Machine learning and natural language processing on the patent corpus: data, tools, and new measures*

Benjamin Balsmeier[****], Guan-Cheng Li[*], Tyler Chesebro[***], Guangzheng Zang[***], Gabe Fierro[***], Kevin Johnson[***], Aditya Kaulagi[***], Sonja Lück[**], Doug O'Reagan[*], Bill Yeh[***], and Lee Fleming[*]

[*]UC Berkeley, Coleman Fung Institute for Engineering Leadership
[**]University of Paderborn, Germany
[***]UC Berkeley, Electrical Engineering and Computer Science
[****]ETH, Switzerland

November 20, 2016

Abstract

Drawing upon recent advances in machine learning and natural language processing, we introduce new tools that automatically ingest, parse, disambiguate and build an updated database using United States patent data. The tools disambiguate inventor, assignee, and location names mentioned on each granted US patent from 1976 and applications from 2001 to the end of May 2016. We describe data flow, algorithms, user interfaces, descriptive statistics, and a novelty measure based on the first appearance of a word in the patent corpus. We illustrate an automated co-authorship network mapping tool and visualize trends in patenting over the last 40 years. 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, and location) are only listed by their names on each patent document and not always receive a unique identifier from the patent office (at worse they 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 of the lack of unique identifiers, individual researchers spend significant amounts of time and resources on labor-intensive manual disambiguations of relatively small numbers of patents. A few of these researchers (laudably) make these efforts available to the community, which results in a patchwork of retrospective coverages of different periods, types of data, and manual cleaning methods.

The problem of disambiguating inventor names has received considerable attention (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 often labor-intensive process (for the most recent effort, please see Kogan et al. forthcoming). 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.

The intent of this paper is to provide working prototypes for automating the disambiguation of patent entities and patent data manipulation, with the ultimate goal of providing tools and reasonably accurate and timely disambiguation data. The tools and data presented here are far from perfect and have not been fully characterized for their many potential applications; adopting an open innovation model, the thrust of this work is to put forth prototypes of automated disambiguation and data manipulation, with the hope that it greatly decreases manual cleaning and motivates further automation. The work provides a simple user interface that emails data to users. Of interest to network scholars, it also provides an automated co-inventor network tool that renders in real time, based on seed inventors or a specification of a particular technology classification. For updates and addresses to authors, data, tools and code, please see `http://www.funginstitute.berkeley.edu/`.

This paper also presents novel data that opens up two attractive areas of investigation.

First, it disambiguates application data; almost all research to date has used granted patent data, which ignores the many applications that were denied, and may have introduced bias in the many patent papers published to date. 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. These data will enable the ambitious researcher to question all the patent literature to date that has relied upon granted patents. We also provide a measure of novelty, based on the first occurrence of a word in the patent corpus, since 1975. This measure enables a prospective measure of a patent's novelty that remains orthogonal to the typically used measure of citations (which have also been used to measure very different concepts such as impact, financial value, or knowledge diffusion). Code is available from the last author to any non-profit research organization.

Data Sources, Parsing, and Databases

Rather than parse weekly data available from the USPTO (in formats that have varied over time), we scraped every granted patent and application from the USPTO website and parsed, cleaned and inserted the data into a SQL database. Figure 1 illustrates the process.

As the data are extracted from USPTO documents, they are streamed into a relational database that contains records linking patents, citations, inventors, 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.

The input to the parsing algorithms is a stack of HTML files downloaded from the USPTO website[1]. The output of the parsing algorithm is a Tab-Separated-Values (TSV) file, with each row corresponding to a particular patent parsed. Getting from raw HTML to TSV requires five steps.

[1]The URL is http://patft.uspto.gov/netacgi/nph-Parser?Sect2=PTO1&Sect2=HITOFF&p=1&u=/netahtml/PTO/searchbool.html&r=1&f=G&l=50&d=PALL&RefSrch=yes&Query=PN/XXXXXXX, where "XXXXXXX" is the 7-digit ID of the target patent.

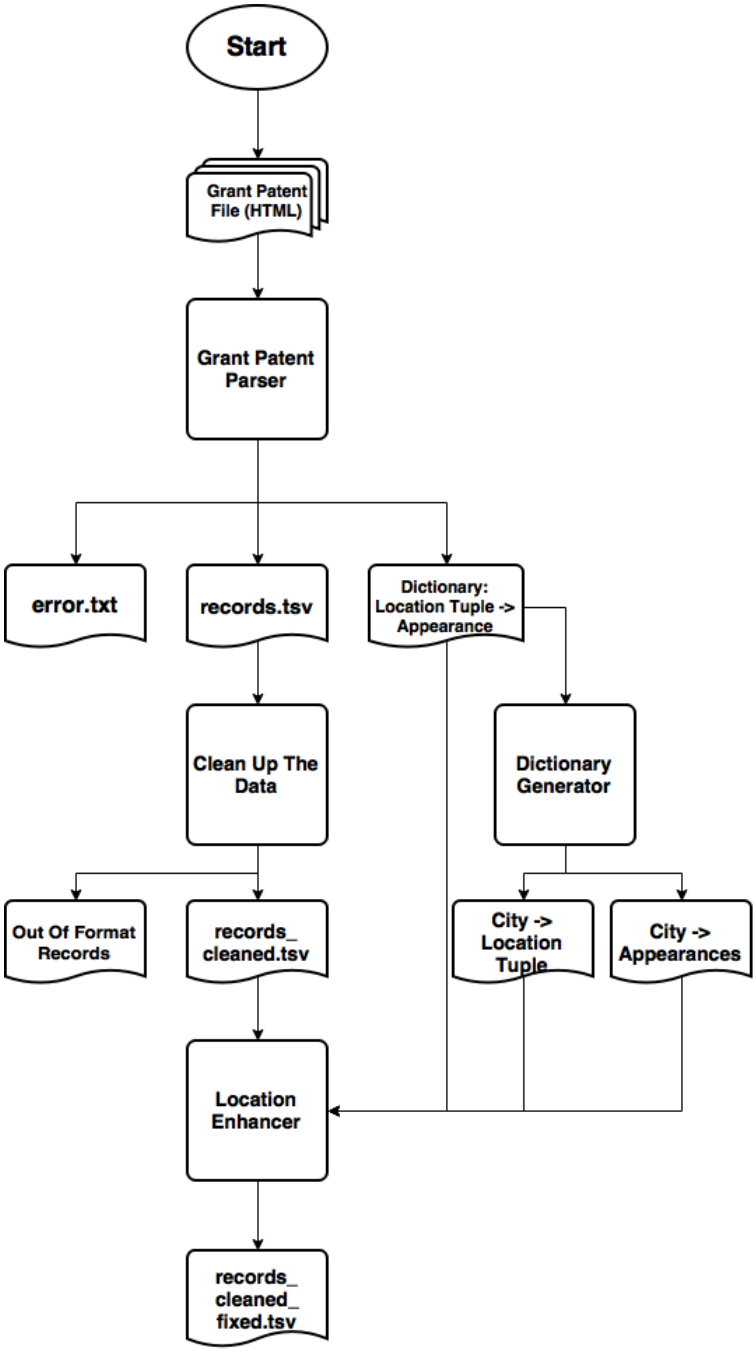

Figure 1: Tool and data flow for scraping and parsing USPTO website data. A similar process extracts application data.

Step 1: Vectorization

Step one of the parsing algorithm iterates through the list of HTML files, and extracts useful data from them. A python module called "the beautiful soup" is used to convert the raw

HTML text into a well-maintained python object, making it easier to extract text strings. The data is extracted into 23 fields, including grant ID/date, application ID/date, assignee names/locations, inventor names/locations, US Class, CPC Class, referred US patents, etc. If there is error extracting data from a particular patent, the program would record that patent's ID in error.txt, and skip it to continue to next patent. Moreover, in the process of extracting locations (for both assignees and inventors), the program maintains a dictionary file to keep track of the total number of appearances for each location tuple, which is defined as a "(CITY, STATE, COUNTRY)" tuple. This (location from tuple to appearances) dictionary will later be used for enhancing location accuracy.

Step 2: Soup to records

The second step is designed to clean up the parsed records.tsv from step one. This process is to ensure that parsed data is in a reasonable format so that each row at least "looks like" a correctly parsed patent. Specifically, we checked:

1. For each parsed patent (row), whether the total number of fields is 23
2. For each parsed patent (row), whether the last field is "1" (indicating that this is a granted patent)
3. For each parsed patent (row), whether any of the field contains a newline character
4. For each parsed patent (row), whether the second last field is "0", "1", or, "WITHDRAWN" (this field indicates whether there is an error flag ("*") near the grant date)
5. Whether there are repetitions of patents parsed

The output of Step two contains two files: records_cleaned.tsv (which stores the cleaned lines), and records_outOfFormat.tsv (which records all out-of-format lines). So far most fields in records_cleaned.tsv should be accurate, except for the locations of assignees and inventors. For example, in patent 8449912, the assignee location is (Bandra-Kurla Complex, Bandra East, Mumbai, IN) which contains sub-city level locations. This is abnormal because in most assignee/inventor locations, only the city and country level locations appear. Therefore, the parser program in Step one would parse (BandraKurla Complex, Bandra East, Mumbai) into the city field. This creates consistent mismatches and deserves attention and fixing in future work.

Step 3: Geographical dictionary construction

Before starting to enhance the locations, Step three of the algorithm constructs a dictionary: (from city to appearances), which will be useful in identifying the correct city name among sub-city locations. To construct this dictionary, the program iterates through the (from location tuple to appearances) dictionary generate by Step one, identifies the city name in the location tuple, and adds the tuple appearance count to the city appearance count.

Step 4: Geographical location cleaning

Step four will enhance the locations in records_cleaned.tsv with the help of the (from city to appearances) dictionary. The program goes through records_cleaned.tsv; for each patent,

it grabs the city-level location, and splits text string around if able. Then, the program finds the substring that has the highest appearances based on the (from city to appearances) dictionary, and changes the corresponding locations in the parsed data fields. The final output is records_cleaned_fixed.tsv.

Step 5: Validation

In order to verify the accuracy of the parsed data in records_cleaned_fixed, we randomly selected 30 utility patents. For each of the selected patents, we compared all relevant data fields parsed from the HTML file against its original PDF file. To find the PDF file, we first went to www.google.com/patents/USXXXXXXX, where "XXXXXXX" is the 7-digit ID of the target patent. Then, we downloaded the PDF file by clicking a "View PDF" button at the upper right corner of the page.

Having both the PDF and the parsed data available, we compared each of the data fields and see if they matched each other. Table 1 illustrates the specific data fields we compared and the result:

Data Fields Compared	Results
Patent Grant No.	30/30: match
Grant Date	30/30: match
Application No.	10/30: match; 20/30: convention difference, first two digits are missing in the PDF file
Application Date	29/30: match; 1/30: in PDF, Application Filed Date is seemed to be replaced by PCT Filed Date
Assignee Name/Location	30/30: match
Inventor Name/Location	29/30: match; 1/30: inventor name mismatch (in PDF: Günter, Bäumgen; in parsed: G/u/ nter, B/a/ umgen)
Referred US Patents	30/30: match
US Class	25/30: match; 5/30: convention difference, US Class updated
CPC Class	30/30: convention difference, CPC Class not appearing in PDF

Table 1: Validation Results

For each instance of mismatch (marked in red) shown above, we checked the HTML file. We found out that the mismatches all resulted from the inconsistence between the HTML file and the PDF file. Starting from the USPTO website, the parsing appears accurate for these 30 patents.

Algorithms

Inventors[2]

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; as a result, the inventor disambiguation runs last.

Step 1: Vectorization

We model a patent as an unordered collection of that patent's attributes. The current version attributes are described below:

- Inventor name
- Co-authors (implicitly, by association on the same patent)
- The (disambiguated) assignees
- Any overlap in CPC technology classes
- The (disambiguated) city, state, and country location.

The occurrence of each attribute is used as a feature (i.e., independent variable) for training a classifier (the machine learning term for a tool that labels a record as a particular entity). We build a document-by-attribute incidence matrix, which is extremely sparse because a patent cannot use all possible attributes.

Figure 2 illustrates the matrix and sub-matrices conceptually. All the approximately 6M patents are arrayed horizontally. Each block arrays the different attributes as columns. For example, each inventor name has a column (such as John Davis), each assignee such as IBM (inventor disambiguation is done after assignee disambiguation), each CPC class, and each geographical location. The presence of an attribute is indicated with a "1" in that matrix entry. Using a discrete indicator may appear crude, however, it is a conventional approach that is computationally efficient and favored with large datasets.

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

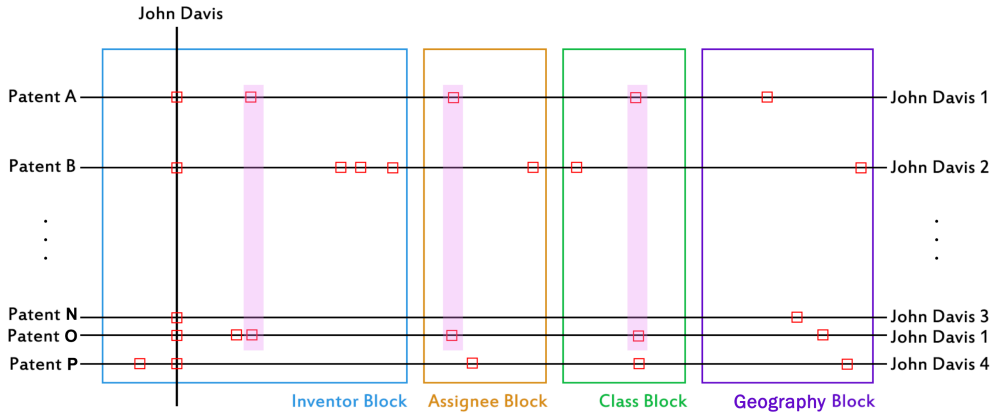

Figure 2: Document (patent) by attribute (name, assignee, class, geography) incidence matrix.

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 line in Figure 2 (essentially blocking on the exact name "John Davis", where blocking is a machine learning term for temporarily restricting consideration to a subset of records, in this case, the five records with an inventor named John Davis). We look closer at the five rows that have 1s along that column, namely the index of the patents being filed by the inventor named 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: Bottom up K-means merging

Common names usually represent multiple persons, e.g., John Davis. If, in our 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 place 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 works iteratively to place an observation into the cluster with the most similar mean. It stops this process when the objective function described below reaches its optimum.

Step 3: Lumping name aliases

Once the k-means bottom up merging completes, we assign the most central observation of each cluster to be the unique identifier and augment the matrix column-wise with these unique identifiers, as depicted in Figure 3. In other words, we uniquely label each cluster believed to be a different John Davis and affix this label to each column. This results in columns being split, and hence the number of columns in the working matrix increases, as separate inventors are identified. Compare Figures 2 and 3, where Figure 2 has one column marked John Davis and Figure 3 has four columns.

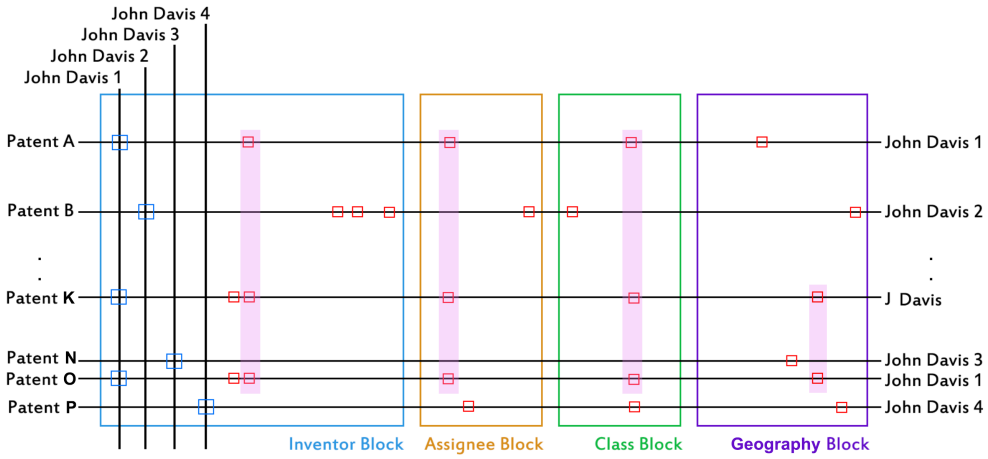

Figure 3: Augmentation with unique identifiers: as individuals are identified, additional columns are split out.

Following the k-means algorithm, we add an additional step that attempts to merge incorrectly split inventors. The typical case occurs when there is matching upon the non-name attributes (co-inventor, assignee, class, geography) and yet the name fields (first name, middle initial, last name) do not match. This step is designed to merge 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).

There are three cases where merging occurs if the k-means algorithm determines high similarity except for the name fields: 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. In essence, if there is high non-name matching (co-inventor, assignee, class, geography), there needs to be a match of at least one of the three name fields.

Summary and some technical details

The goal of the K-means algorithm is to produce high intra-cluster and low inter-cluster similarity between inventors. The objective function to be maximized is expressed in Equation (1) as:

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

$\phi^{(Q)}$ represents the cluster quality; the higher the score, the better. 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. Taking each term in turn, the first term in the numerator normalizes for the size of the cluster, the second sums the distances between clusters, and the denominator sums the distances within clusters.

Figure 4 illustrates how the algorithm decides when to stop splitting clusters for the toy example. In this case, there appears to be little if any additional benefit of splitting into fewer than four clusters. When the sum no longer decreases (within some very small amount), there is almost no benefit to continuing to split clusters, and an increasing risk of splitting the same person into different clusters.

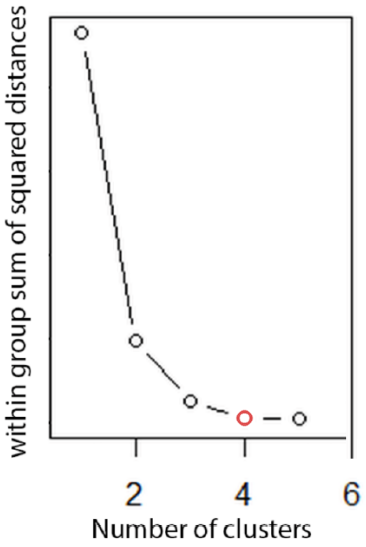

Figure 4: The optimal number of clusters is determined by the point of decreasing returns (4 in this example).

In summary, the disambiguator considers each inventor's co-inventors, geographical location, overlap in CPC technology class, 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. Current work plans to incorporate new data fields.

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 was compared in same manner as Li et al. 2014, with the exception that patents after May of 2014 are not assessed. Comparison files are at `http://funglab.berkeley.edu/guanchengli/disambig.golden.list.txt`.

Asian names are particularly challenging, in that first and last names often match, city and state data are often lacking, and much invention occurs within large firms. This makes it nigh impossible for any algorithm, including manual, to determine unique inventors. For example, Kim Lee is a very common Korean name, Samsung accounts for a large proportion of Korean patents, and the only location data is often the country of Korea. The disambiguation must rely on technology and co-authors only and understandably, sometimes fails.

We strongly recommend that all users sample their data and assess its accuracy for their research question. We provide metrics where possible that flag our doubts about any particular data point and encourage users to check such points most carefully. While we strive to provide the most accurate disambiguation possible, the user should conceptualize the data in its current state as a starting point (and help the community progress towards fully automated and blissfully accurate disambiguations).

Assignees

The assignee records are used to determine the ownership of patents at the point the patent was initially applied for. This is 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).

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 creative interpretations of General Electric demonstrates the most basic challenge of getting past typos and misspellings of the same intended entity. The unaltered records differ in structure from the inventor records because the majority of assignee records are organizations rather than individuals. Entity resolution of personal names struggle most to differentiate 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.

We treat assignee disambiguation as a clustering problem as well, but with a different set of approaches than that of the Inventors. Assignee disambiguation poses some unique challenges. The amount of patents per assignee is skewed; a very small number of assignees, for instance IBM, hold thousands of patents, while many assignees are small organizations that hold less than ten patents. A second challenge is the many spellings of a single organization, as illustrated above.

We start with the dataset of disambiguated assignees generated by the National Bureau of Economic Research, herein referred to as the "NBER data". This contains over 2 million semi-automated disambiguated patents from the US Patent Office with grant dates from 1976 through 2006. With the assumption that the NBER data is accurate, we use it to build a dictionary of all instantiations of the assignee name (basically, a complete list for each firm similar to the sample for General Electric above). We also use the NBER data to build a feature space or description of each patent that contains a 1) location in country, state, city format 2) First and Last name of inventors 3) list of other patents cited and 4) Cooperative Patent Classification (CPC).

Disambiguation starts by examining each patent after the 2006 NBER data individually. If a patent's assignee is found in the dictionary (in other words, there is an exact lexical match of the assignee name with any entry in the assignee dictionary), the patent is linked to that assignee. If none is found, we attempt to find a match by looking at the patent's k-nearest neighbors (where k=5 in this case). If no assignee within the k-nearest neighbors passes the similarity threshold, the assignee is considered to be a new label (that is, an assignee not present in the data before 2006). Clustering is performed on these remaining unassigned patents separately, as they are assumed to be (most often recently founded) firms which patented for the first time after 2006.[3] We describe the details of each of these steps below.

Step 1: Dictionary build

First from the NBER data, we build a look-up dictionary (technically, a hash table) that links all abbreviated and raw names within the NBER disambiguated patents to the unique

[3]The post 2006 clustering of the approximately 300,000 patents is still in development and expected to be posted in January of 2017.

identifier created by the NBER pdpass (patent database project assignee). Because the table includes both the standardized name given by the NBER and the raw name from the original patent, we are able to account for many common misspellings and abbreviations of each assignee already existing in the dataset. The look-up dictionary aims to account for all abbreviations and misspelling of an existing assignee that have occurred in patents within the NBER database, between the years of 1976 and 2006. This creates quick matching during the first phase of the disambiguation process for representations of an assignee that have already been seen.

Second, we consider each assignee on a per patent per assignee basis. For each patent, the attributes noted in the above paragraph are recorded as binary vectors (1 where they occur and 0 where they do not) so that a binary matrix is built from the NBER data with one line per patent per assignee. This is the data matrix that we use to cluster the nearest neighbors of new assignees.

Step 2: Dictionary Match

New assignees names are first standardized and queried for in the lookup dictionary constructed from the NBER data. If an exact match is found, the assignee is grouped with the assignee of its match and the algorithm moves on. We find that approximately 74% of the assignee names are already contained within our look-up dictionary, for the post 2006 time period.

Step 3: K Nearest Neighbors Classifier

If no match is found in the lookup dictionary, we then use an attribute matrix to find the k-nearest neighbors to a patent. The features of each assignee are binary, meaning they cannot exist along a gradient scale; each feature is either present or absent (0 or 1). The best way to represent an assignee is as a vector in N dimensional space. In our algorithm, the binary feature space contains about five million different observations composed of the possible entries for location, inventor, classification, and citation. We use cosine similarity because it measures the cosine of the angle between each vector; the more features two vectors share, the smaller the angles between them and the greater the cosine similarity. Cosine similarity was chosen over other distance metrics because it is more effective when comparing categorical or binary features than a more traditional distance metric like Euclidean distance.

This algorithm uses cosine distance to find the closest five assignees in the NBER dataset to the current patent based on the attributes specified in the above section. The assignee's name is compared to that of the 5 closest neighbors, meaning the 5 assignees with the closest matches in locations, inventor names, citations, and classifications. The comparison is done through Levenshtein string distance, a measurement of the number of single-character edits (i.e. deletions, substitutions, or insertions) that are required to change the first string into the second, and any two names that are found to have a 75% similarity or greater are lumped together. If no match is found then the assignee is assumed to be a new company and is assigned a new label.

The combination of k nearest neighbors and string matching has several advantages over conventional assignee matches of simple pairwise comparison of all assignees. First, it is much less computationally expensive as only 5 string comparisons per assignee are made rather than n comparisons, where n is the size of the entire data set. Second, this helps reduce false positives by only comparing assignees that have a high probability through comparison of attributes. Third, assignees are matched to already existing clusters of names from the NBER dictionary, which include many common variations, abbreviations, and misspellings of company names. This method is especially effective at finding acronyms (e.g. "IBM" for "International Business Machines") and variations in corporate affixes ("Corporation" vs "Corp" vs nothing at all).

The process ends when the assignees of all patents are either linked to an existing assignee identifier, or are determined to be an assignee that is not present in the NBER database. The disambiguated results are organized into a table with one entry per patent, per assignee with the following features:

- Patent: the patent number
- OriginalAsgName: the "raw" assignee name
- MatchedAsg: This is the assignee name it was matched with. If a match was not found this is the raw assignee capitalized and punctuation removed for standardization
- MatchCertainty: The degree of certainty for the match. 100 means dictionary match, -1 means the assignee found no match and is a new label
- MatchMethod: how the match was determined
- AsgIdent: unique assignee identifier. If 8 digits, this is the pdpass identifier from the NBER disambiguation. If 9 digits, this is our unique identifier
- Matches: The specific features that each assignee matched with the assignee name being clustered.

Step 4: Common Name Voting

After disambiguation occurs, linked assignees are clustered into a single Assignee Object that contains each raw name and patent occurrence of the assignee. The occurrence of each variation of an assignee's name is counted and the one with the greatest counts is label as the standard name used for the Assignee for this paper. Also listed are the first and last assignee names by patent grant date.

Feature selection

The assignee matching features were further divided as the following:

- Location represented as Assignee country, state, and city.
- Inventor: Inventor first name with middle initial, Inventor last name
- CPC classification: Main class, Sub Class
- Citation: All patents cited

To standardize names we followed the same procedure as the NBER PDP process[4].

Step 5: Validation

We randomly selected 30 patents from the post 2006 data that were not included within the NBER dataset and investigated their matches by hand. Of the 30 tested, 27 were dictionary matches and 3 were considered new assignees. From inspection, it appears that only one of the new assignees was incorrectly assigned "Verigy (Singapore) Pte. Ltd." because of the parenthetical city inside of the name.

MatchAsgName	OriginalAsgName	MatchMethod	Patent	Check
INTEL CORPORATION	Intel Corporation	dictionary	8560484	correct
QUALCOMM INCORPORATED	QUALCOMM Incorporated	dictionary	8862113	correct
THE PROCTER & GAMBLE COMPANY	The Procter & Gamble Company	dictionary	7402554	correct
LG ELECTRONICS INC	LG Electronics Inc.	dictionary	8626437	correct
CANON KABUSHIKI KAISHA	Canon Kabushiki Kaisha	dictionary	7549734	correct
CONTINENTAL AUTOMOTIVE GMBH	Continental Automotive GmbH	new_asgn	8541959	correct
RESEARCH IN MOTION LIMITED	Research In Motion Limited	dictionary	8175275	correct
KABUSHIKI KAISHA TOSHIBA	Kabushiki Kaisha Toshiba	dictionary	9007846	correct
SYNOPSYS INC	Synopsys, Inc.	dictionary	8091055	correct
SAMSUNG ELECTRONICS CO LTD	Samsung Electronics Co., Ltd.	dictionary	7190939	correct
HITACHI KOKUSAI ELECTRIC INC	Hitachi Kokusai Electric Inc.	dictionary	8535479	correct
RED HAT INC	Red Hat, Inc.	dictionary	9060274	correct
MICROSOFT CORPORATION	Microsoft Corporation	dictionary	7605804	correct
LECO CORPORATION	Leco Corporation	dictionary	7851748	correct
ILLINOIS TOOL WORKS INC	Illnois Tool Works Inc.	dictionary	8551562	correct
VERIGY (SINGAPORE) PTE LTD	Verigy (Singapore) Pte. Ltd.	new_asgn	7519827	incorrect
SAMSUNG ELECTRONICS CO LTD	Samsung Electronics Co., Ltd.	dictionary	8027545	correct

[4]The URL is https://sites.google.com/site/patentdataproject/Home/posts/namestandardizationroutinesuploaded

KONICA MINOLTA BUSINESS TECHNOLOGIES INC	Konica Minolta Business Technologies, Inc.	dictionary	7476481	correct
MITSUBISHI CHEMICAL CORPORATION	Mitsubishi Chemical Corporation	dictionary	7830472	correct
DEPUY PRODUCTS INC	DePuy Products, Inc.	dictionary	7993407	correct
KYOCERA CORPORATION	Kyocera Corporation	dictionary	8045829	correct
LG ELECTRONICS INC	LG Electronics Inc.	dictionary	8077572	correct
SIEMENS AKTIENGESELLSCHAFT	Siemens Aktiengesellschaft	dictionary	7194381	correct
MONTEFIORE MEDICAL CENTER	Montefiore Medical Center	dictionary	8765399	correct
OMRON CORPORATION	Omron Corporation	dictionary	7649491	correct
SUMCO TECHXIV CORPORATION	Sumco Techxiv Corporation	new Label	8251778	correct
GENERAL ELECTRIC COMPANY	General Electric Company	dictionary	7988420	correct
GM GLOBAL TECHNOLOGY OPERATIONS INC	GM Global Technology Operations, Inc.	dictionary	7491151	correct
MOLTEN CORPORATION	MOLTEN CORPORATION	dictionary	7621008	correct
EUV LLC	EUV LLC	dictionary	7196771	correct

Locations[5]

Geographic disambiguation has been greatly simplified by using the scraped data directly from the USPTO website. The locations have already been cleaned quite thoroughly and only need disambiguation. 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 "Île-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.

[5]A prior version of this paper expended much effort parsing and cleaning the raw USPTO data; by scraping the data directly, we have now bypassed much of that work, with improvements in accuracy. For details please see: `http://www.funginstitute.berkeley.edu/sites/default/files/GeocodingPatent.pdf`.

- **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.

Applications

Treatment of application data since 2001 is similar to that of grant data. 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).

The raw application data is ingested into a database table separate from grants. Applications which have been granted are 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, 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 (Object Relational Mapper) 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. Table 1 lists the raw and disambiguated patent grant data for each disambiguation where available. Figure 5 illustrates yearly grants from January of 1976 through May of 2016, inclusive. The figure compares a raw count of the number of patents during the time period from the USPTO search tool with the number of patents available in Fung database. Most discrepancies appear to be double counted patents, though the numbers are very close in most years. Figure 6 provides a histogram of the number of patents by each unique inventor, up to 50 patents. The distribution is highly skewed; almost 2 million inventors have only one patent, and 22,006 inventors—0.6134% of the whole sample—have more than 50 patents.

Figures 7 and 8 illustrate county level chloropleths in the United States. Figure 7 illustrates patenting per capita and indicates an outlier in Santa Clara, California, corresponding to the heart of Silicon Valley (where there are 0.061 patents per person). The list of top ten counties highlights newly emerging hubs in the west, such as San Francisco and environs, Boulder, Seattle, and Corvallis, and northern Virginia in the east. The high technology regions of the industrial age in the northeast continue to patent (though that is weakening, as illustrated in Figure 8).

Figure 8 illustrates the change in patenting between recent activity and the availability of data in 1976. Blue indicates a drop in recent patenting and includes much of the northeast, including Pittsburg (in Allegheny county), as well as a decline in Los Angeles county in

California. Raleigh (in Wake county), North Carolina and Merced, California have increased their patenting, assumedly due to high technology development in the former case and the construction of a new University of California campus in the latter.

	Number of observations in raw data	Number of observations in disambiguated data
Granted Patents	5,932,062	5,932,062
Applications	5,328,752	5,328,752
Inventors	13,774,902	3,587,666
Assignees	5,281,442	365,716
Locations	19,056,344	132,793

Table 3: Descriptions of raw and disambiguated data. Except for applications, all numbers refer to only granted patents. Location data include inventors and assignees and location is a 3 tuple of City, State, and Country.

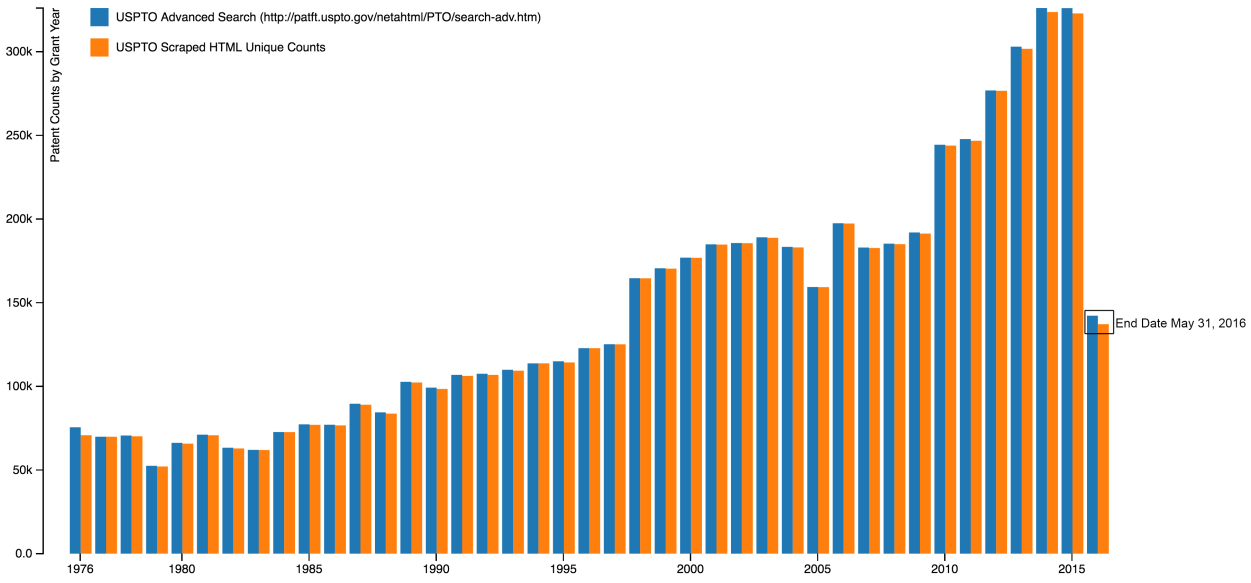

Figure 5: US utility patent grants by year, from beginning of January 1976 through end of May 2016, comparing scraped patents available vs. a raw count of patents from the USPTO system.

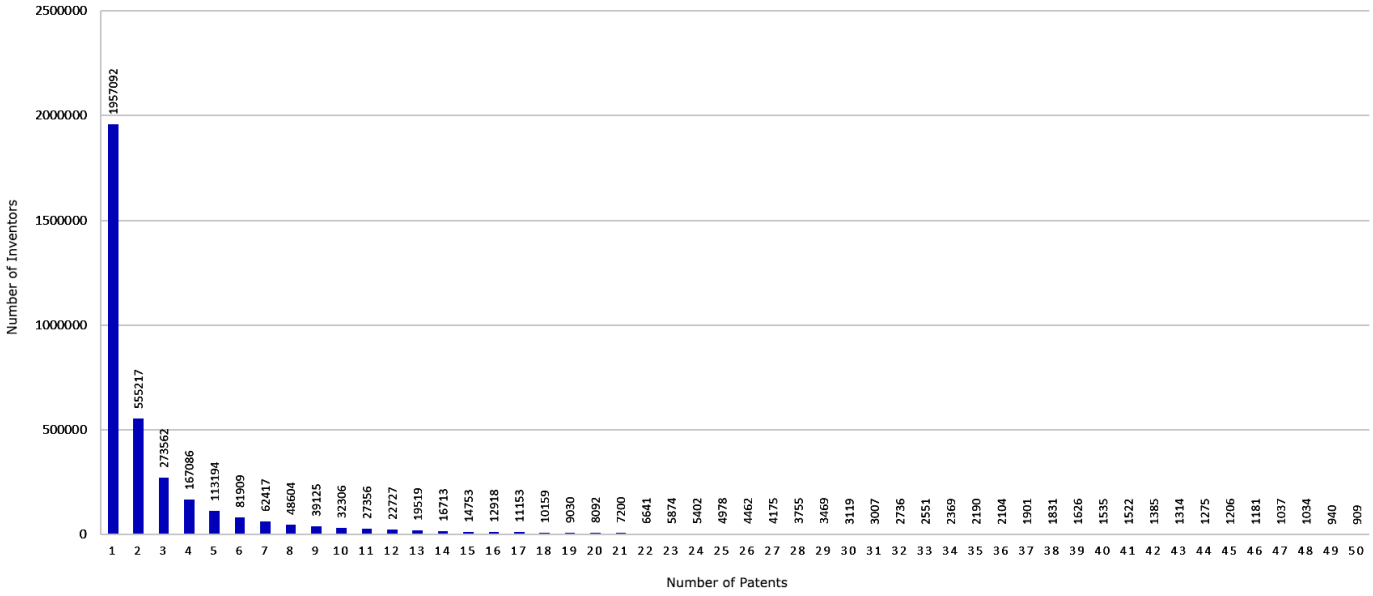

Figure 6: Histogram of number of patents by inventor. 22,006 inventors, or 0.6134% of the total, have more than 50 patents.

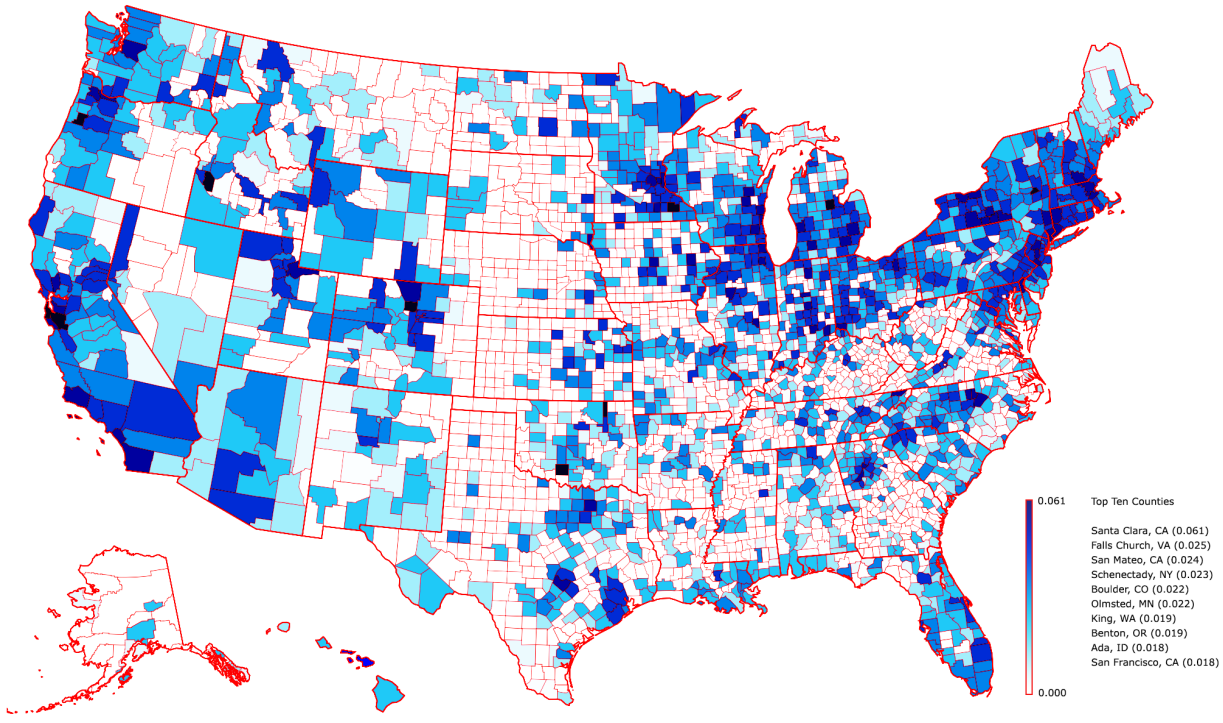

Figure 7: Patenting per capita by county, 1976-2016.

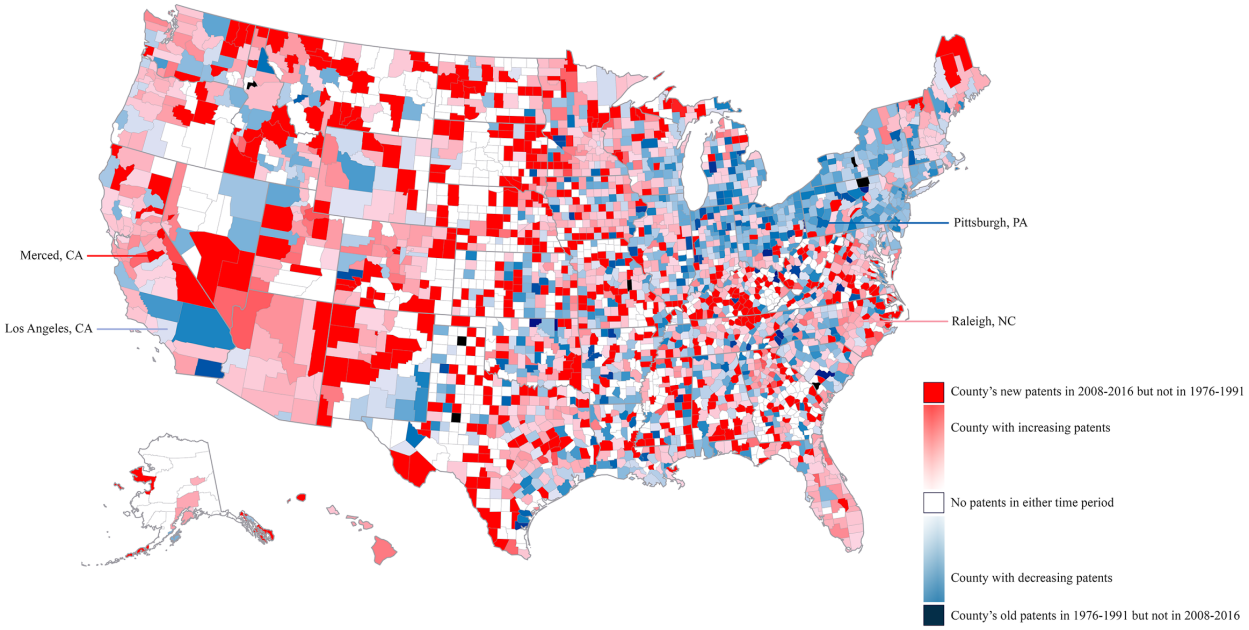

Figure 8: Change in patenting by county, comparing 1976-1991 to 2008-2016.

A Simple Database Access Tool

Note to reader: The database access tool is not available until early 2017 (it requires the SQL database to be rebuilt). All data are currently available via bulk download at: http://funglab.berkeley.edu/pub/fung-pat-data-20160531.zip.

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.

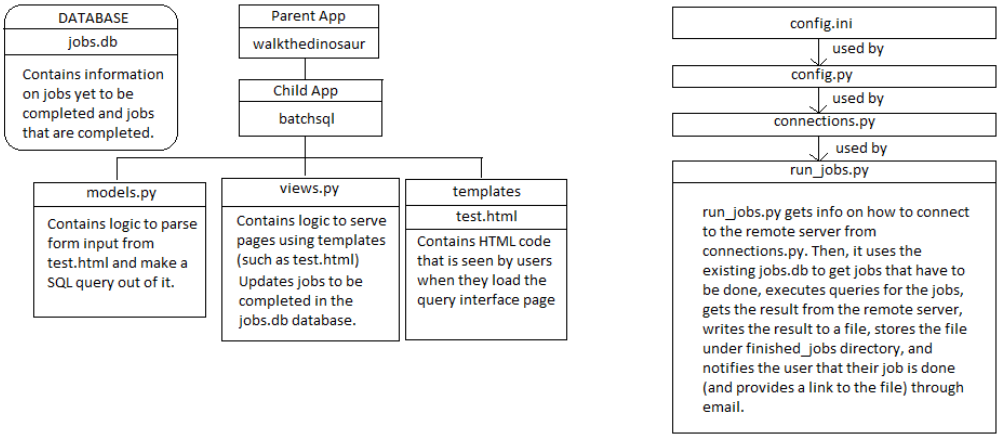

Figure 9: Block diagram of database access tools.

For 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". These entries are translated into the SQL query below. Currently, jobs take anywhere between 2 minutes and a couple hours based on the depth of the query. Multiple queries cannot run simultaneously; users will simply experience a longer wait for the email.

```
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);
```

FirstPatWord: A lexical measure of patent novelty

Scholars have long sought to explain the sources and process of creativity and innovation, in an effort to optimize the production and reap the benefits of novelty. Unfortunately, this research continues with little resolution in sight. Varied definitions and measures cause part of the problem; for example, in the patent literature, citations have been used to measure both creative search (Bernstein 2014) and financial impact (Hall and Harhoff 2012), even though highly cited patent portfolios can result from an exploitation strategy that focuses on extant firm capabilities rather than search (Balsmeier, Fleming, and Manso forthcoming).

To provide a cleaner measure of novelty, we developed a baseline dictionary of all words used in the patent literature between 1975 and 1985, and then identified the patents from 1985 to 2014 that used a new word that was not included in that dictionary. Words with

leading or ending hyphens or a double hyphen were dropped, as well as words consisting entirely of numeric characters. Even though many words could be typographic errors, further cleaning was avoided, as it is very difficult to differentiate between such errors and truly new words. This effort resulted in:

- 4,791,515 patents with an application date 1985 and 2014 (including 530,799 patents where the NBER categories are not known)
- 1,121,468 patents with at least one new word
- 3,670,047 patents with no lexical novelty
- 2,816,425 new words used from 1985-2014 that had not been previously used

To get more detailed information about the lexical novelties, we separated the patents into the six technological categories introduced by Hall, Jaffe, Trajtenberg (2001). To get the technological categories for all patents up to 2014 we linked our dataset with the USPTO Historical Master File described by Marco, Carley, Jackson, Myers (2015). The following graphs illustrate the new measure. Figure 10 illustrates the total number of new words by year and six NBER categories.

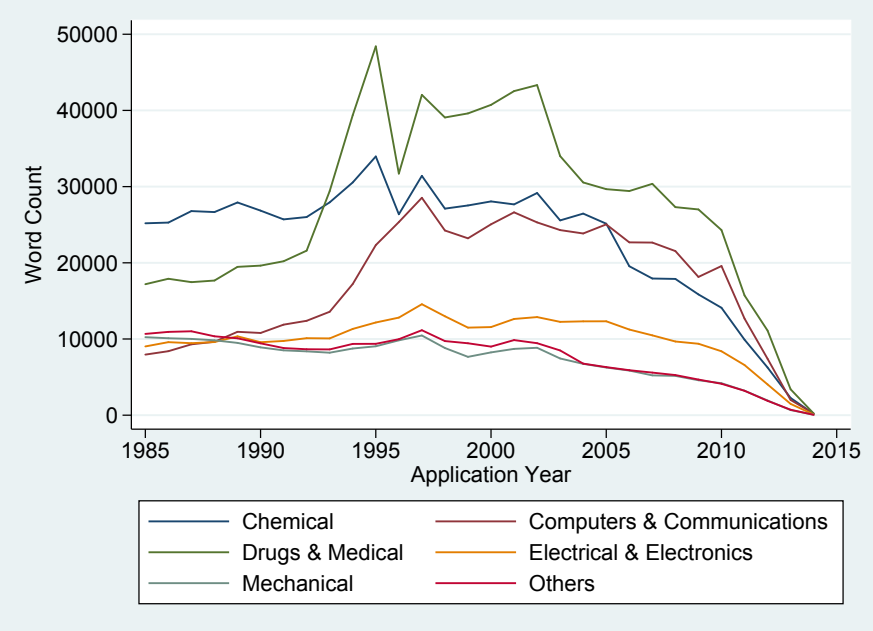

Figure 10: Total number of novel words by the NBER technological category and application year; the peak in the 1990s appears to be caused by an increase in the number of patents granted.

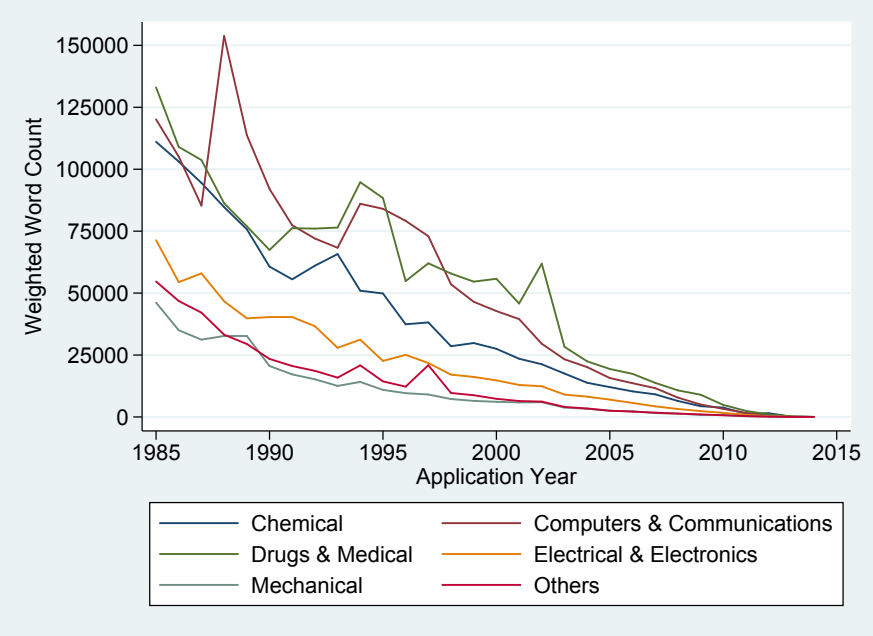

Figure 11: Novel words applied for in a given year, weighted by their future reusage, NBER technology category

Figure 11 illustrates novel words that are frequently reused and could be interpreted as "successful" novelty and Figure 12 lists the top 20 reused words in each five year time period. The peaks in the Figure 11 can be partially explained by the following particular words:

Computers & Communications:

Peak 1988–1990:

- 1988: internet (55,922) e-mail (8,355) hypertext (5,520)
- 1989: computer-executable (28,042) browser (15,794)
- 1990: metadata (14,752)

Peak 1994–1996:

- 1994: email (10,941) html (5,704)
- 1995: http (5,612) java (5,097)
- 1996: website (6,812) (and websites (1,643))

Drugs & Medical:

Peak 1991–1995:

- 1991: abiotic (2,486) transgenes (2,114) cdr3 (1,256) cdr1 (1,191) cdr2 (1,180) (note: CDRs (Complementarity determining regions) are part of the variable chains in immunoglobulins (antibodies) and T cell receptors.)
- 1992: backcross (3,794) apoptosis (3,166) unicast (2,666) chemokine (1,401) scfv (1,170)
- 1993: wap (1,179)

- 1994: introgressed (1,184)
- 1995 (many novel words with future reuse, because of the peak in the number of novel words in 1995 in Drugs & Medical): arylc (686) irinotecan (562) atorvastatin (522) leptin (478) polyaxial (434) aptamers (427) cycloalkylc (409)

Peak 2002:

- sirna (1,382) broxynil (1,062) cycloshexone (825)

Chemical:

Peak 1993: nanotubes (5,802) nanotube (4,600) paclitaxel (1,993) micro-electromechanical (1,319)

Mechanical:

Peak 1987–1989:

- 1987: drm (763)
- 1988: nanocrystalline (1,076) batio (1,045)
- 1989: mems (7,545) nanocrystals (1,081)

Others:

Peak 1994:

- oled (3,446)
- Peak 1997: xml (8,452) izo (1,340)

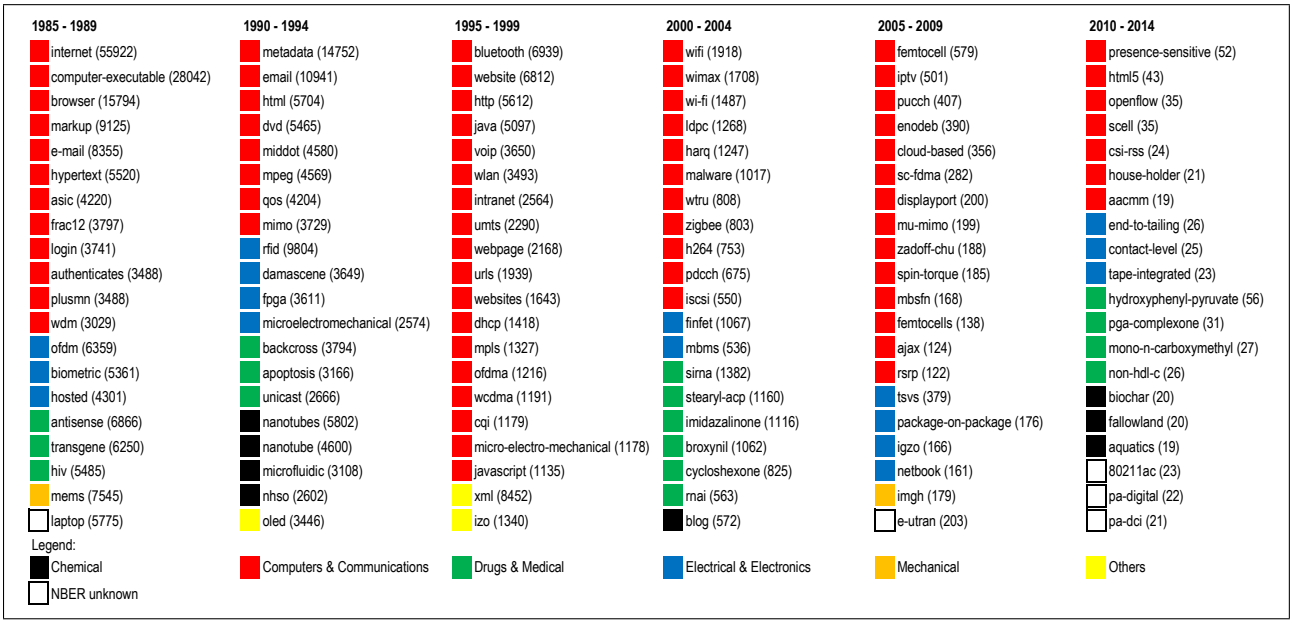

Figure 12: Top 20 most reused novel words within a 5-year-period.

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) or patent(s) chosen by the user. Users may start with either a list of inventors or a list of patents. The tool is at: http://patentnetwork.berkeley.edu/inventors/.

If starting from chosen ("seed") inventors, the tool can find all patents those 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 13 diagrams a process flow.

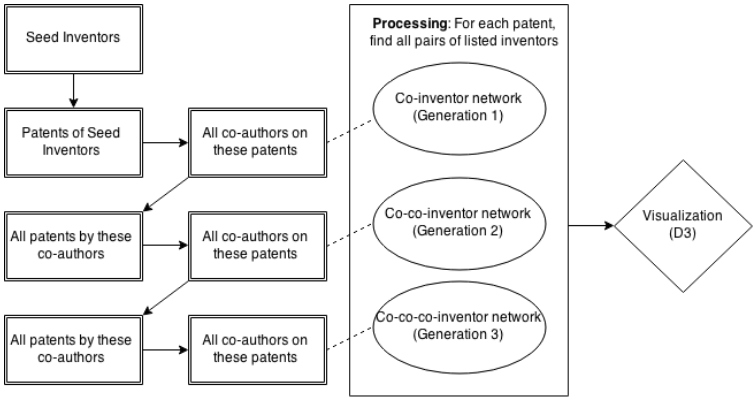

Figure 13: Flow Diagram for Network Visualization Tool

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.

Figure 14 illustrates an example from the semiconductor industry, with two additional co-authorship relationships, for patent sub-class 438/283 from 1998-2000. This follows from the search option that takes a list of patents as input and graphs all inventors associated with that input.

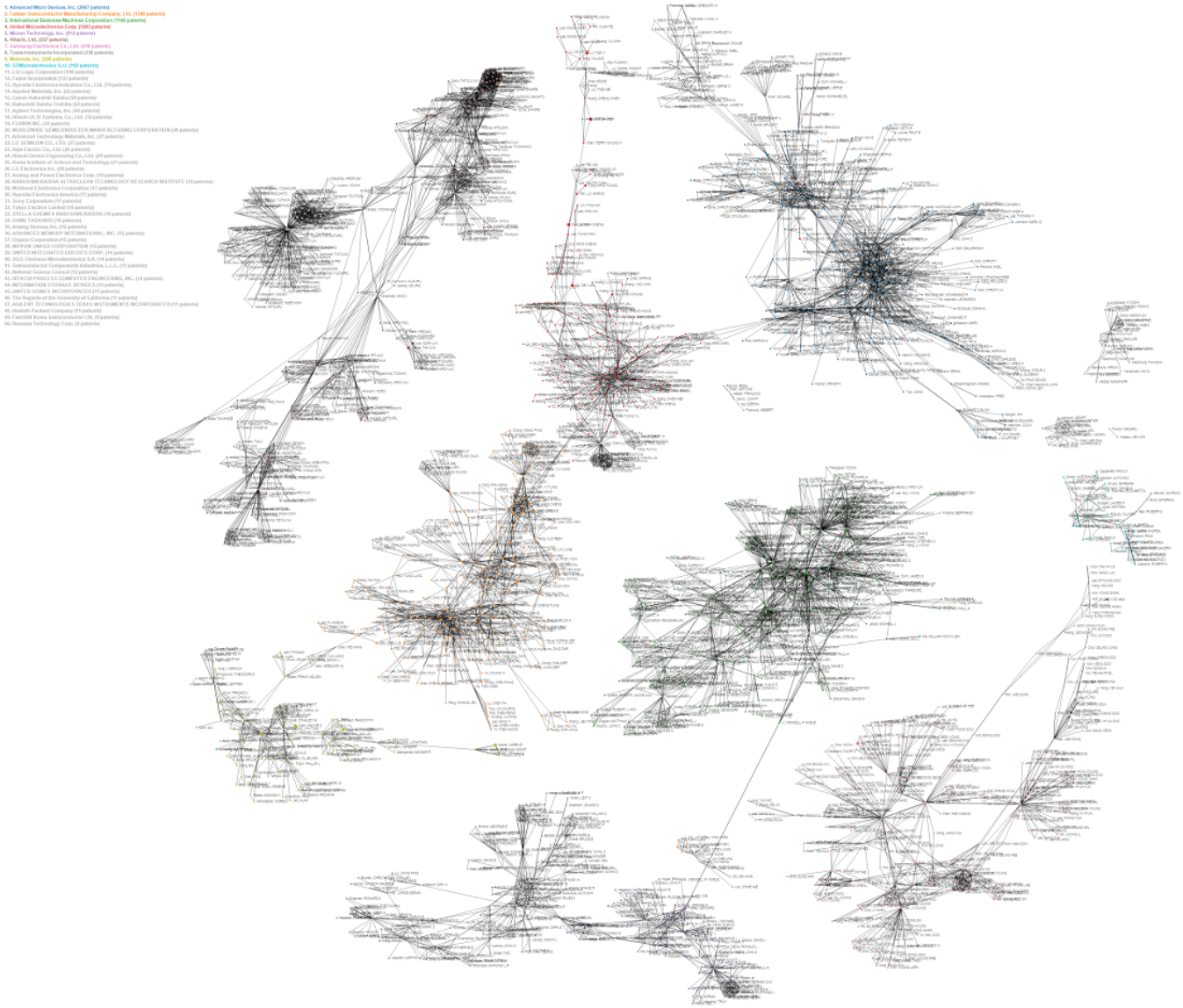

Figure 14: USPTO Semiconductor Patents in Sublass 438/283 from 1998 to 2000, with 2 additional levels of co-authorship included.

Potential improvements

The intent of this work was to make fully automated methods of disambiguation available to the community. Resource constraints unfortunately kept us from exploring some very obvious improvements and deeper characterizations. The methods presented here are simple, and while less accurate than some prior and competing approaches, the hope is that they will provide an automated platform for improvement.

Much work remains to be done (and we encourage the community to continue it). A variety of disambiguation approaches are now available (for example the UMass Amherst inventor disambiguation or the OECD HAN name harmonization); comparison and identification of each dataset's strengths and weaknesses would be invaluable. The social network illustration tool could 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 a distance between technology class measure. 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 (indeed, the USPTO now uses the European classes). 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. For the assignee disambiguation, one could 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.

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 1976 through June of 2016. The tools are simple but provide a platform for future improvement. They first provide inventor, original assignee, and geographical location disambiguations. The work also provided a tool that dynamically renders patent inventor networks, a user interface for generating CSV files, and a measure of patent novelty based on the first appearance of a new word in the patent lexicon. 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.* Forthcoming, Journal of Financial Economics. Working paper, http://www.funginstitute.berkeley.edu/sites/default/files/bfm20150505%20%281%29.pdf

[2] Bernstein, S., 2014. Does Going Public Affect Innovation? The Journal of Finance 70 (4), 1365-1403.

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

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

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

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

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

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

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

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

[11] Kogan, L. and D. Papanikolaou, A. Seru, N. Stoffman forthcoming. *Technological Innovation, Resource Allocation and Growth* Quarterly Journal of Economics.

[12] 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.

[13] 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.

[14] Marco, A. C., and M. Carley, S. Jackson, and A. F. Myers, 2015. *The USPTO Historical Patent Data Files. Two centuries of invention.* USPTO Economic Working Paper No. 2015-1, http://www.uspto.gov/sites/default/files/documents/USPTO_economic_WP_2015-01_v2.pdf.

[15] 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.

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

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