Automated Disambiguation of US Patent Grants and Applications*

Benjamin Balsmeier**, Alireza Chavosh*, Guan-Cheng Li*, Gabe Fierro*, Kevin Johnson***, Aditya Kaulagi***, Doug O'Reagan*, Bill Yeh***, and Lee Fleming*

*UC Berkeley, Coleman Fung Institute for Engineering Leadership
**KU Leuven, Department of Managerial Economics, Strategy and Innovation
***UC Berkeley, Electrical Engineering and Computer Science

July 1, 2015

Abstract

We introduce a new database and tools that will ingest and automatically build an updated database using United States patent data. The tools disambiguate inventor, assignee, law firm, and location names mentioned on each granted US patent from 1975 to 2014 inclusive all applications 2001 to 2014. While parts of this first version are crude, all the approaches are fully automated and provide the foundation for much greater improvement. We describe a simple user interface, list descriptive statistics, illustrate an automated co-authorship network mapping tool, provide a lexical distance measure between all classes since 1975, and discuss future opportunities such as matching the disambiguated patent data with other frequently used datasets such as Compustat. The documentation and disambiguations can be found at `http://www.funginstitute.berkeley.edu`.

*This work is supported by NSF grant 1360228, the US Patents and Trademark Office, the American Institutes for Research, and the Coleman Fung Institute for Engineering Leadership; errors and omissions remain the authors'. Balsmeier gratefully acknowledges financial support from the Flemish Science Foundation.

Introduction

Patent data have been used to study invention and innovation for over half a century (see Hall et al., 2012 for a recent overview). Its popularity stems largely from the rich, consistent, and comparable information that can be obtained for a huge number of entities, i.e. organizations, individuals, and locations. Aggregating patents remains difficult because entities (inventors, assignees, applicant law firm, and location) are only listed by their names on each patent document and receive no unique identifier from the patent office (they essentially remain inconsistent text fields). Looking at these fields as they appear on the patent document reveals various forms of misspellings or correct but different name spellings. The ambiguous names further limit the possibility to assemble patent portfolios for research as it is difficult to foresee all the kinds of name abbreviations that can occur. As a result, individual researchers spend significant amounts of time and resources on labor-intensive manual disambiguations of relatively small numbers of patents. The intent of this paper is to provide a working prototype for automating the disambiguation of these entities, with the ultimate goal of providing reasonably accurate and timely disambiguation data.

A wide variety of efforts have recently been made to disambiguate inventor names (Fleming and Juda 2004; Singh 2005; Trajtenberg et al., 2006; Raffo and Lhuillery 2009; Carayol and Cassi, 2009; Lai et al., 2009; Pezzoni et al., 2012, Li et al. 2014). These efforts are gaining in sophistication, accuracy, and speed, such that fully automated approaches can now compete with smaller and hand crafted and manually tuned datasets.

Concurrent efforts have been made at the assignee level using automated and manual methods. Hall et al. (2001) disambiguated the assignees and introduced their patent data project under the auspices of the National Bureau of Economic Research (NBER). These data are widely used, partly because many assignees have also been matched to unique identifiers of publicly listed firms, which in turn enables easy matching with various other firm level databases, e.g. Compustat. Producing updates of the NBER patent data is costly, however, due to the quite sophisticated but still labor-intensive assignee disambiguation process.

Many patent applications contain information about the applicant's law firm (though an inventor or his/her employer need not retain a law firm, many do); however, to the best of our knowledge, this field has never been disambiguated. Identifying unique law firms could aid in understanding patent pendency and disambiguation of other fields. Location data are available for most inventors (their home towns in particular) and while these data have been used previously, there does not exist a comprehensive and automated approach to their disambiguation.

To further aid the research community, this paper also includes application data for all these fields as well. Almost all research to date has used granted patent data, which ignores the many applications that were denied, and probably introduces bias. Application data have been recently made available by the USPTO, but to our knowledge, have not been made easily available to the research community yet. A simple but automated tool is also provided that graphs reasonably sized co-authorship networks in real time, based on seed inventors or a specification of a particular technology classification. We also provide a pairwise distance measure between the 600 US patent classes that have been used since 1975, based on the

lexical overlap of words in abstracts, description, and claims.

Here we introduce four automated disambiguation algorithms: inventor, assignee, law firm, and location. All need much further accuracy improvement but they are fast enough to disambiguate over 7 million patents and pending applications within days, and thus could enable automated updates on all USPTO granted patents and applications. Application data since 2001 are included as well. A simple user interface that emails data to users is available and documented below. Code is available from the last author to any non-profit research organization. For updates and addresses to authors, data, tools and code, please see `http://www.funginstitute.berkeley.edu/`.

Data Sources

Figure 1 illustrates the automated tool chain for processing the patent data (for technical details see Fierro, 2013). As data is downloaded from the USPTO weekly patent releases hosted by Google, they are parsed, cleaned and inserted into a SQL database. The output data from these processes are combined with data from the Harvard Dataverse Network (DVN, see Li et al. 2014), which comes from from the National Bureau of Economic Research (NBER), weekly distributions of patent data from the USPTO, and to a small extent, the 1998 Micropatent CD product. Disambiguations begin from this database.

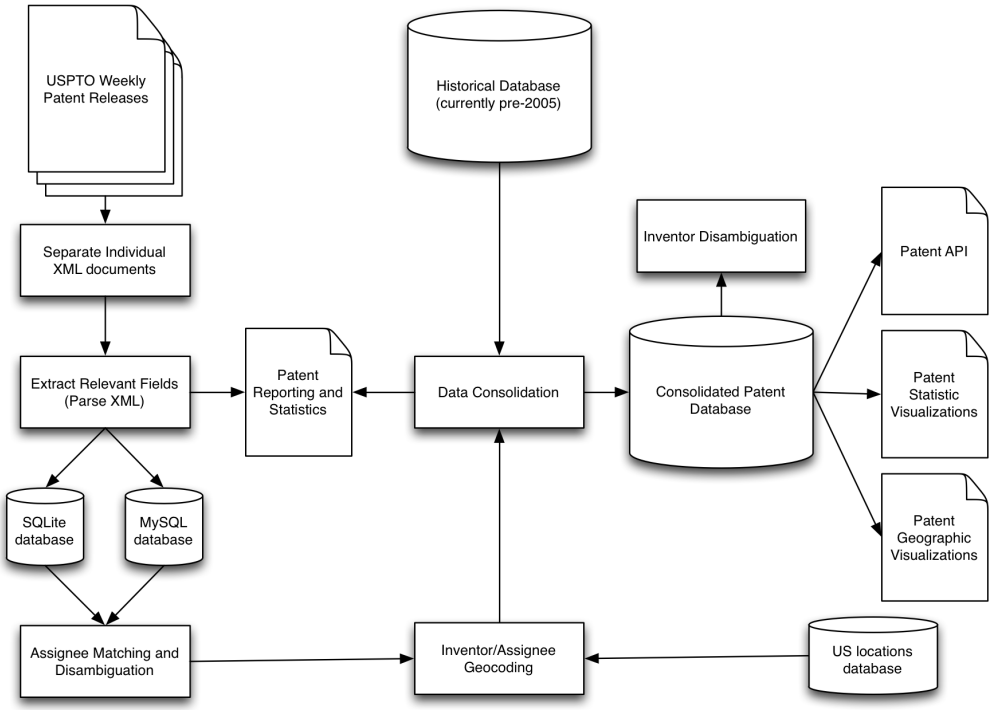

Figure 1: Tool and data flow for automated disambiguations of US patent grant and application data.

Every week, the USPTO releases a file that documents the patents granted the prior week. Since the inception of digital patent record release in 1975, the USPTO has used no less than 8 different formats for representing granted patents alone. Patents granted since 2005 are available in various XML (Extensible Markup Language) schemas, which allows the development of a generalized parser to extract the relevant information from the patent documents. The Fung Institute parser transforms XML documents from the canonical tree-based structures into lookup tables. Typical XML parsers must be tailored to the exact structure of their target documents, and must maintain a careful awareness of contingent errors in the XML structure. The Fung Institute parser circumvents this issue by using generalized descriptors of the desired data instead of explicit designation. This robust approach allows the parser to handle small schema changes without adjustment, decreasing the need for constant development.

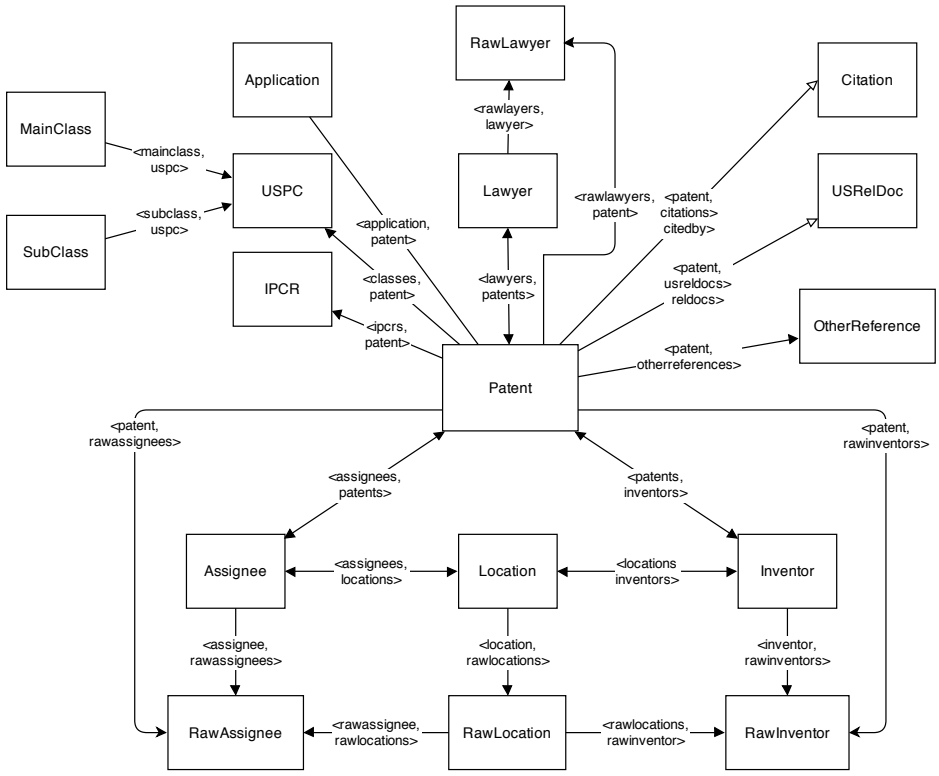

Figure 2: Abstract view of relational database for patent grant and application data.

As the data are extracted from USPTO documents, they are streamed into a relational database that contains records linking patents, citations, inventors, law firms, assignees, technology classes and locations. The database itself uses the MySQL database engine, but the patent library uses an object relational mapper (ORM) to allow database entries to be manipulated as Python objects. This simplifies development by removing the need to write code for a specific database backend, and facilitates use by not requiring the user to be familiar enough with relational databases in order to query and manipulate data. The raw tables

in the database contain data as it appears in the patent record; this preserves the resolution of the data and gives the user freedom to develop their own processes over the original data. As the disambiguations are run, the raw records are linked to the disambiguated records.

Algorithms

Inventors[1]

We treat inventor disambiguation as a clustering problem; to disambiguate names, we seek high intra-cluster similarity and low inter-cluster similarity amongst patent-inventor pairs. While conceptually simple, the challenge is to accurately and quickly cluster approximately 5 million patents and 2 million applications and over 13 million names (roughly 17 million when including inventors from non-granted applications). Inventor disambiguation uses the assignee and location disambiguations described above; as a result, the inventor disambiguation runs last.

Step 1: Vectorization

We define a document unit of a patent as an unordered collection of that patent's attributes. The current version attributes are 1) inventor name 2) co-authors (implicitly, by association on the same patent), 3) the full text description of assignees, 4) the full primary patent class and lexical distance between classes when the primary class does not match exactly, and 5) the city, state, and country location. The occurrence of each attribute is used as a feature (or independent variable) for training a classifier. We build a document-by-attribute incidence matrix, which is sparse because a patent cannot use all possible attributes. Figure 3 illustrates.

[1]The following is developed directly from an internal technical report by Guan-Cheng Li: `http://www.funginstitute.berkeley.edu/publications.`

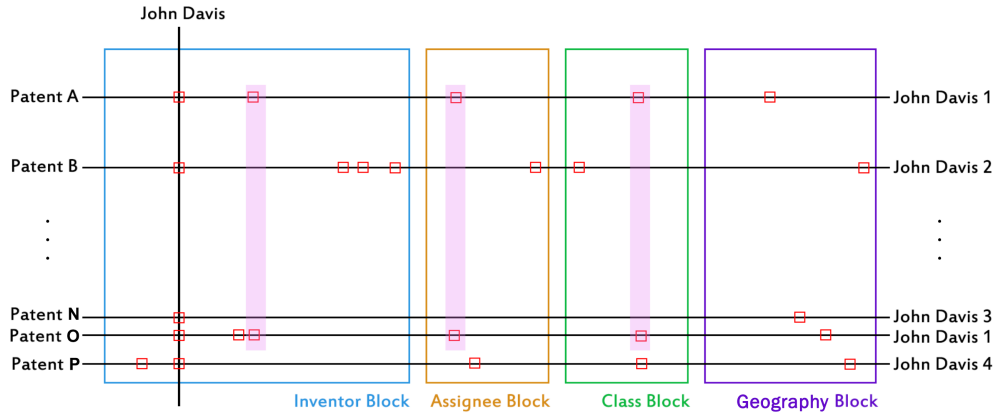

Figure 3: Document by attribute matrix.

Suppose one or more inventors named John Davis have filed five patents, A, B, N, O, P. Let the inventors named John Davis initially be one column (depicted by the black vertical tab in Figure 3 - essentially blocking on the exact name "John Davis"). We look closer at the five rows that have 1s along that column, namely the index of the patents being filed by John Davis. Having formed a sub-matrix from just these five rows and their attributes, we can compute correlations between rows (patents) and columns (inventors).

Step 2: Distance Measurement

Distance measurements can be computed in a number of ways, e.g., Euclidean distance, Manhattan distance or Chebyshev distance. Such measurements enable quantification of the dissimilarity between inventors. Here, we adopt a discrete Euclidean distance. An exact match is at a distance of 0; a non-match is at a distance of 1. All distances currently require exact matches and hence imply a discrete 0 or 1 distance, with the exception of the technology class distance.

$$d(\vec{x} - \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})} \tag{1}$$

The distance between technology classes is a combination of an exact match when it occurs, or a positive correlation, based on the aggregated words in each patent within the two classes that do not match exactly. These distances between classes are available at `http://funglab.berkeley.edu/guanchengli/classdist.full.php`; a subset that creates "class clusters" that are cut off at 0.2) is at `http://funglab.berkeley.edu/guanchengli/classdist.php`; an example of from the "class cluster"file is:

- CLASS 027 (UNDERTAKING) - correlation 1.0000
- CLASS 217 (WOODEN RECEPTACLES) - correlation 0.0370
- CLASS 493 (MANUFACTURING CONTAINER OR TUBE FROM PAPER; OR OTHER MANUFACTURING FROM A SHEET OR WEB) - correlation 0.0327
- CLASS 229 (ENVELOPES, WRAPPERS, AND PAPERBOARD BOXES) - correlation 0.0273

Step 3: Bottom up K-means merging

Common names usually represent multiple persons, e.g., John Davis. If, for a toy example, there were four such different people across the USPTO dataset, the algorithm should cluster and report four unique identifiers for John Davis.

From step 1, we take the short and very wide matrix of exactly similar inventor names and their attributes. To initialize, we treat each of these observations as a distinct person, by assigning unique initial identifiers. Then, we cluster all these unique identifiers into one block (there would be five in this toy example of John Davis). We then apply the K-means clustering algorithm based on a bottom-up approach (that is, we start with five clusters). The result is that some of these names might be merged and replaced with a single identifier. K-means places an observation into the cluster with the most similar mean.

Step 4: Lumping name aliases

Once the k-means bottom up merging completes, we assign the centers of each cluster unique identifiers and augment the matrix column-wise by their unique identifiers, as depicted in Figure 5. In other words, we uniquely label each cluster believed to be a different John Davis and apply this labeling to each column. This results in columns being split, and hence the number of columns in the matrix increases, as separate inventors are identified. Compare Figures 3 and 4, where Figure 3 has one column marked John Davis and Figure 4 has four columns.

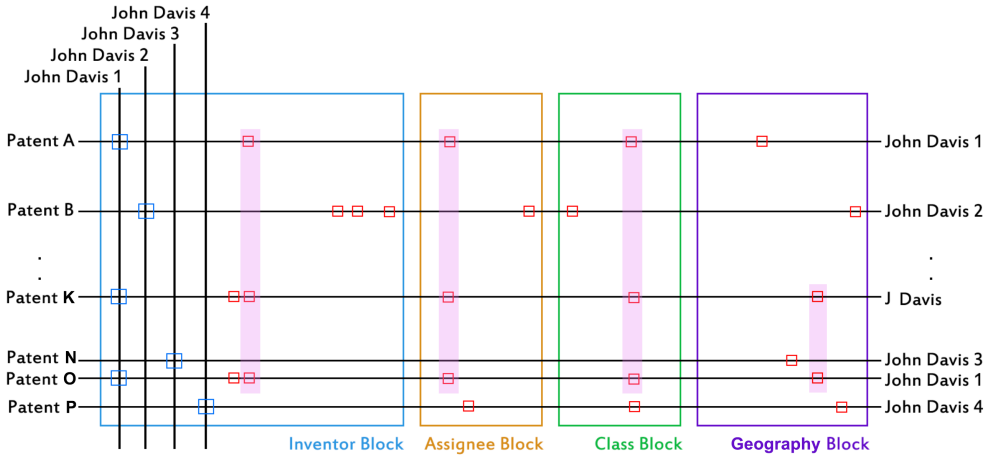

Figure 4: Augmentation with unique identifiers: as individuals are identified, additional columns are split out.

By lumping, we mean to merge naming aliases into one person if, in fact, they should be one person based on what the other factors, such as co-inventor, assignee, class, or city, suggest. This step is designed to treat Jim and James, Bob and Robert, or, Arnaud Gourdol, Arnaud P Gourdol, Arnaud P J Gourdol, and Arno Gourdol as same persons (in our toy example, J. Davis might be one of our inventors named John Davis). The algorithm assigns inventors to the nearest cluster by distance. There are three possibilities for the naming match which results in a combination or lumping of the clusters: 1) the last names agree, the first names disagree, and the first letter of the middle names agree (if any), for example, Jim and James, 2) the last names agree, the first names disagree, and the first letter of the first names disagree, for example, Robert and Bob, 3) the last names disagree, the first names agree, and the first letter of the middle names agree (if any), due to marriage and last name change.

Summary and some technical details

The goal of the automated K-means algorithm is to produce high intra-cluster and low inter-cluster similarity between inventors. The objective function to be optimized is expressed in Equation (2) as:

$$\phi^{(Q)}(\boldsymbol{X}, \lambda) = 1 - \frac{\sum_{i=1}^{k} \frac{n_i}{n - n_i} \sum_{j \in \{1,\cdots,i-1,i+1,\cdots,k\}} n_j \cdot \text{inter}(\boldsymbol{X}, \lambda, i, j)}{\sum_{i=1}^{k} n_i \cdot \text{intra}(\boldsymbol{X}, \lambda, i)} \tag{2}$$

$\phi^{(Q)}$ represents the cluster quality. If the quality is 0 or lower, two items of the same cluster are, on average, more dissimilar than a pair of items from two different clusters. If the quality rating is closer to 1, it means that two items from different clusters are entirely dissimilar, and items from the same cluster are more similar to each other. This will result in a denser k-mean. Figure 5 illustrates the relationship for our toy example - there is no additional benefit past four.

We determine the cluster size by minimizing the sum of each clusters' sum of the inter-cluster's squared distances. As illustrated in Figure 4, the five names, John Davis, are grouped down to four clusters. When the sum no longer decreases, we have reached a local minimum. In other words, increasing the cluster number to five wouldn't decrease the objective function and hence we stop at four clusters for John Davis.

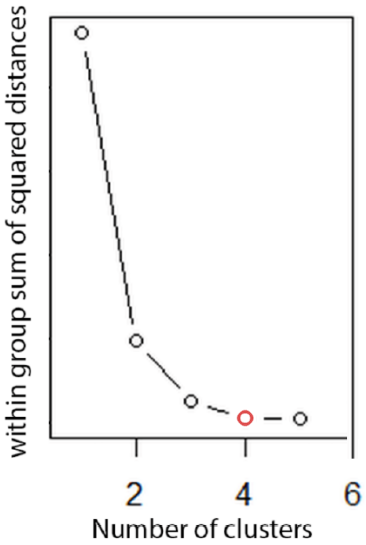

Figure 5: The optimal number of clusters is determined by the point of decreasing returns (4 in this example).

The distance between classes is calculated from lexical correlations of words in the abstract, claims, and descriptions of each patent. For a given class C, we aggregate the words of the patents that are classified into C (note, a patent can have multiple classes). These form a one by two million vector into a data point, call it D (note, there are about two million words that are used at least once in the USPTO patent titles, abstracts and claims).

We define the tech distance between class C1 an C2 by correlating D1 and D2, which is cov(D1,D2) / (std(D1)*std(D2)). By doing so, we let the textual info speak for themselves. A cheaper way is to calculate class co-occurrences, where the accuracy is dictated by USPTO alone.

In summary, the disambiguator considers each inventor's co-inventors, geographical location, technology class, technology class distance, and assignee to determine lumping and splitting of inventor names across patents. One advantage of vectorizing each inventor name's

attributes is that it allows for easy expansion of new attribute blocks to be compared.

Relative to previous efforts that disambiguated the entire U.S. patent corpus (Lai et al. 2009; Li et al. 2014), our splitting error for patents from January of 1975 through April of 2014 was 1.9% and our lumping error was 3.8%. The current method is much simpler and runs much faster and has approximately 1/10th of the code of Lai et al. 2009, however, it is slightly less accurate. Accuracy is assessed in the same manner as Li et al. 2014, with the exception that patents from May to December of 2014 are not assessed. Future work will update the comparison file; the current file is at `http://funglab.berkeley.edu/guanchengli/disambig.golden.list.txt`.

Assignees

The assignee algorithm is much less sophisticated than the inventor algorithm; rather than using a classifier to cluster individual entities, the assignee algorithm applies a crude Jaro-Winkler string comparison. This method will usually group bad spellings and typographical errors correctly, but fails if the strings become too dissimilar. Unfortunately, organizations are often misspelled or alternatively abbreviated and listed in full with little modicum of consistency. Assignees will change their names, often within a given year, and unpredictably list name affixes.

For a given patent, the assignees are the entities, i.e. organizations or individuals that have the original property rights to the patent. The assignee records are used to determine the ownership of patents at the point the patent was initially applied for. This is sometimes helpful for studies that use archival data, but it limits the possibility to assemble patent portfolios of firms that bought or sold ownership rights to specific patents or firms that were involved in mergers and acquisitions (M&As) over time. Ownership changes are not tracked by the USPTO or any other institution yet though this may change (Stuart 2013). We will outline some possible future directions below.

Consider the following results from a cursory search for assignee records that resemble General Electric:

- General Electric Company
- General Electric
- General Electric Co..
- General Electric Capital Corporation
- General Electric Captical Corporation
- General Electric Canada
- General Electrical Company
- General Electro Mechanical Corp

- General Electronic Company
- General Eletric Company

This incomplete list of some of the (mis)representations of General Electric demonstrates the most basic challenge of getting past typos and misspellings of the same intended entity. We provide an automated entity resolution for assignee records by applying the Jaro-Winkler string similarity algorithm to each pair of raw assignee records (for details see Herzog et al. 2007). The Jaro-Winkler algorithm was developed in part to assist in the disambiguation of names in the U.S. Census, and is the most immediately appropriate string similarity algorithm for the task of resolving organization names. Two records with an overlap of 90% or better are linked together. First, all assignees are associated with a "clean identifier", which consists of the organization name (or concatenated first and last names) of the assignee, lower cased, with all non-letter and non-whitespace characters removed. This simplifies the comparison process. Following this normalization, all assignees are placed into a block according to the first letter of their clean identifier. This reduces computational demands and allows us to hold a smaller chunk of the assignees in memory at each step, and achieves similar accuracy.

Disambiguation occurs within blocks, resulting in a set of "pools" indexed by a central assignee and containing assignees that are within some Jaro-Winkler threshold of that central assignee. As assignees are popped off the end of the list of non-disambiguated assignees, they are compared against each of the central assignees. If their clean identifier is within the Jaro-Winkler threshold of some central assignee, then the candidate is placed in that pool; else, it is placed into a new pool of which it is the only member. This continues until all assignees are placed into a pool. A record is chosen from the pool to act as the disambiguated record for that pool, and all raw assignee records are linked to that disambiguated record.

The assignee and law firm disambiguation process follow the same basic model. The unaltered records differ in structure from the inventor records because the majority of assignee and law firm records are organizations rather than individuals. Entity resolution of personal names suffers primarily from duplicates - people who have the same name but are not the same individual. Organizational names are intrinsically free of this sort of false clumping. Instead, the disambiguation of organizational names is made difficult by three common patterns: acronyms (e.g. "IBM" for "International Business Machines"), corporate affixes ("Corporation" vs "Corp" vs nothing at all) and general misspellings.

Locations [2]

A variety of location data are associated with patents, though unfortunately those data are not always consistently available. We exploit the most consistent data, that of inventors' hometown, which provide two main challenges. First, we must identify ambiguous locations,

[2]The following is taken from an internal technical report by Kevin Johnson: `http://www.funginstitute.berkeley.edu/sites/default/files/GeocodingPatent.pdf`.

matching them consistently with real location. Second, we must assign latitude and longitude information to these locations, allowing them to be easily mapped.

There are over 12 million locations present in the patent files provided by the USPTO. Each location is split into up to five fields, depending on what information is available: street address, city, state, country, and zipcode. When non-unique locations are filtered out, there are roughly 900,000 unique locations to identify. However, not all of these unique locations are relevant.

It is rare for all five fields to be present; for example, only 6.5% of locations have any information in the street or zipcode fields. Some locations contain street-level data in the city data field, making it difficult to understand exactly how precise the data are. However, we believe that the vast majority of locations are only precise to the city level. In addition, there is relatively little value in being accurate to a street level as opposed to the city level, since most analysis takes place at a city or state level. Therefore, we disregard all street and zipcode information when geocoding the data. Avoiding these data also minimizes privacy concerns for the inventors.

After disregarding the street and zipcode fields, there remain roughly 350,000 unique locations to analyze. These locations are poorly formatted and difficult to interpret for many reasons.

Accents are represented in many different ways in the data. Often, HTML entities such as Å are used. However, not all representations are so straightforward. For example, all of the following strings are intended to represent the letter , often referred to as an angstrom: ".ANG.", ".circle.", "Å", "dot over (A)", and "acute over (Å)". These must be cleaned and converted into single characters.

Some foreign cities contain additional information that must be identified and dealt with consistently. For example, many cities in South Korea end with the suffix "-si", which indicates that the location is a city - as opposed to a county, which ends with the suffix "-gun". These suffixes are represented in a variety of ways, and should be interpreted consistently.

In some cases, data is recorded incorrectly on a consistent basis. For example, locations in the United Kingdom are often recorded with the country code "EN," and locations in Germany can be recorded as "DT."

In some cases, correct data for one field may be incorrectly assigned to a different field. For example, there are seven entries for a location with a city field of San Francisco and a country field of CA. There is no city named "San Francisco" in Canada; instead, "CA" was erroneously placed into the country field instead of the state field. This problem is especially prevalent with foreign names. The state and zipcode fields only contain information for US locations. When such information exists for foreign locations, it is added to the city field - either in addition to or instead of the actual city.

All manner of creative and potentially ambiguous spellings can be found within the data. For example, all 31 of the following spellings are intended to refer to the "San Francisco" in California:

- San Francais

- San Francesco
- San Francico
- San Francicso
- San Francis
- San Francisc
- San Franciscca
- San Franciscio
- San Francisco
- San Francisco County
- San Francisco,
- San Francisco, both of
- San Franciscos
- San Franciscso
- San Francisico
- San Franciso
- San Francisoc
- San Francisoco
- San Francsco
- San Francsico
- San Francsicso
- San Frandisco
- San Franicisco
- San Franicsco
- San Franisco
- San Franscico
- San Franscisco

- San Fransciso
- San Fransico
- San Fransicso
- San Fransisco

This is by no means an exhaustive overview of the many ways that "San Francisco" can be spelled. Identifying and correcting these misspellings is an important challenge.

Converting location names from languages with different alphabets is a difficult task. To use a simple example, the city "Geoje" in South Korea is represented in six different ways: Geojai-si, Geojae-si-Gyungnam, Geoje, Geoje-si, and Geoji-si. The more complex the name, the more ways it can be converted into English, and the more difficult it is to identify what the name of the city is supposed to be from a given romanization. Before performing any disambiguation work, we first focus on cleaning the raw location data. After cleaning, each location consists of a comma-separated string of the format "city, state, country".

Because the format used to identify accents is so idiosyncratic, we individually identify and replace many accent representations using a handcrafted list of replacements. In addition, we automatically convert HTML entities to their corresponding Unicode characters.

We make some corrections for consistent error patterns that are difficult for our disambiguation method to decipher automatically. Though the list of corrections is small, this will be a major area of development going forward as we learn what kinds of locations are most difficult to interpret.

We deal with mislabeled states by using a format for cleaned locations that does not explicitly label the state and country fields. Though this slightly increases ambiguity, our disambiguation method is capable of interpreting the information. For our purposes, is better to have slightly ambiguous data than unambiguously false and misleading data. In addition, we automatically remove house numbers and postal code information from the city field.

In addition to the above, we perform a variety of minor alterations and corrections - pruning whitespace, removing extraneous symbols, and formatting the locations into a comma-separated format.

After cleaning the data, approximately 280,000 unique locations remain that must be disambiguated. For this process, we consulted with Jeffrey Oldham, an engineer at Google. He used an internal location disambiguation tool similar to that used by Google Maps and gave us the results. For each location, Google's API returns six fields:

- **city**, the full name of the city. For example, "San Francisco" or "Paris."
- **region**, the name or two-letter abbreviation of a state, province, or other major institutional subdivision, as appropriate for the country. For example, "TX" or "le-de-France."
- **country**, the two-letter country code corresponding to the country. For example, "US" or "FR."

- **latitude and longitude**, the latitude and longitude of the precise location found, depending on how much information is available.
- **confidence**, a number ranging from 0 to 1 that represents how confident the disambiguation is in its result. If no result is found, the confidence is given as -1.

Because the latitude and longitude data provided are more precise than we want them to be, we run the results of the disambiguation through the geocoding API again, giving each location a consistent latitude and longitude. Preliminary results suggest that we will be able to geocode more than 95% of all locations with reasonable accuracy. The most recent version of our code can be found online at GitHub (Johnson 2013).

Applications

Treatment of application data since 2001 is similar to that of grant data, demonstrating the ease with which the current preprocessing framework handles new formats. To gather data from the six SGML and XML patent application formats released by the USPTO within the past decade, we have developed three application data format handlers which conform raw application data to a database schema similar to the one defined for patent grants.

Application parsing also fits the patterns established for gathering grant data: the same processes of parsing, cleaning, and eventually, full disambiguation, are automated for applications (in fact, the disambiguations are performed jointly between the two datasets). In addition to the old configuration options that allowed users to specify which processes to run and which data files to use for patent grants, there are new options to process either grants or applications, or both. Therefore, the same commands and programs previously used only for grants can now process grants and applications, with minimal configuration changes.

The raw application data is ingested into a database table separate from grants. Applications which are granted will be flagged and their records linked. While there are a few differences between the application schema and the grant schema - grants have references to their application documents, for example, while applications, of course, do not - the crucial tables involving inventors, assignees, lawyers, and locations remain the same, in both their raw and cleaned forms. The uniformity of application data and grant data ensures that our current means of disambiguation and access are easily adapted to the new application data, simply by changing the schema from which our ORM draws its data.

Results

We will post a simple diagnostic readout for basic descriptive statistics and some graphs (please search the Fung Institute Server if the interface is not at http://rosencrantz.berkeley.edu/; bulk downloads are available at http://rosencrantz.berkeley.edu/batchsql/downloads). This section highlights some of those diagnostics. Kia Silverbrook is the most prolific U.S. inventor with 4,666 patents (a March 26 2014 Wikipedia entry lists 4,665 patents). Table 1 lists the raw and disambiguated patent grant data for each disambiguation where available. Figure

6 illustrates yearly grants 1975 to 2014, inclusive. Figures 7 and 8 compare the number of cities in each state where patenting has occurred between 1975 and 2014; note the dramatic decrease in the number of cities and relative increase in New York and Pennsylvania after disambiguation.

	Number of observations in raw data	Number of observations in disambiguated data
Granted Patents	5,399,259	5,399,259
Applications	5,328,752	5,328,752
Inventors	12,429,538	3,752,775
Assignees	5,128,871	387,500
Law Firms	1,374,634	121,268
Locations (U.S. Cities)	61,840	16,114

Table 1: Descriptions of raw and disambiguated data. Except for applications, all numbers refer to only granted patents.

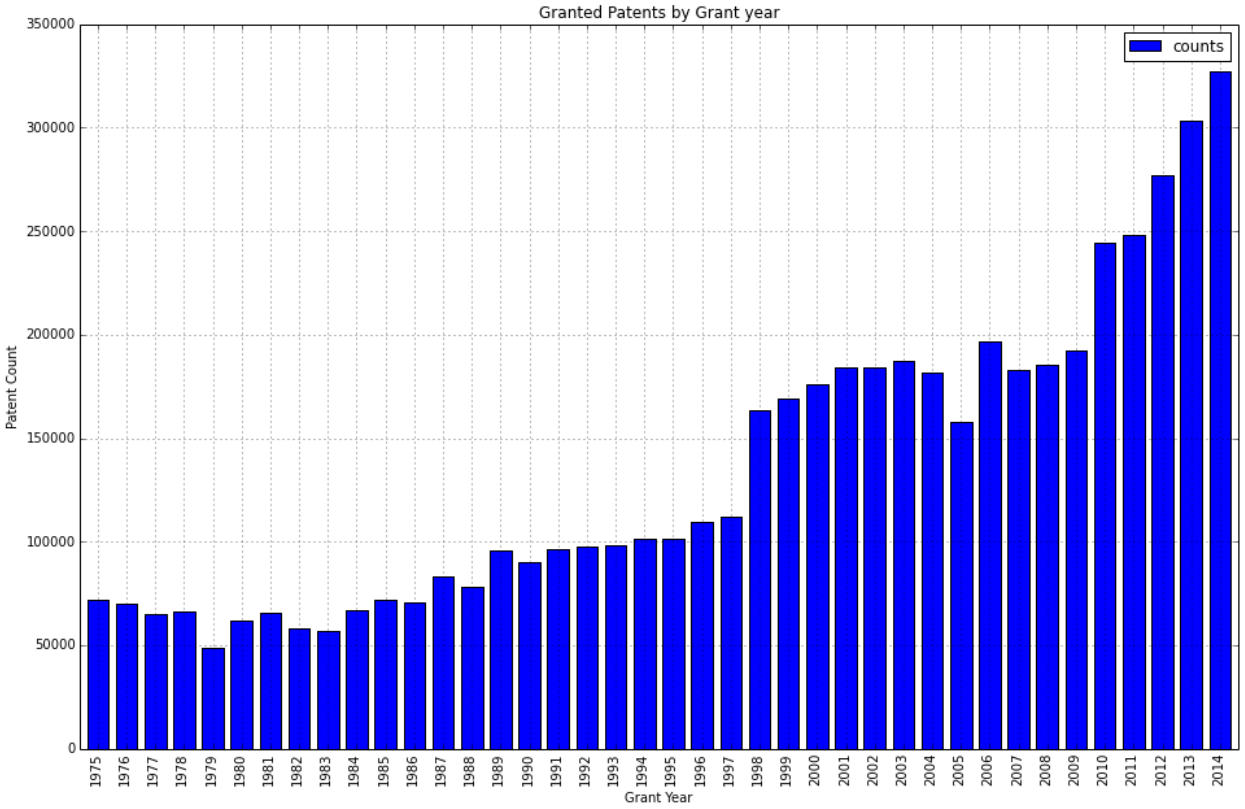

Figure 6: US utility patent grants by year, from beginning of January 1975 through end of December 2014.

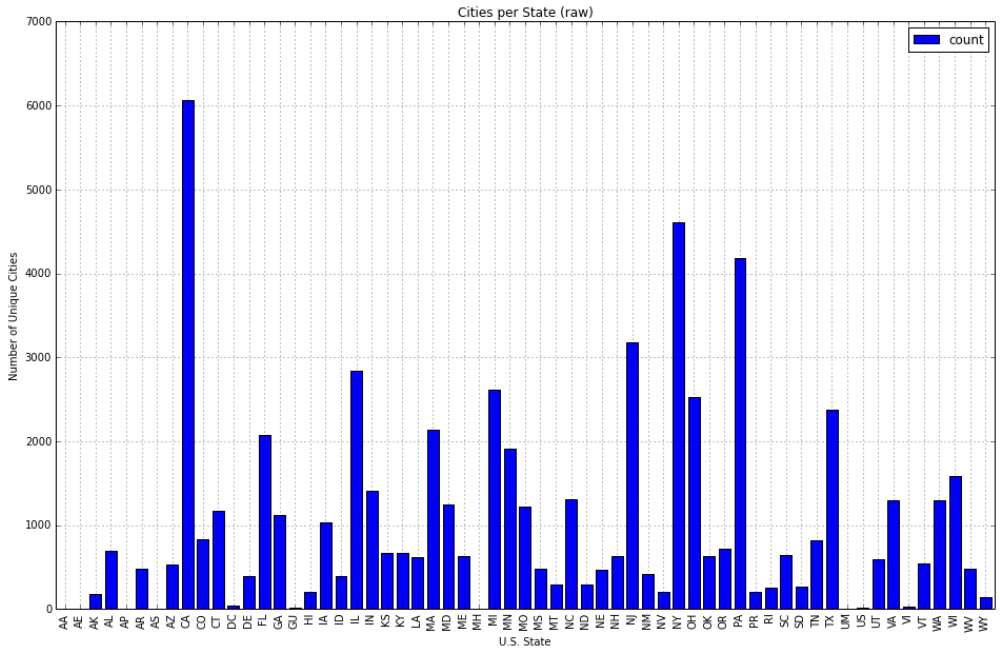

Figure 7: Number of unique raw cities by U.S. state, from beginning of January 1975 through end of December 2014.

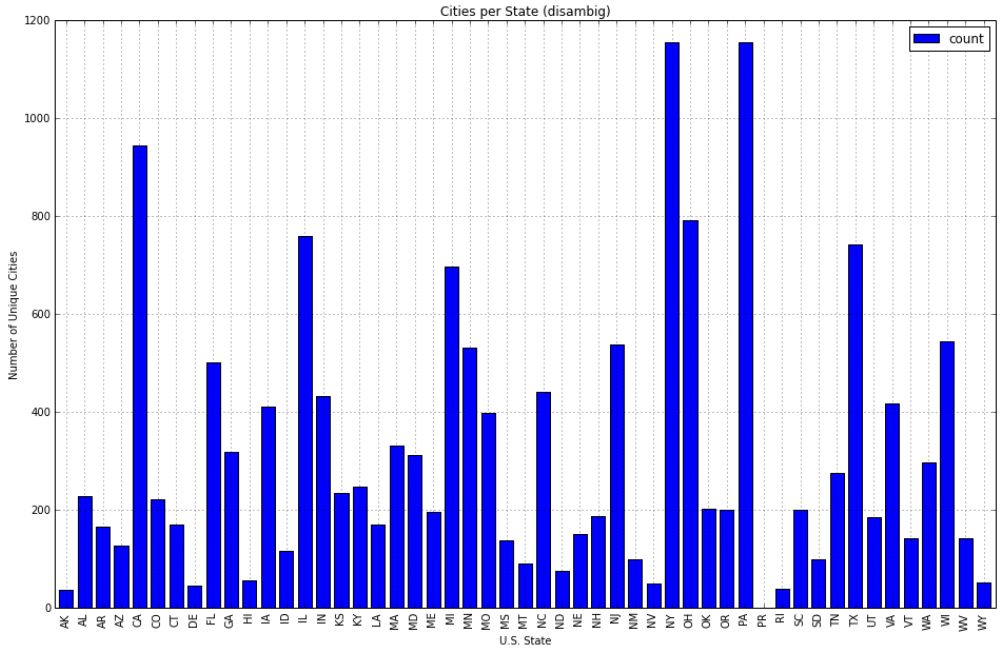

Figure 8: Number of unique disambiguated cities by U.S. state, from beginning of January 1975 through end of December 2014. Note decrease in the number of entities compared with raw data in previous figure.

A Simple Database Access Tool

Our database process makes use of the MySQL database engine to remain scalable and efficient, but to facilitate access to the data, we provide a simple, user-friendly web interface for downloading segments of data. The tool (source code available at `https://github.com/gtfierro/walkthedinosaur`) translates a simple web form into a query that is executed over the patent database. Users specify the desired segments of patent data (e.g. title, grant year, inventor name, etc) and constraints on those segments. The web tool queues the user's request, and the user is emailed a download link to a CSV/TSV (comma/tab-separated values) file when the query has completed. Figure 9 provides a technical block diagram of the process.

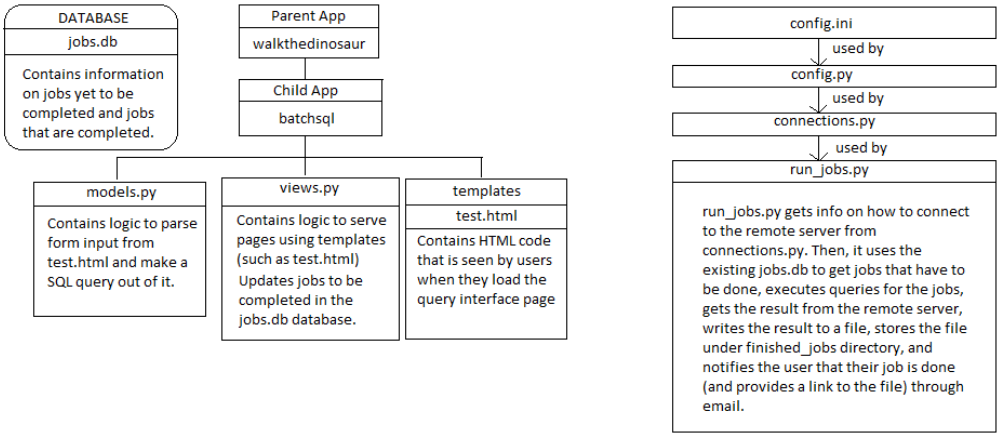

Figure 9: Block diagram of database access tools.

An example

To get the title of all the patents that had been invented in Texas between the period January 2005 and February 2005, one should select the check besides "Title of Patent" in primary information. In the filters section, set From as 2005-1-1 and To as 2005-2-1. Also, type 'TX' in inventor's state textbox. Finally, type in your email address on the bottom of the page, choose the filetype, and click on "Submit". Figure 10 illustrates the Filter interface. This form is translated into the SQL query:

```
SELECT patent.title FROM patent, rawinventor, rawlocation WHERE
(patent.date BETWEEN '2005-1-1' AND '2005-2-1')
AND (patent.id = rawinventor.patent_id)
AND ((rawlocation.state LIKE '%TX%')
AND rawlocation.id = rawinventor.rawlocation_id);
```

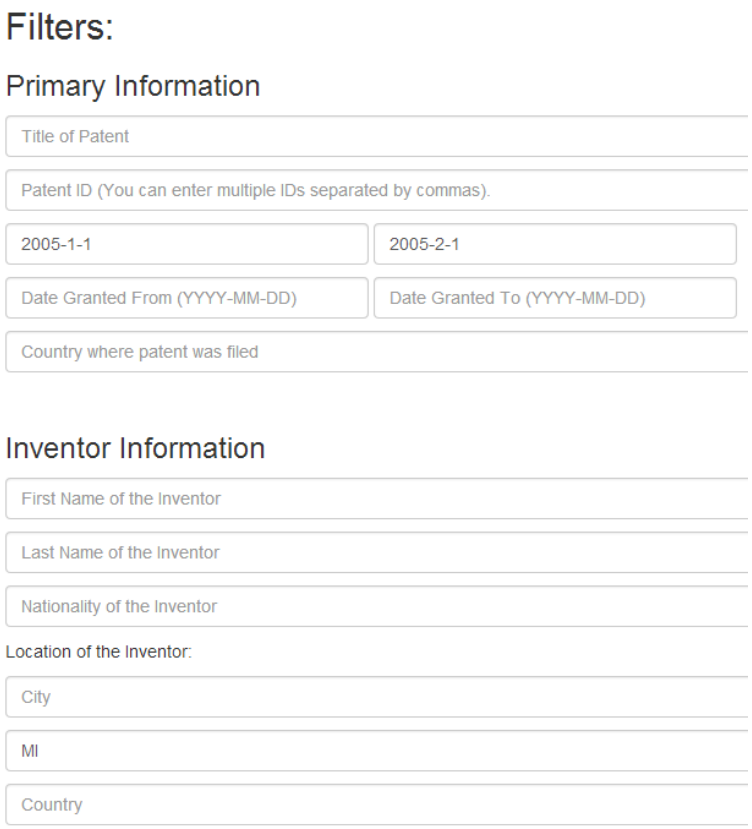

Figure 10: Filter interface with law firm

To get the names (first and last) of the law firms that prosecuted a patent in Michigan between January 2005 and February 2005, for inventors and assignees that were also in Michigan, select First Name of Law firm and Last Name of Law firm under Law firm Information (see Figure 11). In the filters section, make sure to fill in dates as before, and this time, fill in the textbox for Inventors with State with 'MI' and same for the Assignee's State textbox. Finally, just as before, fill in your email, choose your filetype, and click on "Submit"'. This form is translated into the SQL query:

```
SELECT rawlawyer.name_first, rawlawyer.name_last FROM
patent, rawlocation, rawinventor, rawassignee, rawlawyer WHERE
(patent.date BETWEEN '2005-1-1' AND '2005-2-1')
AND (patent.id = rawinventor.patent_id)
AND (rawassignee.patent_id = rawinventor.patent_id)
AND (rawlawyer.patent_id = rawinventor.patent_id)
AND ((rawlocation.state LIKE '%MI%')
AND rawlocation.id = rawinventor.rawlocation_id)
AND ((rawlocation.state LIKE '%MI%')
AND rawlocation.id = rawassignee.rawlocation_id);
```

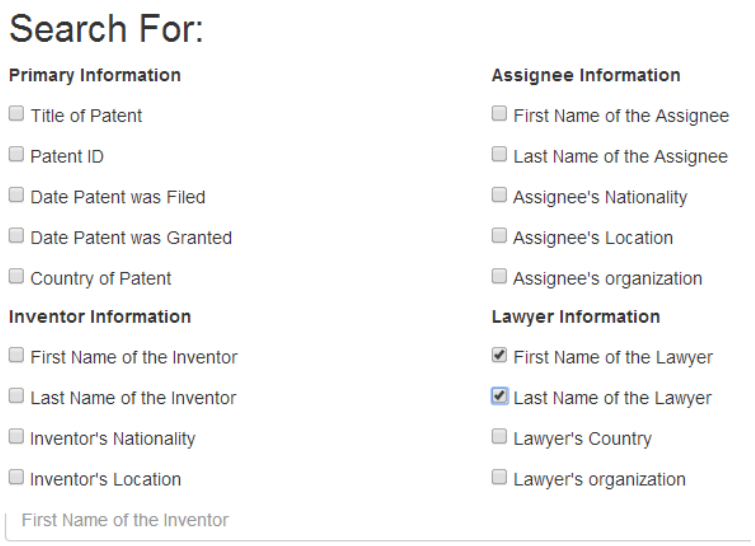

Figure 11: Filter interface

Currently, jobs take anywhere between 2 minutes and a couple hours based on how much information matches the records in the database. This is due to the fact that the database is running with limited memory, meaning multiple queries cannot run simultaneously.

A Co-Inventor Network Visualization Tool

Social networks have become increasingly popular, yet visualizing such networks requires time and technical expertise. Using the disambiguated patent database, we have developed a tool that dynamically renders the immediate co-authorship networks for any inventor(s) chosen by the user.

Starting from chosen ("seed") inventors, the tool can find all patents these seed inventors have applied for within the chosen dates. For each patent, it then creates a co-inventor link between all possible pairs of the patent's inventors. For a co-co-inventor network (defined as 2 "generations" or levels of co-authorship), it then uses this larger set of co-inventors as the "seed" inventors and cycles through again. In principle, this process can be repeated to n-generations of co-inventorship. To limit demands on bandwidth and processing, users can currently only choose one, two, or three generations. Figure 12 diagrams a process flow.

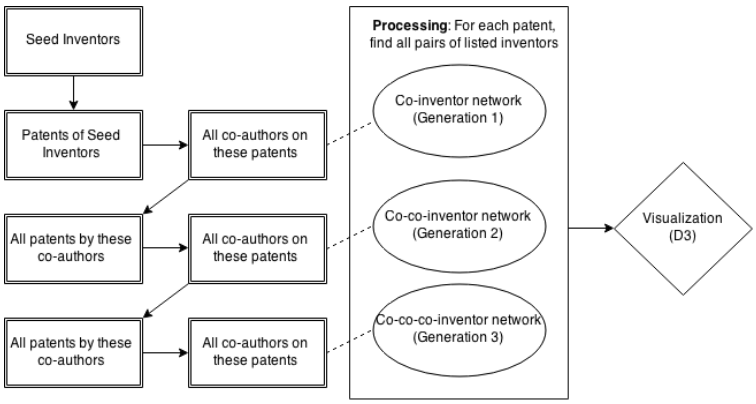

Figure 12: Flow Diagram for Network Visualization Tool

This program is driven by PHP, drawing data from the MySQL patent database and storing results in a separate, local MySQL database. After co-inventor relationships have been calculated, the end result is stored in a JSON-formatted file for visualization. The annotated PHP/MySQL code used to generate this data (with database access credentials removed) can be found at https://github.com/oreagan/socialnetwork.

Once all co-inventor relationships have been identified, the tool generates a visualization in which inventors are represented by dots that act as charged particles, repelling one another, but with co-inventors bound together. The visualization itself uses the Data-Driven Documents (D3) force layout graph, modified from the publicly available version at https://github.com/mbostock/d3/wiki/Force-Layout. This graph uses a Barnes-Hut algorithm (http://arborjs.org/docs/barnes-hut) to calculate the forces between the charges and find an equilibrium position. This process renders directly on the user's browser, typically within seconds. Particularly large networks can take up to minutes to generate and render.

Another search option will find all inventors who have patented within a technology subclass in the chosen time window, then expand out n-generations from there. Figure 13 illustrates an example from the semiconductor industry, with two additional co-authorship relationships, for patent sub-class 438/283 from 1998-2000.

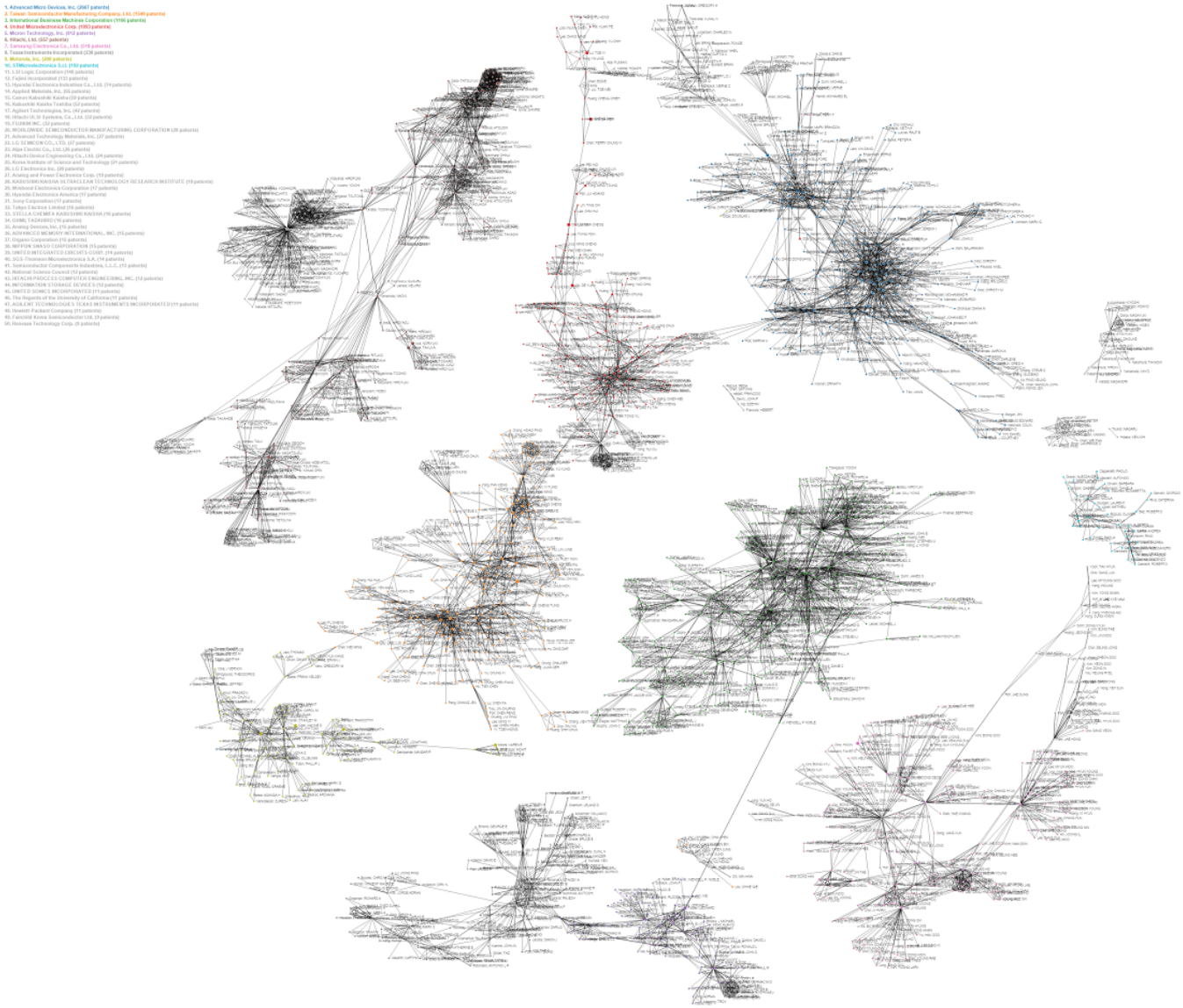

Figure 13: USPTO Semiconductor Patents in Sublass 438/283 from 1998 to 2000, with 2 additional levels of co-authorship included.

Potential improvements

The methods presented here are simple, and while less accurate than some prior approaches, provide an easier platform for improvement.

Currently, XML data are parsed from 2010 onward; before then, we rely on the Harvard Dataverse data (Li et al. 2014). The data ideally should be parsed from the raw XML for all years. All locations for the assignee and all inventors should be presented, rather than

just the first inventor. The social network illustration tool could be improved to reflect failed applications as well as successful patents. Ideally, this would be rendered such that the two networks are laid out similarly, allowing easy identification of the failed inventions and relationships.

To improve the inventor accuracy, one could introduce additional attributes, including law firms, non-patent and patent prior art, and all technology subclasses. The last has typically been done with patent classes (see Li et al. 2014). Technology classes evolve, however, and get abolished or established over time. Another approach would develop a bag of words, or descriptive "tags", that would differentiate each patent from another. By assuming that the words, or tags, or certain keywords are statistically specific to inventors, we may feed patent contents as another block, to aid disambiguation. Distance measures can also be relaxed, such that close but non-exact matches can contribute to the correct clustering.

A number of new measures will be made available in the next revision of this paper. We will provide a measure for each patent of whether a new word appears for the first time in the patent lexicon, as well as how many times that word is used subsequently. This will provide an easily interpreted measure of novelty along with a measure of how successful that novelty was in future use (analogous to future prior art citations, through arguably a more accurate measure of novelty and the impact of that novelty). Finally, we will include a count of future blocking actions from the USPTO PAIR data; these data will indicate which patents blocked subsequent invention. Assumedly, these patents were more valuable and/or better written.

To improve the assignee accuracy, additional metadata and more global string comparisons might be utilized. Normalizing common organizational affixes such as "corporation" and "incorporated," and weighting frequent (normalized) affixes with the invers ratio of their occurrence will certainly increase the accuracy of the disambiguation process. Additional improvements will be possible by incorporating the CONAME files available from the USPTO, which contain harmonized names of the first assignee of patent grants through 2012, but were not available to us at the time of writing. More sophisticated approaches will take acronyms into account in order to further consolidate records. It would also be useful to train the disambiguation algorithm with the specific improvements identified by the NBER, which would tackle some of the specific issues that arise when non-identical names within two datasets have to be matched. This will also help to provide unique and commonly used firm identifiers, e.g. those used by Compustat, to all relevant patents. A related task is to compile comprehensive sets of patent portfolios across conglomerates and large organizations that operate under different names of their subsidiaries. A test download of the recently available Bureau van Dijk (BvD) database suggests that significant improvements over previous approaches might be possible.

Using application data to decrease sample bias

To date, almost all patent based research has drawn inferences from granted patents; for those studying the process of invention or the patenting process, this amounts to sampling on the dependent variable. Hence, all this work (literally thousands of papers) could be revisited. For example, work that studies the social networks of successful collaboration could look at the impact of unsuccessful collaboration (Fleming and Juda 2004). Work that models invention as a process of exploration vs. exploitation (March 1991, Balsmeier, Fleming, and Manso 2015) would benefit from considering failed applications, as this work predicts that exploration should increase both breakthroughs and failed inventions (if the theory is correct, then exploration should result in more granted patents with 0 citation and failed applications). Continuing with the theme of extending the observed distribution of attempted invention, it would also be interesting to see if networks more prone to failure were more similar to networks that produced uncited or breakthrough patents; in other words, did the network fail because the inventors were not taking enough risk in order to cross the threshold of being granted a patent, or too much risk (that also might have resulted in a breakthrough). A great deal of recent work has focused on the application process at the USPTO (Carley, Hedge, Marco 2014); work in this genre could benefit from more complete data on the failed applications.

Conclusion

Automated updates and disambiguations of U.S. patent data would greatly facilitate fundamental and policy research into innovation, innovation policy, and technology strategy. We provide an initial set of fully automated tools aimed at accomplishing this goal for patents from 1975 through the end of 2014. Some of the tools are crude but provide a platform for future improvement. These tools provide inventor, original assignee, law firm, and geographical location disambiguations. Application data for all these fields will also be available and all data will be accessible with a public interface that emails requested data. The work also provided a tool that dynamically renders patent inventor networks, a user interface for generating .csv files, and a distance measure between classes.

Ideally, a sponsor will be found, such that these data can be curated and updated on a weekly basis.

References

[1] Balsmeier, B., and L. Fleming, G. Manso. *Independent Boards and Innovation.* Working paper, `http://www.funginstitute.berkeley.edu/sites/default/files/bfm20150505%20%281%29.pdf`

[2] Carayol N., and Cassi L., 2009. *Who's Who in Patents. A Bayesian approach.* Cahiers du GREThA 2009-07.

[3] Carley, M. and D. Hedge, A. Marco. *What is the Probability of Receiving a US Patent?.* Harvard Business Review 82: 6.

[4] Fleming, L. and A. Juda, 2004. *A Network of Invention.* USPTO Economics Working Paper No. 2013-2.

[5] Hall, B. H., A. B. Jaffe, and M. Trajtenberg, 2001. *The NBER patent Citations Data File: Lessons Insights and Methodological Tools*, NBER Working Paper.

[6] Hall, B. H., D. Harhoff, 2012. *Recent research on the economics of patents*, NBER Working Paper 17773.

[7] Herzog, T., F. Scheuren and W. Winkler, 2007. *Data Quality and Record Linkage Techniques.* New York, Springer Press.

[8] Johnson, K. 2013. *Inferring location data from United States Patents* Fung Institute Technical note.

[9] Johnson, K. 2013. *USPTO Geocoding* `https://github.com/Vadskye/uspto_geocoding` GitHub.

[10] Lai, R. and A. D'Amour; L. Fleming, 2009, "The careers and co-authorship networks of U.S. patent-holders, since 1975", `https://dataverse.harvard.edu/dataset.xhtml?persistentId=hdl:1902.1/15705`.

[11] Li, G. and Lai, R. and A. D'Amour, D. Doolin, Y. Sun, V. Torvik, A. Yu, and L. Fleming. *Disambiguation and co-authorship networks of the U.S. Patent Inventor Database, 1975-2010,* Research Policy 43 (2014) 941-955.

[12] Pezzoni, M. and F. Lissoni, G. Tarasconi, 2012. *How To Kill Inventors: Testing The Massacrator Algorithm For Inventor Disambiguation.* Cahiers du GREThA n 2012-29. `http://ideas.repec.org/p/grt/wpegrt/2012-29.html`.

[13] Raffo, J. and S. Lhuillery, 2009. *How to play the "Names Game": Patent retrieval comparing different heuristics.* Research Policy 38, 1617-1627.

[14] Trajtenberg, M., G. Shiff, and R. Melamed, 2006. *The Names Game: Harnessing Inventors Patent Data for Economic Research.* NBER.