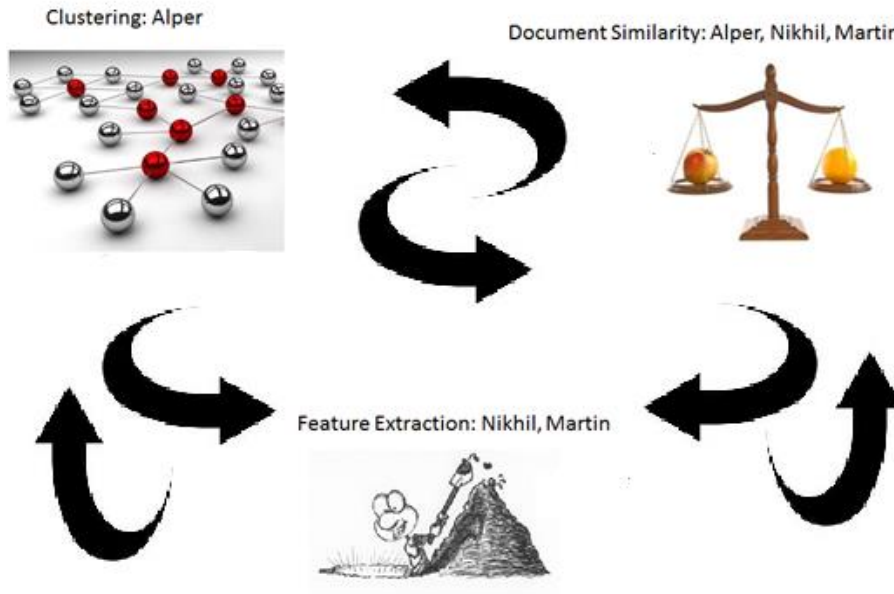
### Work Breakdown and Further Technical Definitions

We have access to three key data sources in accomplishing our project goal. The first is the ‘claims’ sections of previously filed legal patents stored in a MySQL database that can be programmatically accessed. The claims section of a patent describes what the patent protects. The second data set is essentially a text file where each line corresponds to one patent and contains information about what prior art citations of previously filed patents that patent has. The third data set is a text file where each line contains a pair of patents in which one patent blocked the other. This means that the other patent was too similar to the previous patent to be accepted. Our work with this data can be divided into three tasks: feature extraction, document similarity, and clustering. Feature extraction involves finding a way to concisely represent the text of each patent, document similarity is a way to compute how similar two of these representations are, and clustering involves separating the patents into groups. Feature extraction applies specifically to the patent claims data set, whereas document similarity and clustering apply to both data sets.

My partners, Nikhil and Martin, are working on feature extraction, I am working on clustering, and we all play a role in document similarity. Each of these tasks is dependent on the

others. For example, the features Nikhil and Martin extract affect what similarity metric is best.

Figure 1 emphasizes how the three tasks are interrelated in this way.



*Figure 1:Relation of tasks*

Additionally, Martin conducts business research, including looking up information about potential competitors we would have if we turned our technology into a startup.

### Constraining Resources

We have two main limitations regarding our tool. The first is memory, as we have a server with 64GB of memory, which is large, but this is still a finite amount. The second is time, as we want the tool to be able to accomplish our goals within a reasonable amount of time. For example, it would be unreasonable if our tool took a week to output similar patents to an input patent, and it would be unreasonable if our tool took half a year to find groupings for all the existing patents. Computer science algorithms have a “Big O” notation, which describes how the runtime or space consumption of the algorithm increases relative to its input size. For example,  $O(n)$  for time and  $O(n^2)$  for space means that the time taken has a linear relation with the input

size and the space used has a quadratic relation with the input size. For the citation data, the input size was 5 million, as we had the citation data for 5 million patents.

## Purpose of This Paper

The purpose of this paper is to justify my use of a relatively simple algorithm for the tasks of document similarity and clustering, as well as my stance on the lack of usability of some of our data. I will first do this by discussing my work in narrowing down our other options for clustering the patents, my final implementation, and my final results. From there, I will discuss why my work is helpful to my team, including how this paper is related to that of Martin and Nikhil, and why my work is helpful to the public.

## Clustering Goals

There are two goals I hoped to achieve with my clustering. The first was to create groupings of patents that could each be considered a category, and this could be a potential replacement for the current US Patent Office patent classification system. Because my advisor hinted that patents can be considered to be within one of 6 broad groupings, I aimed for a small number of clusters here, less than 10. The second was to create groupings for a multi-level hierarchy so that given an input patent, we start at the top of the hierarchy and go down to the next level in the direction of the group that the patent was most similar to. Once we reached the bottom, we would have a small group of patents that were very similar to the input patent. Since the hierarchy had multiple levels, I also aimed for a small number of clusters per split for this goal, again less than 10. In both cases, I wanted to patents to be evenly distributed among the clusters if possible.

## Necessity of the Sparse Matrix



With the help of my advisor, I found a compact representation for the prior art citation data in my code. The programming language I chose to work in was Python, with emphasis on a library called Numpy. Numpy allowed me to use matrices to store data, and so a naïve approach was to store the citation data in a two-dimensional matrix, where each entry was either a 1 or a 0, and a 1 in row  $i$ , column  $j$  means that patent  $i$  cited patent  $j$ . Unfortunately, this required too much space when using all the data, and so a better approach was to use a sparse matrix from a library called Sci-Kit Learn. Note that I used Sci-Kit Learn for many of coding libraries in my project (Sci-Kit Learn). The sparse matrix represented a two-dimensional matrix as a series of coordinates of the points in the matrix that had a value of 1. Since each patent only had a handful of citations, the majority of the values in the two-dimensional matrix were 0, and so this approach saved an immense amount of space. Thus, the sparse-matrix representation is superior to the default two-dimensional one. Note that only about 114 million of the nearly 3 trillion entries in the two-dimensional representation were 1's and not 0's

### Limitations of Hierarchical Clustering and KMeans

Early on, my team decided it best for me to direct my focus toward clustering our existing database of patents using our prior art citation data, and I discovered the limitations of two algorithms, hierarchical clustering and K means. Hierarchical clustering comes in a variety of forms, but I chose single-linkage clustering, which considers each patent as a separate group to begin with and at each step combines the groups of the two most similar patents in different groups. Figure 2 demonstrates how each item begins in its own group and similar groups are steadily combined together.

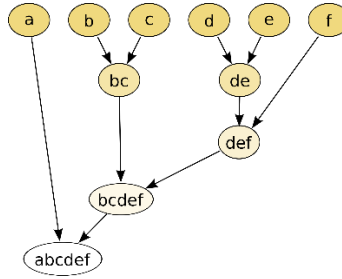


Figure 2: Example of Hierarchical Clustering

I used the straightforward implementation described by Manning, Raghavan, and Schütze (2009, pp. 377). Unfortunately, the algorithmic runtime of this implementation proved to be too costly. Thus, hierarchical clustering is not feasible for our task.

Another method I attempted was K means, which represented the data as coordinate points and sought to find the best centroids for a pre-specified number of groups, called clusters. The centroid of a group is the average of its coordinate points. The algorithm did this by first randomly selecting centroids, assigning the points closest to a centroid to that centroid, adjusting each centroid to be the actual center of the points associated with it, and repeating this process over a certain number of iterations. Figure 3 is an example of how some two-dimensional data points could be labeled by a K means algorithm (it is difficult to visualize higher-dimensional data such as our patent data).

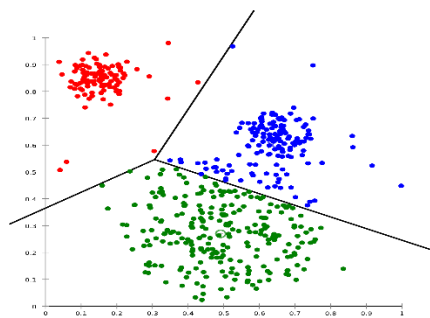


Figure 3: Example of K means

Unfortunately, even when specifying just two clusters, K means would put over 99% of the patents into the same group for various subsets of the patents, which was essentially useless. I hypothesized that the high-dimensionality of this data set was a likely reason for this. In machine learning, data points, known as samples, have features that describe them. Here, the data points are the patents and the features are indications of whether they cite another patent. Data with a large number of features is in a space of high dimension, which means that they will be inherently seen as being one large group relative to the coordinate space (Efros 2015). Regardless, rather than immediately dealing with what I thought was an issue of dimensionality, I next looked into an alternate representation for the patent data, a graph, as a result of a data science lecture (Gonzalez 2015).

### Graph-Based Clustering

My next strategy was to use graph-based clustering. In computer science, a graph is a set of vertices and edges that connect some of the vertices to each other. In this case, each patent was a vertex, and if one patent cited another then the two would share an edge. Each patent was treated as a vertex, and each citation in the citation matrix was treated as an undirected edge between the two vertices. The connected components of a graph refer to subsets where for each subset, one can travel between any two vertices in that subset using the edges of that subset. A basic approach to graph clustering treats each connected component of the graph as its own group. Unfortunately, a quick analysis on a subset of the citation records yielded 85% of the vertices in the same connected component, which would be uneven groupings. I was looking to divide them evenly, which meant that at the very most the largest group should have about only half the patents. Another approach is Spectral Clustering. Ng, Jordan, and Weiss (2002) mention Spectral Clustering, which uses Kmeans in a way that makes up for its shortcomings for certain

datasets (pp. 7). Unfortunately, the Sci-Kit learn Spectral Clustering library took an unreasonably long time to run even for smaller subsets of the patents, making it very impractical. Though more graph-based clustering approaches for the citation data remain, the most available ones did not meet our needs.

## Effects of High Dimensionality on Clustering

I had suspected throughout my project that the high dimensionality of my data was the cause of it being grouped into uneven clusters. Unfortunately, I realized later that this was not actually the case through a discussion with a UC Berkeley professor (J. Canny, personal communication May 9, 2016). In reality, high dimensional data appears to have more uniform distances, but this could not have been the sole cause of my uneven clustering issue (Steinbach, Ertöz, Kumar, pp. 12). Seeing as I was not aware of this at the time of the bulk of my work, I will continue on with what I did try to fix the problem.

## Singular Value Decomposition and Lloyd's Algorithm

Singular value decomposition (SVD), when used in conjunction with Lloyd's algorithm and K means, dramatically increased my ability to evenly cluster the patents, but this approach still had some drawbacks. Practically speaking, SVD allowed me to approximately represent my data with a smaller number of features. In my case, this would be especially helpful in reducing the dimensionality of my data, thereby circumventing the previously mentioned issue of high-dimensionality. Lloyd's was a process of shifting the grouping around so that some data points would be moved to a group that was not quite the closest one in order to emphasize even groupings. Before Lloyd's, using a subset of 50k of the patents and an SVD reduction to 100 features, whether the number of clusters was 2 or 100, the largest cluster had essentially all of the patents.

Using the full citation data and Lloyd’s algorithm yielded the following results. Note that a major issue I faced was that a significant chunk of the patents had the same SVD compressed values, which made Lloyd’s only partially effective. As part of Lloyd’s I calculated the distance of each point from one of the centroids, and about a third of the points had the same distance, which I equated to having the same value, as the alternative is very unlikely. Table 1 shows more gives more details on what percentage of the points had the same distance.

Patents	SVD Features	% having median distance	% in largest cluster
~5m (all)	10	37.41%	86.30%
~5m (all)	100	37.15%	78.02%

*Table 1: Duplicate Distances for SVD, Lloyd's*

Although SVD reduces the number of features, some information is lost in the process (J. Demmel, personal communication, February 2, 2016). In fact, as indicated in the above table, over a third of the samples had the same feature values in the cases of 10 and 100 features. An easy solution would have been to increase the number of features, but unfortunately using SVD with a higher number of features caused memory issues on our server. In order to confirm that information loss was an issue, I used the “Variance Captured” feature of Sci-Kit Learn’s TruncatedSVD function for various patent subset sizes, as shown in Table 2.

Patents	SVD Feats	Time (no parallelism)	Variance Captured
50k	10	5s	1.60%
500k	10	37s	2.00%
5m	10	282s	1.40%
50k	100	37s	6.70%
500k	100	196s	6.20%

5m	100	921s	4.70%
50k	1k	975s	20.40%
500k	1k	memory ran out	memory ran out
5m	1k	memory ran out	memory ran out

Table 2: Information Kept by SVD

Notice first that runtime is not at all an issue, as for all the patents and 100 features, the time is only about 15 minutes, but unfortunately under 5% of the information is kept. Thus, SVD and Lloyd’s moved me closer to my goal of grouping the citation data but still ultimately fell short.

### Final Implementation and its Accuracy

In order to make the best of the results I had, I tried an alternative method to even the groups, which proved only somewhat effective. Using an idea inspired by Lloyd’s, I ran K means on the data to divide it into 2 clusters with SVD at 100 features as before, retrieving the centroid of the first and second groups. From there, I found the distance from every point (each patent is a point when its citation features are considered as coordinates) to the centroid. Since about a third of the points had the same values, their distances were the same. Let us call this distance  $d$ . Since these points had the median distances, about a third of the patents were closer and about a third were farther. Thus, I placed all the patents with distance less than  $d$  into one cluster, with distance exactly  $d$  into another cluster, and with distance more than  $d$  into a third cluster. I was thereby able to achieve some form of even clustering using this inventive approach.

In general, verifying the accuracy of any patent classification system is difficult, but some patent invalidation data my team had proved reasonably sufficient. This invalidation data consisted of pairs of patents that had blocked each other in the past (a patent was represented by its patent id). Two patents that block each other should be relatively similar and thus should be in

the same grouping. Thus, for each pair that we had citation data for both its patents, a “hit” would be two patents being in the same grouping, and a “miss” would be the two patents being in different groupings. I measured accuracy by dividing the hits by the sum of the hits and the misses. For my three groupings, I scored about 56% accuracy, which is noticeably better than the about 33-40% baseline of patents randomly being placed into groups but is still not very accurate. Thus, I achieved some success, but significant potential for improvement remains. At this point, it was deemed a most beneficial use of my time to assist Nikhil and Martin with their portions.

### Value of my Work to the Public

The current system of patent classification is imperfect, but advances are being made, and my work is relevant to these advances. An overview page from the US Patent Office site mentions how there are more than 150,000 subclasses than are repeatable in 450 classes, which demonstrates that the system is complex and outdated (USPC Classifications section, 2012). Furthermore, some of these categories can enter into a state where there exists no more incentive for inventions in that area to be created, which further complicates matters (L. Fleming, personal communication). Google is currently attempting to create a better way of classifying patents, and they are using citation data in some form to do this (L. Fleming, personal communication, March 2016). My work would encourage them to look more closely into if the citation data is actually benefitting them and potentially even encourage them to look at other data sources to achieve their goals.

Working towards a better patent classification system is significant because it could help an inventor with finding prior art when he or she has an invention that they need to file a patent for. Inventors can overcome their lack of legal background by filing through a law firm, but this

costs anywhere from “\$1000 to \$3000,” so any kind of cheaper solution would be appreciated (Quinn, 2015, para. 11). One of these cheaper solutions is for them to find the prior art themselves through a patent search engine. Kahn mentions Google, Microsoft, and Yahoo as the three major corporate players in the search engine field (2014). Thus far, only Google has a patent search engine, and though that is a good place to start, it takes a large amount of time to type in individual key terms. The individual also needs to be able to figure out what these key terms are. Even then, some patents purposely use synonyms of words to make their patent harder to detect. Better patent groupings would improve the quality of a patent search engine, which in turn would benefit inventors.

Although I did not succeed as expected in clustering the patents, I gained information about the limitations of using the citation data, which will be helpful for people or groups in the future who are looking to cluster the space of existing patents, which in turn is helpful to inventors.

## Future Work

There are other clustering techniques that future capstone teams should try. One of them is a graph-based technique called a normalized cut, which is related spectral clustering and could be used to divide the data in half and provide a quality measure. Next, a tool known as BIDMach can run the same algorithms that Sci-kit Learn does, but with more time and space efficiency. This would allow them, for example, to run singular value decomposition on a greater range of values. Furthermore, they could use tools that have parameters for indicating desires for even clusters in algorithms such as K Means, which would be very helpful.

## Value of my Work to the Team, More Future Work, and Conclusion



My two main contributions to the team were the identification of one of the data sets as being less useful, a better understanding of the limitations of clustering, and the opportunity for work by future capstone teams in this area. Nikhil and Martin's main work involved extracting features from the text of the patent claims, and they would then be used for clustering so that a new, incoming patent could be placed into the cluster with the closest centroid, thereby identifying similar patents to it. My features could be combined with theirs in the following way: I would add 3 features, and the value of the  $n$ th features would be 1 if that patent got placed in cluster  $n$  by the citation data, and otherwise it would be 0.

With better features, a future capstone team can make better use of the citation data, and having this framework to combine it with the patent claims data would save them time. Furthermore, future capstone teams can understand that there are options for exploring the citation data, such as finding a SVD with more features that works under the memory constraints and looking into more graph-based clustering methods, but at the same time it is likely more useful to focus efforts on the patent claims data. The Guardian describes Apple suing Samsung for two billion dollars over stolen iPhone and iPad features in devices (2014). With such potentially severe consequences for patent infringement, accuracy is a high priority for our project, and I identified the citation data as only being a bit helpful at best in moving towards high accuracy.

Next, even if the features Nikhil and Martin generate are very accurate, K means, and many clustering algorithms may still cluster them unevenly, and my experience with the patent data, as well as my knowledge of Lloyd's algorithm, equips me to better handle these cases. I also possess experience with using K means and a couple other clustering algorithms that saves me time in my decision of which algorithm to use. In fact, in one instance using K means, I was

able to cluster some of their features into 5 relatively even clusters, and I am able to more evenly distribute them with Lloyd's if need be.

All in all, I was unable to complete my specific goal, but the work I did was still helpful to the capstone team as a whole and will be helpful to potential future capstone teams.

## Team Paper

(everything besides intro by Alper Vural, Martin Gouy, Nikhil Narayan)

### Abstract

Our capstone team performed various market and industry research to ensure that a web application to automate the process of searching for similar patents would be commercially viable. First, we define the minimum viable product for lawyers as a tool that could guarantee ease, speed, and reliability in the searching process. Next, we analyze the current competitive landscape and claim that our combination of novel machine learning algorithms could outperform existing solutions in semantic search. Lastly, we propose a go to market strategy that emphasizes the existence of a large need for our product. As part of our go to market strategy, we make a very rough estimate that our product could generate \$1.2 million per year.

### Introduction

Our capstone project involves building an application that will allow lawyers and inventors to efficiently search for similar patent applications. Currently, the process of finding similar patents is extremely labor intensive and often requires an experienced lawyer to sift through claims to determine how closely related two applications actually are. In order to justify the need for such a product, we present our preliminary market and industry analysis followed by the key components of our tech strategy.

One of the first steps in successfully commercializing a product involves identifying a specific target audience. To begin, a major problem faced by many inventors is determining whether their innovation can distinguish itself in the market so they can be granted exclusive use of that technology. Especially in recent years, successful patents have become increasingly

important for companies since they “represent a qualitative weighting of R&D output and thus reflect those elements of technological progress that lead to economic growth” (Ernst, 2001). Therefore, as companies continue to compete over intellectual property rights, Guellec argues that the patent landscape is becoming increasingly saturated (Guellec, 2008). Having come up with a patentable idea, inventors will typically hire lawyers to sift through existing grants to determine whether the application has a reasonable chance of being filed. Since inventors usually don’t have the necessary legal expertise, this process can be extremely expensive. Therefore, we believe that our target audiences are inventors and lawyers who wish to reduce the amount of time they spend manually searching through the patent space.




In order to provide significant market value, this application needs to satisfy some minimum functionality in retrieval and clustering to convince users to adopt the technology. Since our research team already has connections with certain law firms, we planned on targeting the minimum viable product towards lawyers, since they will most likely be early adopters. At a high level, instead of making the lawyer input different combinations of keywords into traditional search engines, this product would be able to take an entire application and retrieve similar patents. Most importantly, the retrieved patents have to pass a high bar of similarity, since lawyers will be using this data to see if there are grounds to block a new application. Second, this application should allow the lawyer to discover new clusters of patent applications. Though this feature is less critical than the previous one, lawyers often use patent data to conduct research about a novel field. After implementing the required set of features discussed above, this application will also need to meet certain metrics requirements on reliability and speed in order to be actually useful. We believe that efficiently implementing similar patent retrieval and clustering comprises our minimum viable product for lawyers.

## Industry Analysis

Before thinking about how we could monetize our tool and bring it to market, we first need to establish the competitive landscape in order to determine who are our main competitors, what they offer and how we could differentiate ourselves.

In order to draw this competitive landscape, we conducted an online research and eventually, we found a wiki maintained by an organism called PIUG (Patent Information Users Group) which contained an extensive and up-to-date list of the different “Patent analysis, mapping and visualization tools” (PIUG 2015). This list maintained by experts from the patent industry, happens to be a valuable resource, and since it is frequently updated, we will keep monitoring it in order to spot eventual new competitors. Figure 4 visually displays an overview of our analysis.

### Taxonomy of the competitive landscape

	 LexisNexis*	 CandorMap	 Google
<b>Type of company</b>	Established player in the IP solutions industry	Startup	Google patent is a minor service built on top of Google search
<b>Price</b>	On Demand	500\$ / month	Free
<b>Semantical Analysis</b>	Promoted but performances not verified	No	Not promoted but implemented
<b>Competitive Advantage</b>	Great number of features	Great visualizations	Easy to use / performance
<b>Weakness</b>	Costly for inventors	Few features	Not suited for lawyers (no possibility to save searches etc.)

*Figure 4: Analysis of Competitors*

The conclusions we can draw from our competitive analysis are the following. First of all, the competitive landscape is quite strong, probably stronger than what we expected when we started this project. The taxonomy of competitors varies a lot and goes from recently created startup such as Candormap to well established players such as Reedtech (a LexisNexis company). Candormap is an Israeliian Startup founded in 2014, they offer a mapping solution of the IP world,

promoting the use of semantic similarity. In comparison, Reedtech is a well-established player, they have a wide range of products that cover the whole life of a patent (patent filing/ claim analysis, litigation tools), they promote semantical search and they have a partnership with the USPTO. In addition to commercial solutions, we must mention the free solution offered by Google through their patent search engine. Their tool has been recently updated and now offers very interesting features such as a document similarity feature similar to the one we are working on.

The different products available on the market can be differentiated according to three factors: their patent retrieval engine, the data they use and the add-on features they offer in addition to the engine to make the experience less tedious. The description of their search engine is most of the time very vague, however, and some of our competitors promote, as we do, semantical search. Regarding the data they use, solutions such as the one offered by Innography promote the use of international patent database, financial records, data about inventors and also include non-patent literature. Finally, most of our competitors offer add-on features on top of their search engine such as in-depth claim analysis, advanced visualizations and tools to track the evolution of the patent activity within an industry.

When it comes to their business model, they mostly adopted daily/monthly/yearly subscriptions systems with different plans depending on the number of features or number of databases the customer wants to have access to. Regarding the price, it is very difficult to have an idea since the subscription process requires contacting their sales team and asking for a cost estimate. However, Candormap offers their services for \$4800/year, which is presumably the lowest bound of the price range of the market.

Taking into account this competitive analysis, we decided to focus on creating a highly performant search/comparison engine in order to beat the performances of our competitors. We

have good reasons to think that even if we are not the only one to promote semantical search, the cutting edge machine learning techniques we use coupled with a combination of exclusive datasets (Litigation and blocking datasets) could allow us to outperform existing solutions. Furthermore, by focusing on working on a solid MVP, we create a solid base for eventual future Capstone teams to build on top of our work and make our product even more attractive to lawyers and inventors.

### Go to Market Strategy

The going to market strategy of our project is key to its success in the face of this competition, and such strategy includes identifying who we are as a company and who our market is. Naeem Zafar in a Berkeley graduate student lecture describes a startup in part through the following: the unmet need, the market size, any differentiated positioning, and the scalability of the business model (A. Vural, personal communication, January 4, 2016). The unmet need of efficient patent verification is described in our introduction. Our Industry Analysis section describes our competitors and how we are different.

One way of calculating market size is to consider that over 600,000 patent applications were filed to the USPTO in 2014 (US Patent Statistics Chart, 2016). For an inventor to ask a legal firm for help in the filing process, it can cost “\$1000 to \$3000” (Quinn, 2015, para. 11). Even though some of those who filed were not inventors, choosing the middle of this range would mean an estimated \$1.2B going to legal firms for 2014, assuming all of these applications involved paid consultation to a legal firm. Likewise, an IBISWorld Industry Report on Law Firms indicates revenue of about \$267B for 2014, and patent services for individuals are listed as a notable portion of the “Other services” category and can be estimated to be about 1% of total revenue (Morea, 2015). This would result in \$2.67B of revenue (1% of 267B), which is similar to the earlier estimate. Anything that heavily impacts this industry thus has potential for

significant profit. Our business model is therefore also scalable in the sense that it is potentially applicable to any company as well as any inventor seeking to file a patent.

Since filing help costs \$1000-\$3000 for an inventor, and likely a comparable amount but a bit less for a company, our product being able to do a similar job for a smaller value would be very helpful. We can assume that we would charge a yearly subscription with price equivalent to about \$100 per patent filed. Of the about 600,000 patents filed last year, we can make the rough estimation that 2% of these would be interested in our tool. Factors involved in this include our assumption that our tool is perfected by future capstone teams to have verified high performance, the fact that other tools do not have access to the same patent database, the fact that our tool does not provide the legal knowledge of a lawyer, and our flexibility to target inventors, companies, and/or law firms. Alternate approaches would involve targeting the law firms and pricing based on how much time we would save the lawyers, but with this approach, the given percentage of the patents means about 12,000 patents per year at \$100 each, would mean \$1.2 million of revenue.

Zafar also described the potential importance of market segmentation for a startup trying to break into an industry. Two key components of a Go to Market Strategy include segmentation and product (A. Vural, personal communication, January 6, 2016). We here discuss the first two. Our product, as previously described, will consist of a web UI to take in a patent and give back similar/invalidating patents. Only customers will be able to access the web UI.

## Conclusion

Despite a variety of uncertain factors regarding the profitability of our capstone project, we are confident that we can at least make progress toward creating something that people need.



## Appendix

### Modified K Means Instructions

The following code is written in Python 2.7.10. It requires the files containing the patent citation data and the patent invalidation data in the same folder to run. It converts the citation data into a 2-dimensional “samples” and “features” matrix, where each sample or row is a patent and the  $i$ th feature is a 1 or 0, depending on if that patent cites the  $i$ th patent. This two-dimensional matrix is implemented as a sparse matrix due to memory constraints. The matrix has its dimensions reduced using Sci-Kit Learn’s singular value decomposition and then is given as input into Sci-Kit Learn’s K Means clustering algorithm. The groupings that are outputted by this algorithm are more evenly distributed using Lloyd’s algorithm. From there the accuracy of these groupings is checked using the invalidation data set. This approach should achieve an accuracy of about 55% for 3 groupings, which is noticeably better than the expected baseline of 33% yet still not anywhere near the almost-perfect accuracy we had hoped for. It is straightforward to replace K Means with another Sci-Kit Learn clustering algorithm and leave the rest of the code the same. To verify the information loss from the SVD, use the `explained_variance_ratio_` property.

### Modified K Means Code

```
# -*- coding: utf-8 -*-
from scipy.sparse import csr_matrix
from sklearn.cluster import KMeans
from scipy.cluster.vq import kmeans
import numpy as np
import time
from scipy.spatial.distance import cosine
from sklearn.decomposition import TruncatedSVD
#import matplotlib.pyplot as plt

def compute(numdocs, nclust, feats):
    start = time.time()
    numDocs = numdocs
    f = open('citation.network.tsv', 'rb')
```

Alper Vural

```

#maps my given document name starting from 0 to the actual document name
indexToNameMap = {}
indexToCitesMap = {}
#reverse mapping of previous
nameToIndexMap = {}

curIndex = 0
rows = []
columns = []
data = []

#Only use the 7 digit format documents
def is7Digit(doc):
    return len(doc) == 7 and doc.isdigit()

uniqueCites = {}

for i in range(0, numDocs):
    split = f.readline().split()
    cites = []
    for ind in split[1:]:
        if(ind != '|'):
            cites.append(ind)
    doc = split[0]
    if(is7Digit(doc)):
        if(doc not in nameToIndexMap): #doc cites itself
            nameToIndexMap[doc] = curIndex
            indexToNameMap[curIndex] = doc
            indexToCitesMap[curIndex] = cites
            indexToCitesMap[curIndex].append(doc)
            curIndex += 1
        else: #has already been cited
            docInd = nameToIndexMap[doc]
            indexToCitesMap[docInd] = cites
            indexToCitesMap[docInd].append(doc)
    rows.append(nameToIndexMap[doc])
    columns.append(nameToIndexMap[doc])
    data.append(1)
    for citation in cites:
        if(is7Digit(citation)):
            if(citation not in uniqueCites):
                uniqueCites[citation] = True
            if(citation not in nameToIndexMap):
                nameToIndexMap[citation] = curIndex
                indexToNameMap[curIndex] = citation
                curIndex += 1
            rows.append(nameToIndexMap[doc])
            columns.append(nameToIndexMap[citation])
            data.append(1)

L = len(nameToIndexMap.keys())
csr = csr_matrix((data, (rows, columns)), shape=(L, L))

svd = TruncatedSVD(n_components = feats, random_state=42)

csr = svd.fit_transform(csr)
print(csr.shape)

nclust = nclust
km = KMeans(n_clusters = nclust)
#km = KMeans(n_clusters = nclust, n_jobs = 10)
fit = km.fit(csr)
#print(fit.cluster_centers_)
p = fit.predict(csr)

```

Alper Vural

```

f.close()

#more evenly cluster documents
medianCount = 0.0
centers = fit.cluster_centers_
a = np.zeros(csr.shape[0])
p = np.zeros(csr.shape[0])
for i in range(0, csr.shape[0]):
    a[i] = np.linalg.norm(csr[i] - centers[0])
median = np.median(a)
for i in range(0, csr.shape[0]):
    if(a[i] > median):
        p[i] = 1
    if(a[i] == median):
        medianCount += 1
        p[i] = 2

#organize the document clusters to be easily reached
patIdToCluster = {}
clusterSizes = {}
for i in range(0, 3):
    clusterSizes[i] = 0
for i in range(0, p.shape[0]):
    cluster = p[i]
    clusterSizes[cluster] += 1
    patId = indexToNameMap[i]
    patIdToCluster[str(patId)] = cluster
#calculate the percentage the largest cluster is of all documents
sizes = clusterSizes.values()
maxSize = 0
maxClust = 0
for i in range(0, nclust):
    if(sizes[i] > maxSize):
        maxClust = i
        maxSize = sizes[i]
maxPercent = maxSize / float(curIndex)
#now score
numRight = 0
testSize = 0
spooft1 = str(6481691)
spooft2 = str(5138145)
first = False
f = open('invalidation.txt', 'rb')
for line in f.readlines():
    pid1, pid2 = line.split('\t')
    pid2 = pid2[:-1]
    if(first):
        pid1 = spooft1
        pid2 = spooft2
        first = False
    if(pid1 in nameToIndexMap and pid2 in nameToIndexMap):
        c1 = patIdToCluster[pid1]
        c2 = patIdToCluster[pid2]
        testSize += 1
        if(c1 == c2):
            numRight += 1
f.close()
score = numRight / (testSize + .001)
end = time.time()
return (end - start, score, testSize, maxPercent, maxClust,
medianCount/csr.shape[0])

clusters = 2
featsarr = [200, 300]

```

Alper Vural

```
for f in range(0, 2):
    for i in range(6, 7):
        docs = int(5 * np.power(10,i))
        secs, score, testSize, maxPercent, maxClust, medianAmount =
compute(docs, clusters, featsarr[f])
        f = open('outB' + str(i) + "F" + str(featsarr[f]) + '.txt', 'wb')
        string = 'Clustered ' + str(docs) + ' documents into ' +
str(clusters) + \
        ' clusters in ' + str(secs) + 's.\nThis run scored ' +
str(round(score*100,2)) + \
        '% with a test size of ' + str(testSize) + '.\n Cluster ' +
str(maxClust) + \
        ' was the largest cluster with ' + str(round(maxPercent*100,2)) + '%
of the patents.\n\
        This was trial 10.\n ' + str(round(medianAmount*100, 2)) + \
        '% of the vectors had the median distance from the first cluster
center.'
        f.write(string)
        f.close()
print('Done!')
```

## Patent Feature Clustering Instructions

Note that this code was not discussed in this paper, as it was not the main focus of my work, but it was an important contribution to the project as a whole. This code is written in Python 2.7.10. It requires the patent citation data file, the invalidation data file, and a dictionary mapping patent ids to a vector of feature values in the same folder to run. The features are intended to have been derived from the patent text. This algorithm groups the patents by repeatedly selecting a random patent and making a group out of the patents with the smallest cosine distances to the randomly selected patent. Once a patent is already assigned to a group, it is ignored in future selections. In order to combat the randomness of this algorithm, it is run a large number of times, and the “validation” accuracy of each trial is measured using the citation data. The grouping with the highest “validation” accuracy is selected as the final grouping. The final accuracy is measured using the invalidation data. Even with the repeated trials, this algorithm is extremely inconsistent and achieved accuracies ranging from 60%-90%. Still, it in general performed better than the standard clustering algorithms.

## Patent Feature Clustering Code

Alper Vural

```

import numpy as np
from sklearn.cluster import DBSCAN as db
from scipy.spatial.distance import cosine as cos
from numpy.linalg import norm
import pickle

#finds the indices of the values with the n closest 1-cos distance to a
random sample
def nSimilar(arr, n, noList):
    i = np.random.choice(arr.shape[0])
    samp = arr[i]
    dists = []
    for i in range(0, arr.shape[0]):
        dists.append( ( i, cos(arr[i], samp) ) )
    dists.sort(key = lambda x: x[1])
    out = []
    soFar = 0
    for i in range(0, arr.shape[0]):
        if(soFar == n):
            break
        ind = dists[i][0]
        if(ind not in noList):
            out.append(ind)
            soFar += 1
            noList[ind] = 1
    return np.array(out)

f = open('outAlpersA3.txt', 'wb')
f3 = open('outAlpersB3.txt', 'wb')

simsAccs = []
patent_dict = pickle.load( open( "patDict.p", "rb" ) )
patent_ids = patent_dict.keys()
pidToIndex = {}
for i in range(0, len(patent_ids)):
    pidToIndex[str(patent_ids[i])] = i
patent_values = [patent_dict[pid] for pid in patent_ids]
patent_vectors = np.array(patent_values, ndmin = 2)
a = patent_vectors
s = a.shape[0]
a = a[:s]

citationPairsB = []
count = 0
fx = open('citation.network.tsv', 'rb')
for i in range(0, 5000000):
    split = fx.readline().split()
    doc = split[0]
    cites = []
    for ind in split[1:]:
        if(ind != '|'):
            cites.append(ind)
            if(ind in pidToIndex and doc in pidToIndex):
                citationPairsB.append((doc, ind))
fx.close()

bestAcc = 0
bestMissPairsInv = 0
bestMissPairsCit = 0

for i in range(0, 50000):
    clusters1 = np.ones(s)
    nclust1 = 3
    noList1 = {}

```

Alper Vural

```

noList1[-1] = 1
for i in xrange(nclust1):
    inds = nSimilar(a, s/nclust1, noList1)
    clusters1[inds] = i
centers = np.zeros((nclust1, 500))
for i in xrange(clusters1.shape[0]):
    centers[clusters1[i]] = np.add(centers[clusters1[i]], a[i])
for i in xrange(nclust1):
    centers[i] /= s/nclust1
hit = 0
tot = 0.01
missedPairsInv = []
f2 = open('invalidation.txt', 'rb')
for line in f2.readlines():
    pid1, pid2 = line.split('\t')
    pid2 = pid2[:-1]
    if(pid1 in pidToIndex and pid2 in pidToIndex):
        tot += 1
        if(clusters1[pidToIndex[pid1]] == clusters1[pidToIndex[pid2]]):
            hit += 1
        else:
            missedPairsInv.append((pid1, pid2))
f2.close()
acc = hit/tot
tot = 0.01
hit = 0
missedPairsCit = []
for pid1, pid2 in citationPairsB:
    if(pid1 in pidToIndex and pid2 in pidToIndex):
        tot += 1
        if(clusters1[pidToIndex[pid1]] == clusters1[pidToIndex[pid2]]):
            hit += 1
        else:
            missedPairsCit.append((pid1, pid2))
accB = hit/tot
if(accB > bestAcc):
    bestAcc = accB
    bestMissPairsInv = missedPairsInv
    bestMissPairsCit = missedPairsCit
sim = 0
tot = 0
for i in range(0, nclust1):
    for j in range(i+1, nclust1):
        sim += 1 - cos(centers[i], centers[j])
        tot += 1
simsAccs.append((sim / np.float(tot), acc, accB, nclust1))
f.write(str(nclust1) + "cluster sim: " + str(sim / np.float(tot)) + ",
invalidation acc: " + str(acc) + \
", citation acc: " + str(accB) + "\n")
f.flush()
if(bestAcc > .96):
    break

f.close()
simsAccs.sort(key = lambda x: x[2])
for sa in simsAccs:
    f3.write(str(sa) + "\n")
f3.write(str(bestMissPairsInv) + "\n")
f3.write(str(bestMissPairsCit) + "\n")
f3.close()

```

## References

- Adams, C. (2008, July 24). Blocking Patents and the Scope of Claims. Retrieved October 18, 2015.
- Apple sues Samsung for \$2bn as tech rivals head back to court. (2014, March 30). Retrieved October 18, 2015, from <http://www.theguardian.com/technology/2014/mar/31/apple-sues-samsung-for-2bn>
- Efros, Alexei (2015). *Dealing with the Curse of Dimensionality* [PDF document]. Retrieved March 11, 2016, from [https://www.dropbox.com/s/zauh16bv6cz44o/nn2\\_f15.pdf?dl=0](https://www.dropbox.com/s/zauh16bv6cz44o/nn2_f15.pdf?dl=0)
- Ernst, Holger. "Patent Applications and Subsequent Changes of Performance: Evidence from Time-series Cross-section Analyses on the Firm Level." *Research Policy* 30.1 (2001): n. pag. Web. 18 Oct. 2015.
- Gonzalez, Joseph (2015). *Introduction to Graph Analytics* [PDF document]. Retrieved November 23, 2015, from <https://bcourses.berkeley.edu/courses/1377158/files/63059350/download?wrap=1>
- Guellec, Dominique, Catalina Martinez, and Pluvia Zuniga. "Blocking Patents: What They Are and What They Do." *Directorate for Science, Technology and Industry, OECD* (2008): n. pag. Web.
- Kahn, S. (2014, December). Search Engines in the US. Retrieved October 18, 2015.
- Manning, C., Raghavan, P., Schütze, H (2009). *An Introduction to Information Retrieval*. Cambridge University Press.
- Morea, S. (2015). Law Firms in the US. IBISWorld Industry Report 54111. Retrieved from IBISWorld database.
- Ng A., Jordan M., & Weiss Y. (2002). *On Spectral Clustering: Analysis and an Algorithm*.

Retrieved November 23, 2015, from <http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf>

Overview of the U.S. Patent Classification System (USPC). (2012, December).

Retrieved October 18, 2015.

Quinn, G. (2015, April 4). The Cost of Obtaining a Patent in the US. Retrieved October 18, 2015.

Steinbach M., Ertöz L., and Kumar V. *The Challenges of Clustering High Dimensional Data*.

Retrieved May 10, 2016.

Sci-Kit Learn. (n.d.). Retrieved March 11, 2016, from <http://scikit-learn.org/stable/>

US Patent Statistics Chart 1963-2014. (2016, February 4). Retrieved February 5, 2016.

Vinod Kumar Singh , Elisa Rodgers. “Patent Analysis, Mapping, and Visualization

Tools”. *PIUG.org*. pag. Web. Retrieved February 5, 2016

What is prior art? (2015, September 15). Retrieved October 30, 2015, from

<https://www.epo.org/learning-events/materials/inventors-handbook/novelty/prior-art.html>