Processing USPTO Patent Data

Gabe Fierro
Coleman Fung Institute for Engineering Leadership
UC Berkeley
fierro@eecs.berkeley.edu

March 25, 2014

Abstract

We describe a completely automated data process designed to consume weekly releases of patent grants and applications distributed by the United States Patent and Trademark Office (USPTO). The process downloads and unpacks the zipped distribution files, parses the raw data into a SQL database, and performs various disambiguations and statistical calculations on the database.

1 Introduction

Patent data plays an invaluable role in research into economic trends, invention, innovation policy and technology strategy. Since the digitization of patent data starting in 1975, though patent data has been freely available through the United States Patent and Trademark Office, it has been difficult to use. We present a substantial improvement in data quality and accessibility over previous third-party re-releases of US patent data. This will not only facilitate further research on up-to-date patent records, but also increase the reproducibility of previous research results.

2 Processing Workflow

The Fung Institute has developed a robust and fully automated toolchain for processing and providing high quality patent data intended for research, as illustrated in Figure 1.

As data is downloaded from the USPTO weekly patent releases, it is parsed, cleaned and inserted into a SQL database. From this database, assignee and lawyer disambiguations are performed and the patents are geocoded with a location-based disambiguation. The output data from these processes are combined with the historical data from the Harvard Dataverse Network into a single consolidated database. From this database, an inventor-level disambiguation can be performed, and various applications can take advantage of the completed data.

3 Data Sources

The unified patent dataset is composed of processed data from two separate sources: the Harvard Dataverse Network (DVN) [12] collection of patent data from 1975 through 2010 and the weekly distributions of Google-hosted USPTO records [9][10].

3.1 Harvard DVN

The Harvard DVN patent database consists of data drawn 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 [13]. The schema of the database can be found in the Appendix.

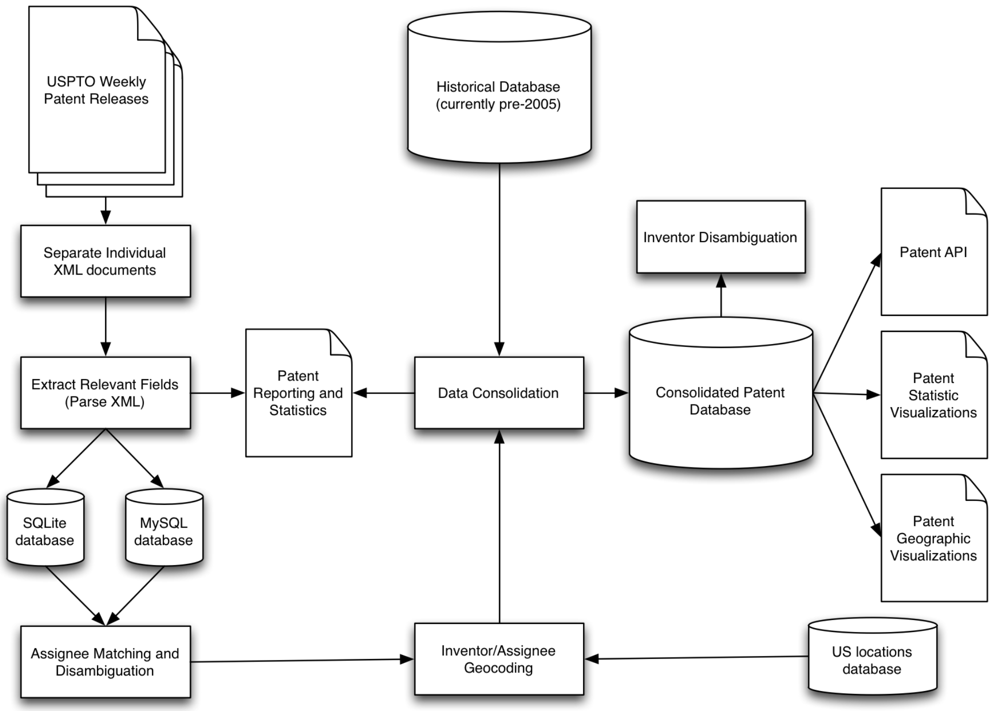

Figure 1: Full patent data process flow

While the Harvard DVN patent database was, prior to the UC Berkeley patent database, the most extensively complete amalgamation of United States patent data, it is not without its problems. Firstly, there is little information as to the actual meanings of the columns in the databases. Without sufficient prior knowledge of patent structure, it is difficult to glean the semantic significance of each column. The names alone are often abbreviated and hard to discern. Secondly, because the DVN database is a combination of several sources into a single database schema, certain patent entries from NBER and Micropatent are incomplete where their data source did not provide all the requisite data. The data obtained from the weekly distributions suffers from being made available in several different formats. The parser that was developed to handle the data is overly complicated and does not handle edge cases well, resulting in missing patent metadata where the parser did not account for a subtle change in format [1]. This is analyzed in greater detail below.

3.2 USPTO Weekly Distributions

The USPTO distributions take the form of zip archives containing concatenated XML (Extensible Markup Language) documents, each of which contains the full text of each patent grant and patent application issued every week. Prior to 1975, the USPTO used a purely paper-based system before transitioning to a raw-text key-value and later SGML-based key-value store [1]. Patent documents were made available in the XML format starting in 2001. Although this data is made freely available, the fact that digital USPTO patent data spans eight different formats and occupies more than 70 GB (compressed) over the 37 years of its existence makes rendering the data into an amenable form a nontrivial problem (see Table 1 and Table 2). Patent application data, though only available in a digital format back

[1]Standard Generalized Markup Language

Time Span	Data Format
-1974	paper-based
1975	unknown. Data obtained from Micropatent
1976 - 2001	Green Book (CITE) APS key-value
2001	SGML ST. 32 v2.4
2002 - 2004	Red Book (CITE) XML ST. 32 v2.5
2005	Red Book XML ST. 36 (ICE) v4.0
2006	Red Book XML ST. 36 (ICE) v4.1
2007 - 2012	Red Book XML ST. 36 (ICE) v4.2
2013	Red Book XML ST. 36 (ICE) v4.3
2013 -	Red Book XML v4.4

Table 1: Table of USPTO grant data formats

Time Span	Data Format
-2001	paper-based
2001	XML ST. 32 v1.5
2002 - 2004	Red Book (CITE) XML ST. 32 v1.6
2005	Red Book XML ST. 36 (ICE) v4.0
2006	Red Book XML ST. 36 (ICE) v4.1
2007 - 2012	Red Book XML ST. 36 (ICE) v4.2
2013 -	Red Book XML ST. 36 (ICE) v4.3

Table 2: Table of USPTO grant data formats

to 2001, is nonetheless available in six different formats [14] [15].

4 Parsing

The process of converting the public patent data into a usable form begins with parsing, the manipulation of a document's grammar and anatomy to extract structured and labeled data. The Fung Institute parser takes as input the weekly USPTO patent distributions and outputs the relevant data into a SQL database. To simplify the problem of parsing the diversity of formats of digital patent data, the current parser addresses only the XML-based documents. At time of writing, the Fung Institute parser is capable of handling patent grants of formats XML v4.0, v4.1, v4.2, v4.3 and v4.4 (spanning 2005 through 2013) and patent applications of formats XML v1.5, v1.6, v4.0, v4.1, v4.2 and v4.3 (spanning 2001 through 2013). Grant data prior to 2005 is drawn from a truncated version of the Harvard DVN database.

The code is written in Python 2 [7] and is available on Github [2].

4.1 XML Overview

XML, or Extensible Markup Language, defines a set of rules for encoding documents that seek to facilitate comprehension by both machines and humans. Since the publishing XML 1.0 standard in 1996, the format gained traction due to the minimal size and flexible structure. In its simplest form, an XML document is a collection of elements, which are each composed of tags and content. Tags, such as `<citation>`, lend semantic structure to a document and allow a reader to determine the significance of the content that follows. An element is a logical component that begins and ends with tags (e.g. `<citation>` and

```
<applicants>
    <applicant sequence="001" app-type="applicant-inventor">
        <addressbook>
            <last-name>Roach</last-name>
            <first-name>Richard</first-name>
            <address>
                <city>Schaumburg</city>
                <state>IL</state>
                <country>US</country>
            </address>
        </addressbook>
    </applicant>
    <applicant sequence="002" app-type="applicant-inventor">
        ...
    </applicant>
</applicants>
```

Figure 2: Sample inventor element from XML v4.2 schema

</citation>) and contains either regular text or additional, nested elements. An example of an element can be found in Figure 2.

4.2 Parser Method

The Fung Institute parser adopts a novel approach to the problem of extracting data from XML documents. As XML documents are fed to the parser, they are transformed from XML's canonical tree-based organization into modified Python dictionaries. Typical XML parsers must make certain assumptions about the nesting and placement of tags and must contain careful allowances for missing, mislabeled, or unexpected tags. The Fung Institute parser circumvents this issue by not requiring a detailed specification of the data to be extracted, instead relying on general descriptors of the location of needed data. This makes the parser more robust and able to handle small schema changes without adjustment, therefore reducing the number of potential runtime errors. The existence of such an engine also expedites the development of additional parsers that handle subsequent changes to the USPTO patent XML schemas.

The XML parsing engine reduces the amount of explicit error checking code while making the source code concise and easy to understand. The engine is easily configurable, and can be directed to automatically download and parse patents in a given date range, apply arbitrary post parsing steps, and deliver the results to a database.

4.3 Data Idiosyncracies

While the USPTO patent data is public and freely available, it is not without its problems.

There is inconsistent usage of HTML idioms and escaping. Underscores, ampersands, emdashes and brackets – to name a few – are not expressed as literal characters in the raw XML, and care must be taken to translate sequences such as `&` and `<sub>—</sub>` so that the extracted data is human-readable.

Accents within names are irregularly represented and follow differing standards. Accented letters are either missing (e.g., "Rémy" becomes "R my") or replaced by description (e.g. "Rémy" becomes "R acute over e my") or replaced by the same letter without accent (e.g., "Rémy" becomes "Remy"). All three versions of the name "Rémy" are found across the DVN databases and USPTO weekly publications.

Last name prefixes such as "van der" and titles such as "Esquire" are varyingly included in either the `<first-name>` or `<last-name>` tags, which complicates the parsing of names into a consistent form.

The document numbers of patents are inconsistently prefixed with letters representing the type of document, and are occasionally padded with a leading "0". These eccentricities exacerbate the logical complexity of the parser, but must be handled in order to maintain consistent notation that enables the reliable tracking of references to documents.

These issues are handled by the Fung Institute parser, and are discussed at length in a previous Fung Institute publication [3].

5 Database

One of the main purposes of the patent processor project is to provide a usable database of relevant patent data. This database should facilitate the retrieval of patent records, citations, inventors, lawyers, assignees, and other patent-related data. The linked nature of these types of records suggests that a relational database model would be most suited to the data, which motivated the decision to model patent data in SQL. SQL, or Structured Query Language, is a language designed for managing data held in a relational database.

Because the majority of the data processing pipeline is written in Python, it is hard to integrate otherwise easy-to-use SQL code. There are multiple flavors of SQL – among them, SQLite and MySQL. SQLite simplifies local development because the whole database is represented as a single efficiently-sized file that can be copied, moved and manipulated much like a traditional file. However, it is hampered by a lack of support for more complex SQL features, and has poor support for concurrent users (e.g. multiple processes attempting to access the same database). MySQL offers advanced SQL features (such as `LEFT OUTER JOIN`) and scales to multiple users and large amounts of data much easier than SQLite, but requires more specialized knowledge to use and access. MySQL is more suited for production environments, whereas SQLite is better for development. We want to be able to easily switch between these two flavors of SQL depending on our purpose without having to develop multiple branches of database integration.

5.1 SQLAlchemy

SQLAlchemy [6] is a Object Relational Mapper (ORM) for Python that seeks to abstract away the differences between SQLite, MySQL, and other SQL-based relational databases. The SQLAlchemy ORM maps Python classes to an underlying SQL database such that the database can be manipulated as though it were a native Python object. This means that the object model and the database schema can be decoupled, effectively removing the need for separate lines of development for each possible database engine.

Database-related code written using SQLAlchemy is much cleaner and easier to work with than the traditional, kludgy idioms. In the case of SQLite, the normal Python module requires the user to excute strings of SQL code:

```
query = 'select * from Patent where \
    number = ''%s''' % patent_number
connection.execute(query)
```

Not only does this require the programmer to know SQL syntax, but this paradigm leaves the database open to SQL injection, wherein unintended and possibly malicious code is executed on the SQL database. For example, here, we are operating on the assumption that the variable `patent_number` contains a valid patent number. It could actually contain the string `''; delete from Patent;--`, which would terminate the original `select` statement, delete all entries from the Patent table, and then exit as though nothing had happened. To avoid such attacks, it is necessary to sanitize all SQL strings

```
query = 'select * from Patent where \
    number = ''%s''' % patent_number
result = connection.execute(query)
patent_id = result[3]
query = 'select * from assignee \
  where patent_id = ''%s''' % patent_id
connection.execute(query)
```

Figure 3: Finding assignees for a patent using traditional Python-SQL

```
patent = session.query(Patent).
    filter_by(number = patent_number)
patent.assignees
```

Figure 4: Finding assignees for a patent using SQLAlchemy

to make sure they contain valid and safe queries.

SQLAlchemy obviates the need to implement such verbose security methods. The SQLAlchemy equivalent to the above query is:

```
session.query(Patent).
    filter_by(number = patent_number)
```

Immediately, we can see that this code is much simpler and cleaner. When SQLAlchemy accepts string input, as with the `patent_number` variable here, it automatically escapes all significant characters like semicolons and apostrophes, essentially nullifying the possiblity of SQL injection attacks.

SQLAlchemy further simplifies the handling of foreign keys and complex joins between tables, and can even implement these features over database engines (such as SQLite) that do not normally have them. Consider Figure 3 versus Figure 4.

5.2 Limitations

The nice features of SQLAlchemy come at a price. The higher level interface to the SQL database requires a nontrivial amount of bookkeeping. Foreign keys lookups and checks introduce a certain amount of overhead, so when a process loops through a list of database items, multiple SQL queries can be executed against the backend for each object if the process asks for linked objects.

SQLAlchemy offers tools to help reduce the number of individual queries sent to the underlying database, but there is an inescapable overhead to using an ORM over the raw SQL.

5.3 New Schema

We wanted to have a highly-linked database that would make it easy for developers to access related information for a given set of patents. The DVN schema, as described in the Appendix, does not take advantage of foreign key relations, and places much manual burden on the user. This was a primary motivating factor in our design, which is summarized in Figure 5.

5.4 Raw vs Processed

If we examine the new database schema, for each of the `inventor`, `lawyer`, `location`, and `assignee` tables, we can see a "raw" version (e.g. `rawinventor`) and a plain version. The `raw` tables contain the inventor, lawyer, location and assignee records *as they appear in the USPTO files*, which means that the naming inconsistencies and misspellings are preserved. These

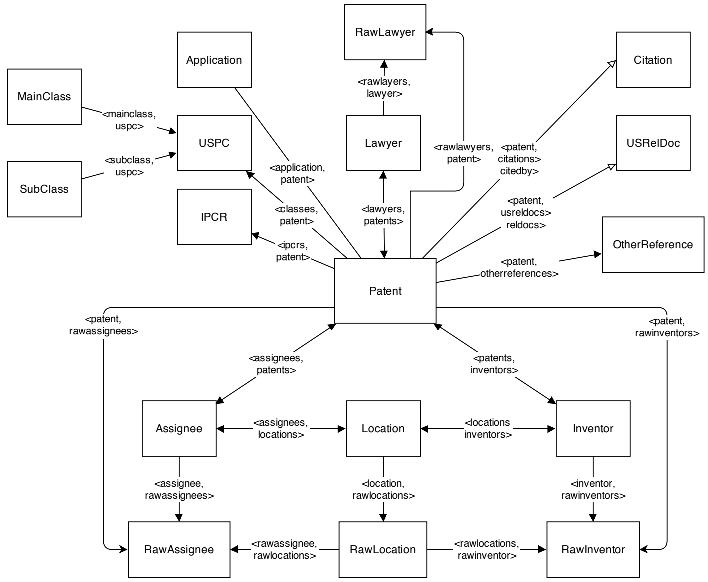

Figure 5: High level view of new database schema

records are run through disambiguation methods of various degrees of rigor, and the cleaned records are stored in the plain tables. See below for a description of these disambiguation methods.

When the cleaned records are inserted, we link them to both the related patent and the raw version using foreign keys in the database, so it is simple to examine groups of related records. See Table 3.

6 Disambiguations

One of the primary problems with conducting meaningful research with USPTO patent data is the high variability in quality. Cities are misspelled or mislisted. Organizations are alternatively abbreviated and listed in full with little modicum of consistency. Inventors, lawyers and assignees will misspell their names, change their names and unpredictably list their middle initials or names. The Berkeley patent database provides facilities to account for these errors, and codifies the disambiguation of such records in order to make possible their accurate retrieval.

6.1 Geocoding

There are over 12 million locations listed in the USPTO patent weekly downloads from 1975 to 2013, with 350,000 unique tuples of `(city, state, country)`. These tuples follow the typical motif of data problems in the rest of the patent data: incorrect or nonstandard country codes, inconsistent romanization of foreign locations and various misspellings. We resolved the ambiguities in the location data using a propietary disambiguation technique developed by Google. When new patent data is processed, we run a series of data cleaning processes to correct for some of the common errors, then cross reference with the lookup table [4] obtained through the Google disambiguation.

A detailed analysis of the problems with USPTO location data and our handling of locations can be found through a related Fung Institute publication [5].

Locations are associated with assignees, inventors and lawyers. Typically, a patent record's "location" is the location of the first inventor listed on the patent.

Table	Access	Value
Patent	`patent`	US8434162
Inventor	`patent.inventors[0]`	Thomas H. Stachler
Raw Location	`patent.inventors[0].rawlocation`	Deyton, OH, US
Clean Location	`patent.inventors[0].location`	Dayton, OH, US

Table 3: Accessing related raw and clean records. Note the spelling correction in the clean record

6.2 Assignees

For a given patent, the assignees are the entities (either organizations or individuals) that have property rights to the patent. The assignee records are imperative for firm-level analysis of patent data, and are used for tracking ownership of patents. The weekly releases of patent documents only contain the original assignee of a patent when it was initially granted.

However, it is difficult to obtain accurate results for simple (and necessary) questions such as *"which patents are owned by firm X?"* because of the pandemic inconsistency of spellings. A cursory search for assignee records that resemble General Electric yields the following:

- 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 is not even a complete list of all the (mis)representations of General Electric, but already we can see the potential issues with trying to get accurate results.

We do not yet provide fully featured entity resolution for assignee records, but we do maintain a preliminary disambiguation of the records that corrects for minor misspellings. We do this by applying the Jaro-Winkler [8] string similarity algorithm to certain pairs of raw assignee records. Two records that are within a certain bound of similarity are considered the same, and are linked together.

It is not tractable to perform pairwise computation on each of the 5,850,531 raw assignee records in the database (at time of writing), so we group the assignees by their first letter, and then perform the pairwise comparisions within each of these blocks. This allows us to hold a smaller chunk of the assignees in memory at each step, with approximate accuracy.

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.

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 dismbiguated record for that pool, and all rawassignees are linked to that disambiguated record.

There is obvious room for improvement in this algorithm – including more global string comparisons and the leveraging of additioanl metadata to further group and lump assignees – but due to current computational constraints, it is not possible to implement these changes within the current framework. This disambiguation delivers a decent fix for the various misspellings occurred in the database.

6.3 Lawyers

The raw lawyer records follow much of the same deficiencies in quality as the assignee records. Again, we only offer a preliminary disambiguation of lawyer records using the same algorithm as described above, but future development will yield more accurate results.

The assignee disambiguation has yet to be implemented for the lawyer record tables.

6.4 Inventors

We provide a polished disambiguation mechanism for inventor records. Using the published name of an inventor, the patent technology class, co-inventor names, published location and original assignee, we are able to infer with more than 95% accuracy which inventor records are the same across all records in the patent database.

A detailed summary of our technique can be found through a related Fung Institute publication [11].

7 Statistics

Many research applications of patent data require records from multiple tables to be linked together: for instance, finding all citations made to a patent, or finding all patents for an inventor. Due to the size of the database, however, gathering all the requisite data and linking it together takes a nontrivial amount of time. To facilitate some common research vectors, we provide three tables of precompiled statistics.

The `FutureCitationRank` table contains the rank of each patent by the number of future citations in each year. This answers the question "in year X, patent number Y got Z citations. It was the Nth most cited patent that year".

The `InventorRank` table contains the rank of each inventor by how many patents they have been granted in a given year.

The `CitedBy` table contains the direct mapping of a focal patent to all patents that cite that patent.

References

[1] Fung Institute Patent Group. `https://github.com/funginstitute/patentprocessor/tree/old_parser`, 2012.

[2] Fung Institute Patent Group. `https://github.com/funginstitute/patentprocessor/`, 2013.

[3] Gabe Fierro . Extracting and Formatting Patent Data from USPTO XML . Tech. rep., Fung Institute; UC Berkeley , 2013.

[4] Jeffrey Oldham, Kevin Johnson, Google Inc. `https://s3.amazonaws.com/fungpatdownloads/geolocation_data.7z`, 2013.

[5] Kevin Johnson . Geocoding Patent Data. Tech. rep., Fung Institute ; UC Berkeley, 2013.

[6] Michael Bayer, SQLAlchemy . `http://www.sqlalchemy.org/`, 2013.

[7] Python Software Foundation. `http://www.python.org/`, 2013.

[8] William E Winkler . Overview of Record Linkage and Current Research Directions . Tech. rep., Statistical Research Division, U.S. Census Bureau , 2006.

[9] Google Inc. `http://www.google.com/googlebooks/uspto-patents-grants-text.html`, 2013.

[10] Google Inc. `http://www.google.com/googlebooks/uspto-patents-applications-text.html`, 2013.

[11] Guan-Cheng Li . Disambiguation of Inventors, USPTO 1975-2013 . Tech. rep., Fung Institute ; UC Berkeley , 2013.

[12] Lai, R., D'Amour, A., Yu, A., Sun, Y., and Fleming, L. Disambiguation and co-authorship networks of the u.s. patent inventor database (1975 - 2010).

[13] MicroPatent. http://www.micropat.com/static/index.htm, 1998.

[14] United States Patent and Trademark Office. http://www.uspto.gov/products/xml-resources.jsp, 2013.

[15] United States Patent and Trademark Office. http://www.uspto.gov/products/xml-retrospective.jsp, 2013.

Appendix

Harvard DVN Database Schemas

Column Name	Column Description
`Patent`	Patent owned by assignee
`AsgType`	Unknown
`Assignee`	Name of assignee
`City`	City location of assignee
`State`	State location of assignee
`Country`	Country of assignee
`Nationality`	Nationality of assignee
`Residence`	Street address of assignee
`AsgSeq`	Order of assignee as appears in patent

Table 1: DVN table schema for assignees

Column Name	Column Description
`Patent`	Patent making the citation
`Cit_Date`	Date of cited document
`Cit_Name`	Unknown
`Cit_Kind`	Type of cited document
`Cit_Country`	Origin of cited document
`Citation`	Number or ID of cited document
`Category`	Unknown
`CitSeq`	Order of citation as appears in patent

Table 2: DVN table schema for citations

Column Name	Column Description
`Patent`	focal Patent
`Prim`	Order of classification
`Class`	USPTO technology class
`SubClass`	USPTO technology subclass

Table 3: DVN table schema for classes

Column Name	Column Description
`Patent`	Patent owned by inventor
`Firstname`	Inventor's first name
`Lastname`	Inventor's last name
`Street`	Inventor street address
`City`	Inventor city
`State`	Inventor state
`Country`	Inventor country
`Zipcode`	Inventor zipcode
`Nationality`	Inventor nationality
`InvSeq`	Order of inventor as listed on patent

Table 4: DVN table schema for inventors

Column Name	Column Description
`Patent`	Patent number
`Kind`	Unknown
`Claims`	Number of claims made by patent
`AppType`	Unknown
`AppNum`	Application reference number
`GDate`	Date of grant
`GYear`	Year of grant
`AppDate`	Date of application
`AppYear`	Year of application
`PatType`	Type of patent (Reissue, Utility, etc)

Table 5: DVN table schema for patents

Column Name	Column Description
`Patent`	focal Patent
`Abstract`	Patent abstract
`Title`	Patent title

Table 6: DVN table schema for patent descriptions

Column Name	Column Description
`Patent`	focal Patent
`Firstname`	Lawyer's first name
`Lastname`	Lawyer's last name
`LawCountry`	Location of lawyer
`OrgName`	Name of law firm or organization
`LawSeq`	Order of lawyer as listed in patent

Table 7: DVN table schema for lawyers

Column Name	Column Description
`Patent`	focal Patent
`Descrip`	Description of scientific reference
`CitSeq`	Order of citation as appears in patent

Table 8: DVN table schema for scientific references

Column Name	Column Description
`Patent`	focal Patent
`DocType`	Type of related document
`OrderSeq`	Order of document as appears in patent
`Country`	Country of origin for related document
`RelPatent`	Patent number of related document
`Kind`	Unknown
`RelDate`	Date of related document
`Status`	Status of related document

Table 9: DVN table schema for US related documents

Fung Institute Patent Database Schemas

Column Name	Column Description
`id`	Unique identifier for application record
`patent_id`	Corresponding granted patent record
`type`	Type of patent application
`number`	Patent application document identifier
`country`	Country in which application was filed
`date`	Date of application submission

Table 10: Application – information on the application for a granted patent

Column Name	Column Description
`id`	Unique identifier for disambiguated Assignee record
`type`	USPTO code for type of assignee
`name_first`	First name of assignee (if individual)
`name_last`	Last name of assignee (if individual)
`organization`	Name of assignee's organization (if firm)

Table 11: Assignee – disambiguated records for the original patent assignee

Column Name	Column Description
uuid	Unique identifier for raw assignee record
patent_id	Patent which contains this record
assignee_id	Identifier of disambiguated assignee record
rawlocation_id	Location of raw assignee
type	USPTO code for type of assignee
name_first	First name of assignee (if individual)
name_last	Last name of assignee (if individual)
organization	Name of assignee's organization (if firm)
sequence	Order in which this assignee was listed on the patent

Table 12: RawAssignee – raw records for the original patent assignee

Column Name	Column Description
patent_id	Patent being cited
citation_id	Document citing the focal patent
year	Year the citation was made

Table 13: CitedBy – precompiled table of future citations

Column Name	Column Description
uuid	Unique identifier for claim record
patent_id	Corresponding patent document for this claim
text	Text of claim
dependent	Sequence number of claim this record is dependent on
sequence	Order in which this claim appears in its patent

Table 14: Claim – full text of patent claims

Column Name	Column Description
uuid	Unique identifier for citation relation
patent_id	Patent making a citation
date	Date of patent making the citation
name	Name of foreign citation
kind	Kind of document being cited
number	Document identifier of cited document
country	Country of origin of cited document
category	Type of citation to cited document
sequence	Order in which the focal patent cited the document

Table 15: ForeignCitation – citations made to foreign patents

Column Name	Column Description
patent_id	Patent being ranked
num_citations	Number of citations the patent received in the given year
year	Focal year
rank	Rank 1 means the patent was the most cited document in the given year

Table 16: FutureCitationRank – rank of each patent by the number of future citations in each year

Column Name	Column Description
id	Unique inventor identifier for disambiguated record
name_first	First name of inventor
name_last	Last name of inventor
nationality	Nationality of inventor

Table 17: Inventor – disambiguated inventor records

Column Name	Column Description
id	Unique raw inventor identifier
patent_id	Patent which contains this record
rawlocation_id	Location of raw inventor
name_first	First name of inventor
name_last	Last name of inventor
nationality	Nationality of inventor
sequence	Order in which this inventor was listed on the patent

Table 18: RawInventor – raw inventor records

Column Name	Column Description
inventor_id	Inventor being ranked
num_patents	Num of patents attributed to inventor in the given year
year	Focal year
rank	Rank 1 means the inventor was granted the most patents in the given year

Table 19: InventorRank – rank of each inventor by how many patents they were granted in each year

Column Name	Column Description
id	Unique lawyer identifier
name_first	First name of lawyer (if individual)
name_last	Last name of lawyer (if individual)
organization	Name of firm (if not individual)
country	Country on record of lawyer

Table 20: Lawyer – disambiguated lawyer records

Column Name	Column Description
id	Unique lawyer identifier of raw record
name_first	First name of lawyer (if individual)
name_last	Last name of lawyer (if individual)
organization	Name of firm (if not individual)
country	Country on record of lawyer
sequence	Order in which this lawyer was listed on the patent

Table 21: RawLawyer – lawyers that worked on the given patent

Column Name	Column Description
id	Unique location identifier
city	Disambiguated city name
state	Disambiguated state name
country	Disambiguated country
latitude	Latitude of disambiguated location
longitude	Longitude of disambiguated location

Table 22: Location – disambiguated location data for assignees and inventors

Column Name	Column Description
id	Unique location identifier
city	Raw city name
state	Raw state name
country	Raw country

Table 23: RawLocation – raw location data for assignees and inventors

Column Name	Column Description
id	USPTO code for main class
title	Title of USPTO main class
text	Description of USPTO main class

Table 24: MainClass – reference table for definitions of USPTO main classes

Column Name	Column Description
uuid	Unique identifier for citation relation
patent_id	Patent making a citation
text	Description of citation
sequence	Order in which the focal patent cited the document

Table 25: OtherReference – citations made to non-patent documents

Column Name	Column Description
type	Category of patent, e.g. "design", "reissue"
number	Unique patent document number
country	Country of origin of patent
date	Date of grant of patent
abstract	Text of patent abstract
title	Text of patent title
kind	USPTO code for type of patent
num_claims	number of claims made by patent

Table 26: Patent – granted patent records

Column Name	Column Description
id	USPTO code for subclass
title	Title of USPTO subclass
text	Description of USPTO subclass

Table 27: SubClass – reference table for definitions of USPTO subclasses

Column Name	Column Description
uuid	Unique identifier for citation relation
patent_id	Patent making a citation
date	Date of patent making the citation
name	Name of US appliation citation
kind	Kind of document being cited
number	Document identifier of cited document
country	Country of origin of cited document
category	Type of citation to cited document
sequence	Order in which the focal patent cited the document

Table 28: USApplicationCitation – citations made to US patent applications

Column Name	Column Description
uuid	Unique identifier for citation relation
patent_id	Patent making a citation
date	Date of patent making the citation
name	Name of US patent citation
kind	Kind of document being cited
number	Document identifier of cited document
country	Country of origin of cited document
category	Type of citation to cited document
sequence	Order in which the focal patent cited the document

Table 29: USPatentCitation – citations made to US granted patents

Column Name	Column Description
uuid	Unique identifier for this relation
patent_id	Patent with this classification
mainclass_id	USPTO main class
subclass_id	USPTO subclass
sequence	Order in which this classification was listed on the document

Table 30: USPC – US patent classes

Column Name	Column Description
uuid	Unique identifier for this relation
patent_id	Focal patent
doctype	Type of related document
status	Status of relation
date	Date of related document
number	Identifier of related document
kind	Kind of relation
country	Country of related document
sequence	Order in which this related document was listed on the document

Table 31: USRelDoc – US Related Documents