Letter of Transmittal

April 10, 2017

Dear Interested Parties,

Team Odin would like to present our final Design Report for the Technology Tracker in its finished form.

The Technology Tracker is a software application that obtains information on American patents from the United States Patent and Trademark Office, performs analysis on the trends of patents, and presents findings on a variety of areas in the form of easily accessible visualizations. Currently, the Technology Tracker suggests other useful searches based on analysis of the results of the initial search and presents data on trends in the number of patents over time, the geographic region in which the patents are being filed, and which companies are filing these patents.

We would like to thank the following individuals for their assistance in bringing this project to fruition: Dr. Greg Donohoe for his direction and valuable advice as our faculty advisor; Dr. Robert Abercrombie for providing the question that led us to pursue this project; and Mr. Bill Thomson for acting as our surrogate sponsor and taking the time to engage with our project and offer direction.

Sincerely,

Team ODIN

Enclosed: Final Design Report

Technology Tracker

By Team ODIN

Christopher Campbell (Camp7325@vandals.uidaho.edu)

Brandon Ratcliff (Ratc8795@vandals.uidaho.edu)

Robert Stewart (Stew9820@vandals.uidaho.edu)

Table of Contents

Background

Originally, the Tech Tracker team had a sponsor who provided some initial direction for the project. The resources included a patent regarding the concept of technology readiness and some code. As the early stages of the project continued, the direction changed and the sponsor was no longer involved. With this change, the team, working with their faculty advisor, developed a new vision. Brainstorming converged on the idea of providing companies, small and large alike, processed data that would help them make the decision about whether or not to invest in a given technology. The team wanted to present in an accessible manner while not simplifying often-complex situations to a simple “invest” or “do not invest” decision.

Problem Definition

Imagine a small company, which for whatever reason has no knowledge of the computer industry, might want know the state of “touch screen” technology. Without a cohesive tool that aggregates essential technology information, the company might be hopelessly lost. They may have some good ideas about technologies like touch capacitive displays, but they would not know enough information about the development of the technology to know if they should invest further in their idea. Enter team Tech Tracker. The team saw the opportunity because technology data, in large part stored in patents, is hard to sort through in its raw form. Creating a tool that automatically accomplishes this would save the user a tremendous amount of time and hopefully provide insight that would not otherwise be available.

The tool could gather, analysis, and display the data in a way that is pain-free and informative to the user. Now, the small company can continue their search regarding touch screens. They might find out that the technology has especially taken off in the past ten years. Perhaps they would see companies like Apple and Samsung as some of the big players in the industry. Further still, they might see San Francisco, CA as one of the hubs of technology development for touch screens. With the Tech

Tracker project, our small company should be able to find out all of this information with an easy-to-use software tool.

Project Plan

The Tech Tracker team consists of three members: Brandon Ratcliff, Chris Campbell, and Robert Stewart. During the two main phases of the project, investigation and development, the team members took on different roles. In the investigation stage, all members performed various small tasks as the brainstorming of the project plan was in its nascent stages. These tasks included things like choosing a specific technology and manually researching data and compiling a report for the technology. More information on the concepts considered during this phase of the project is discussed in the next section.

Once a general plan was developed, each team member loosely took one major role for the remainder of the project. Brandon Ratcliff became responsible for the web app part of the project. This included data pipeline and visualization components. Chris Campbell developed the analysis portion of Tech Tracker. This included keyword analysis for data points like bigrams and trigrams. Robert Stewart spearheaded the data gathering work. This included investigation and data retrieval development.

Since the first half of the first semester was spent resolving our sponsor situation and developing a problem statement and project plan, the initial timeline was not created at the very beginning of the project. Once the team eventually adopted a rough timeline, they generally stuck to it. As the Gantt chart in the appendix shows, the timeline included an investigation period for about eight weeks, about 16 weeks of development, and about 8 weeks to wrap up any loose ends.

Concepts Considered

This section will cover major concepts that were considered as part of the development of the Tech Tracker tool. This section will simply lay out the considered

concepts, but will leave the delineation of the ones selected for the next section. The main decisions highlighted are language choice, tool type, and data sources.

The first major fork in the road was the language selection. The code originally provided to the team from the sponsor was written in Java. The code did not seem to utilize any functionality specific to Java. This language does have some advantages like it's cross-platform nature and familiar C-like syntax. Despite some of these benefits, Java has been out for a number of years and is thus not the most cutting-edge language.

The Tech Tracker team also considered using Python. A more modern high-level language, Python allows quick prototyping but is also incredibly powerful. Wide adoption and a usable package manager has lead to a plethora of libraries that are readily available. As a scripting language, quick checks and tests are incredibly simple. Just a few lines of code, without any extraneous overhead like having to specify input arguments, can get something up and running. Python also incorporates some object oriented functionality. Writing cohesive, complex applications is made feasible. This language also has readable syntax. For example, curly braces are simply replaced with indentation and C's "&&" is understandably "and".

Tool type was one of the other early decisions. One option was to create a desktop application. The main advantage was that desktop applications have been around for awhile and thus are tried and true. Unfortunately, such programs must also be downloaded and installed. This introduces difficulties like cross-platform capabilities and easy of use. Additionally, Python does not have good support for graphical user interfaces.

The other option was building a web app. The more modern approach, web apps make it easy to distribute software to a large scale audience without requiring them to install any extra software. Python is also a great language for a web app backend, so our previous language decision would fit right in. A variety of web visualization libraries exist which make it easy to accomplish what might be tedious for a standard desktop app. Unfortunately, web app architectures require more complicated infrastructure.

Since it runs on the web, it is necessary to have a back end that handles requests, likely a database for user information and cached data, and front end with which the user interacts.

The source of the data was the final major design choice that the team needed to make. From the start, patent data was considered. Unfortunately, no simple, modern, comprehensive data source exists for patent data. Rather, a variety of different data sources fulfill some, but not all, of the important criteria.

Google Patents was one good option. It offered some filtering capabilities and patent full text. Unfortunately, it stopped updating with new patents, so, if the Tech Tracker tool utilized this source, it would be stuck in the past.

The United States Patent and Trademark Office (USPTO) offers a variety of options. Although there are too many to include in this document, there are two that are worth highlighting. The first, Patent Application Information Retrieval (PAIR) Bulk Data offers a modern REST interface to access patent application metadata. The API allows for quick retrieval of aggregate data which is great for trend charts. Complete metadata packages can also be retrieved for specified queries, but these requires multiple requests and sometimes a long wait period. Also, PAIR does not include the full text of patents in its database.

The other main USPTO option is the Patent Full Text (PatFT) database. This database clearly includes full text which is incredibly valuable for gaining insight into data points like patent and keyword relationships. On the downside, PatFT is designed for being used by a human in a browser and not by another application. Thus, it is necessary to parse through html to in order to get valuable data from this source.

Concept Selection

For choosing the language we focused primarily on personal preference and experience, and suitability to the project. We opted to use Python as all of us have experience in both the language and the ecosystem. Additionally, Python seemed well suited to our goals. It has many libraries well suited to web development and data

analysis (some of which we had prior experience in). Python's strength at rapid prototyping was also a strong deciding factor, because at the beginning of the project we still didn't have a clear end goal, so it was likely that we would have to change directions along the course of the project. Python would have allowed us to do so much easier than the alternative languages considered.

When deciding between developing a web application, or a desktop application, we again primarily focused on team member experience, and suitability to the project. Collectively, our team had significantly more experience with web applications compared to desktop applications. Interactive data visualizations are much easier to create using web technologies then in desktop applications, and some of our team members already had some experience with some web visualization libraries. After considering this, a web application was the obvious choice.

When selecting the data sources we would use, we considered three metrics:

1. How easily accessible is this data source?
2. How complete is this data source?
3. How compatible is this data source with other sources?

A complete decision matrix containing our assessment of each data source over these metrics is available in the appendix. We decided to use the USPTO PAIR Bulk Data source as our primary data source, and then supplement it with the PatFT data source for some data points.

System Architecture

Our system architecture consists of two main parts: the frontend which is run in the browser and is responsible for displaying the user interface-- and the backend, which runs on a server and is responsible for providing all of the data to the front end. This section will discuss both of these parts.

Backend

Our backend has three major layers: the communication layer, the data pipeline layer, and the data analysis and retrieval layer.

The communication layer's main job is managing the flow of information between the front end and the backend. The process starts when the frontend opens a new websocket (a type of two-way communication channel) between itself and the backend, and sends the backend a search term. The communication layer then receives this request, and splits it up into several jobs which it then passes into the data pipeline layer. Some of these jobs are dependent upon the completion of previous jobs, so the communication layer is responsible for making sure that each job is run in the correct order, and paralyzing jobs whenever possible. For example, the one of the first jobs that is run retrieves the full list of patents for each query. After the completion of that job, two other jobs to retrieve the location of each of each of the patents, and to retrieve the full text of each patent are started in parallel. While each job is running, the communication layer sends the frontend any data that is generated, and any status updates or errors generated.

The data pipeline layer is responsible for managing and storing the data in the backend. Each job started by the communication layer, and many parts of the data analysis/retrieval layer requires information from the data pipeline. For each of these requests, the data pipeline first checks to see if it has the data already. If it does, it returns the stored data, if not it passes the request down to the data analysis/retrieval layer, takes the response and stores it in the database, and then returns it to the requester. Having this central layer for data allows us to cache data between different queries or data analysis steps.

Lastly, the data analysis/retrieval layer contains all of the code necessary to communicate with our data sources, and perform any data or manipulation functions.

Frontend

Our frontend has two major layers: the communication/data layer, and the visualization layer.

The communication/data layer is responsible for opening a new websocket for every search query, and receiving data from the backend. For each query, it keeps track of the current status as reported by the backend, and the data for each visualization type. Every time an update is received from the server, it is processed, and any affected visualizations are notified.

The visualization layer consists of several self contained components-- one for each visualization we have. These receive their data from the communication/data layer and are responsible for drawing the visualization, or interacting with any data visualization libraries.

While this architecture could be simpler (and it was much simpler when we first started our project), this separation allows us to create a much more interactive and efficient end product. Much of the complexities arise from the fact that retrieving data from the data sources can take a very long time, so we need be able to cache whatever data we can, and we need to be able to send multiple progress updates to the client throughout the data collection process. The end result is a better user experience as we can have visualizations that are quick to display and continually improve as more data is retrieved and process, and similar or identical queries can share data whenever possible.

Future Work

There are many facets of this project that could be improved in order to add value to the project as a whole. Further analysis of the data set as a whole and the implementation of alternative backup of the database and memory dedicated to storing frequently used data are two of the immediately apparent enhancements that could be

implemented. These are facets that we would have liked to include but that were cut due to time constraints.

Analysis of the results and a survey of the data set could lead to the inclusion of more definitive results which may be found useful. Further interpretation of the data into a recommendation system is something that we initially planned to pursue that could more directly translate the data presented into insight for users. This could be pursued with a compilation of trends among a large sample of queries allowing for common result patterns to be identified and compared to new queries in search of similarities.

Long term storage of the database and more local storage of data such as coordinate to geographic location mappings could also be implemented to the advantage of the application. These are currently stored as they are generated, but access to a greater pre-generated set would be substantially faster than the current method. Storage of this information would allow for much faster access to query results at the cost of a great deal of storage space.

For more ideas on the improvement of this work, please feel free to contact the team with ideas or questions.

Appendices

Decision Matrix

	Effectiveness	Speed of search	Cost	Skill Required	Risk of bad decision
Manual Search	1	5	0	2	4
Hire Patent Expert	4	2	5	0	2
Implement Software Solution	5	5	0	3	1
Hire Consultants	4	2	5	0	1
Do no Research	0	0	0	0	5
Scale:	Color Scheme:				
5 - Maximum	Desired Outcome				
0 - Minimum	Undesired Outcome				
Desired Outcome	5	5	0	0	0
Desired Outcome Meaning	Very likely to suceed	Very fast searching	Minimal or nonexistant cost	Skill required is low	Making bad decision if very unlikely

Patent Data Source Decision Matrix

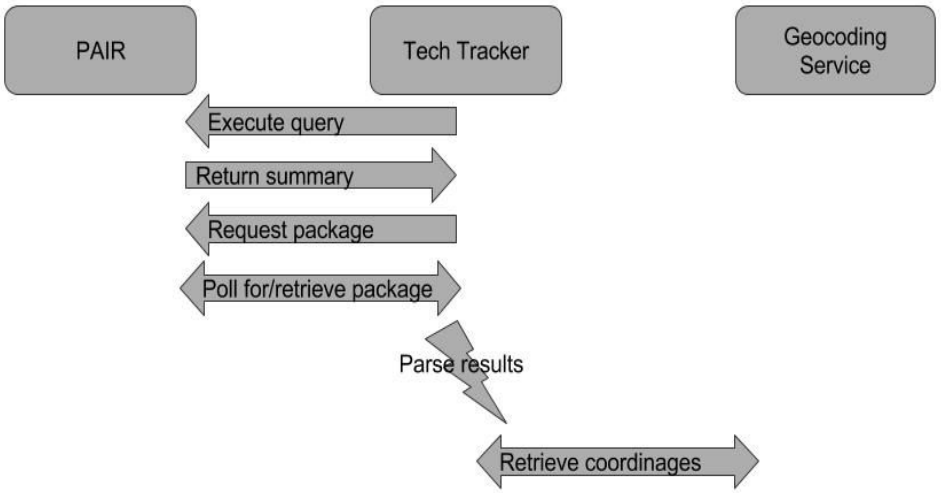
Data Source	Availability	Completeness	Compatibility
USPTO Linked Data	Very easy to access, although it requires specialized tools to access. This data can be queried without downloading it, using a SQL-like language.	Contains only the metadata (no full text), and it is no longer being maintained as of 2015.	If we could find data sets for our other types of data, then it would be very easy to combine them. It's unlikely that these data sets exist though.
USPTO Data Dumps	Not accessible online, we would have to download it, and convert it into a database format we could query	Very complete, contains all the metadata and full text for every patent since 1790. We would have to manually deal with updating our local copy though.	Very compatible, as we could preprocess the data into whatever format we want before storing it.
USPTO API	Accessible online using a query language that doesn't look like it's as powerful as SQL, but still very capable. It's very easy to access using standard code libraries.	Very complete, contains all available metadata, and is constantly updated. A separate API is available for accessing the full text of a patent.	Fairly compatible. We'd have to transform the data before combining it with another data source, but the way the data is returned makes this easy to do.

Specifications

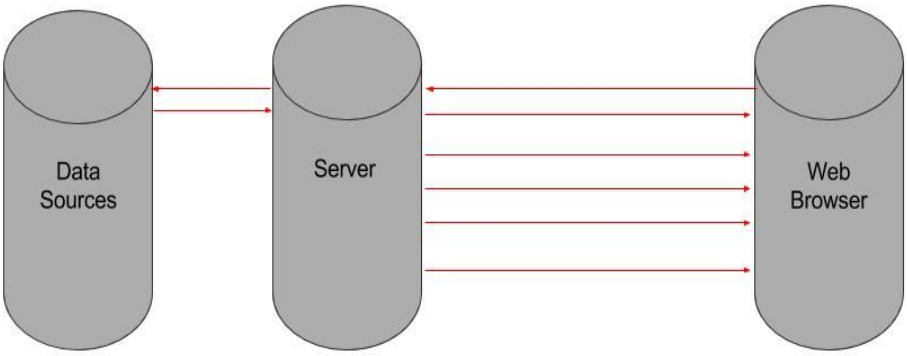
Goals	Requirements

Find trends in datasets	For a data set, identify trends or quantitative data that can be extracted from it that would be useful for understanding the current state of a technology.
Search for a given technology in a dataset	For a given technology, narrow down a data set to include only the things that relate to that technology.
Find related technologies	For a given technology, find any keywords or additional technologies that could be useful to understand the given technology.
Graph data	From the resulting searches and trends, create graphs that accurately represent this data
User Interface	Create a user interface providing the above functions to a user who is not necessarily well versed in technology.

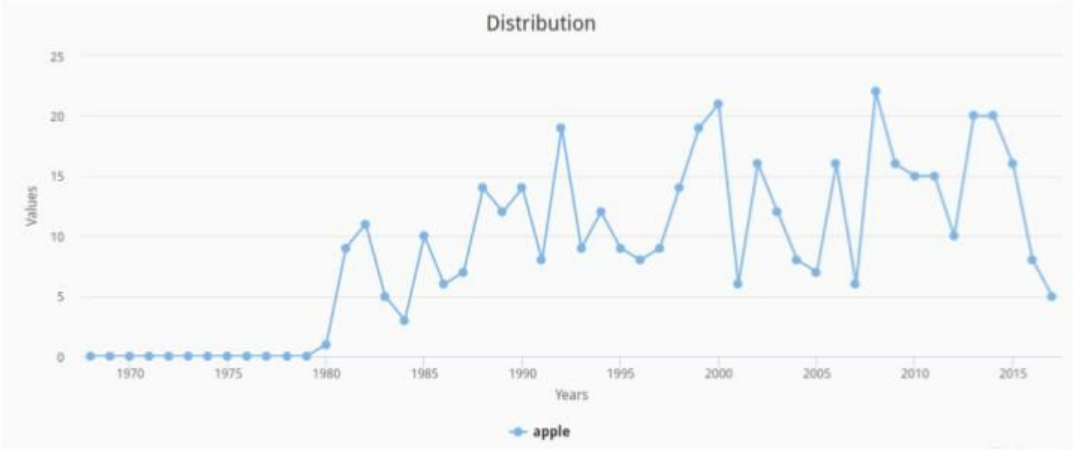
Data Retrieval Process



Data Pipeline



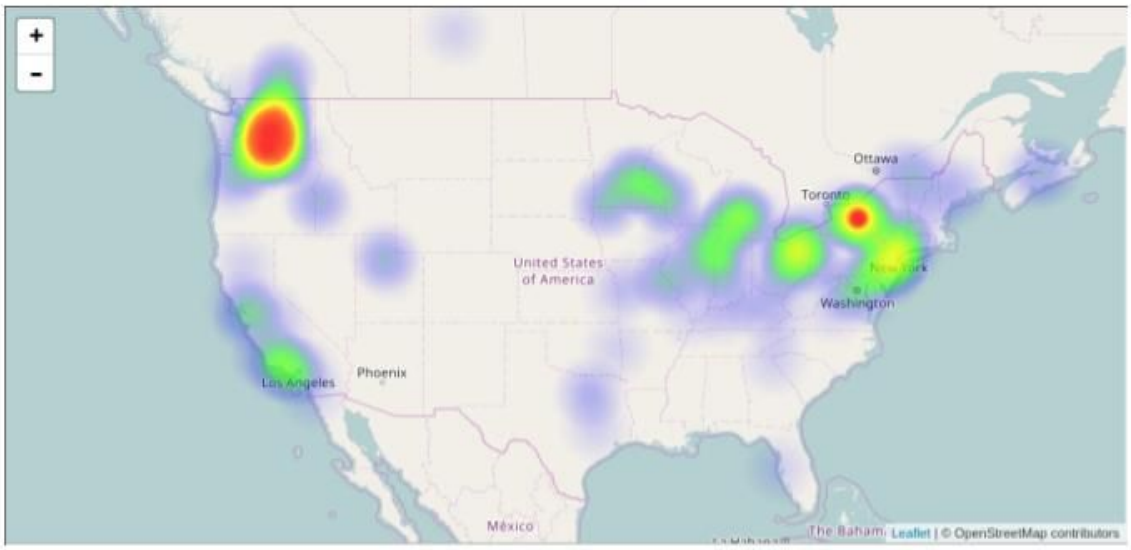
Trend Visualization



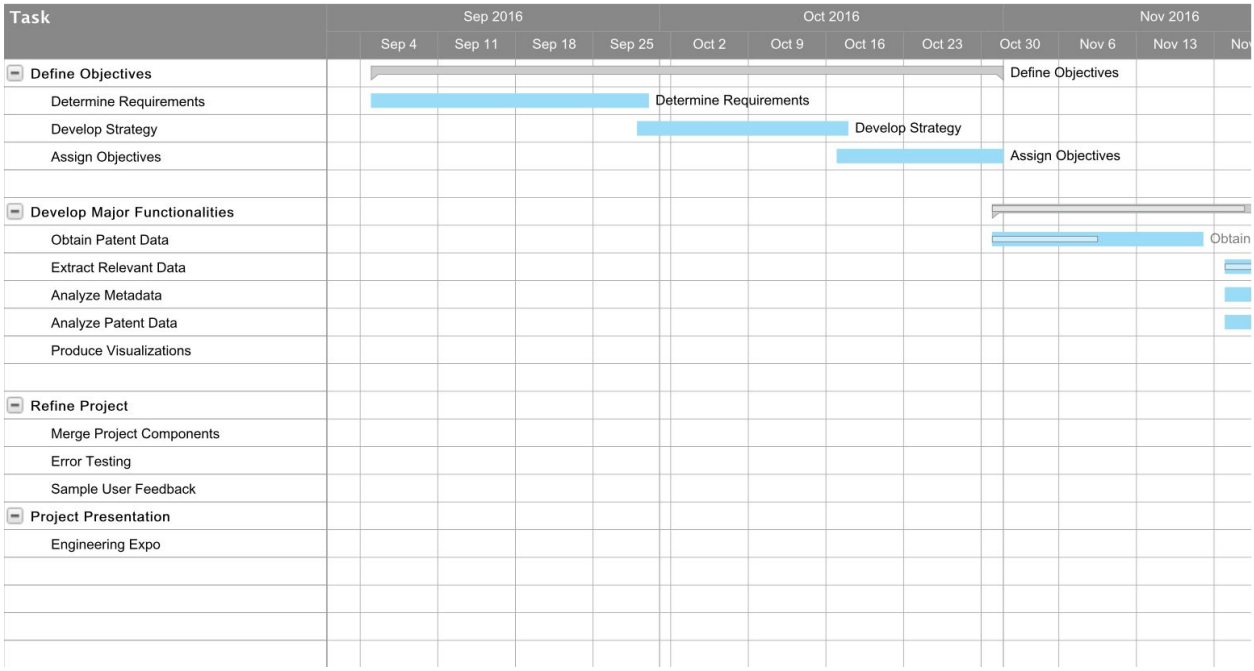
Similar keywords:

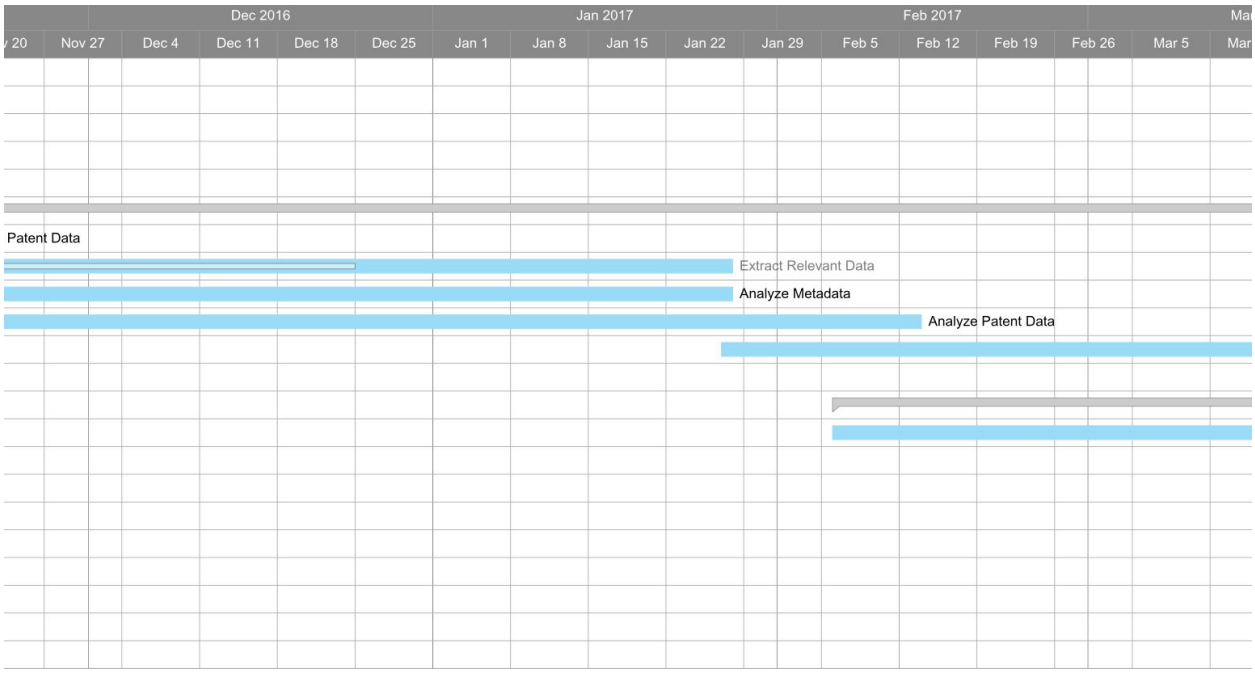
+ apple	+ fruit	+ new	+ variety	+ color	+ tree

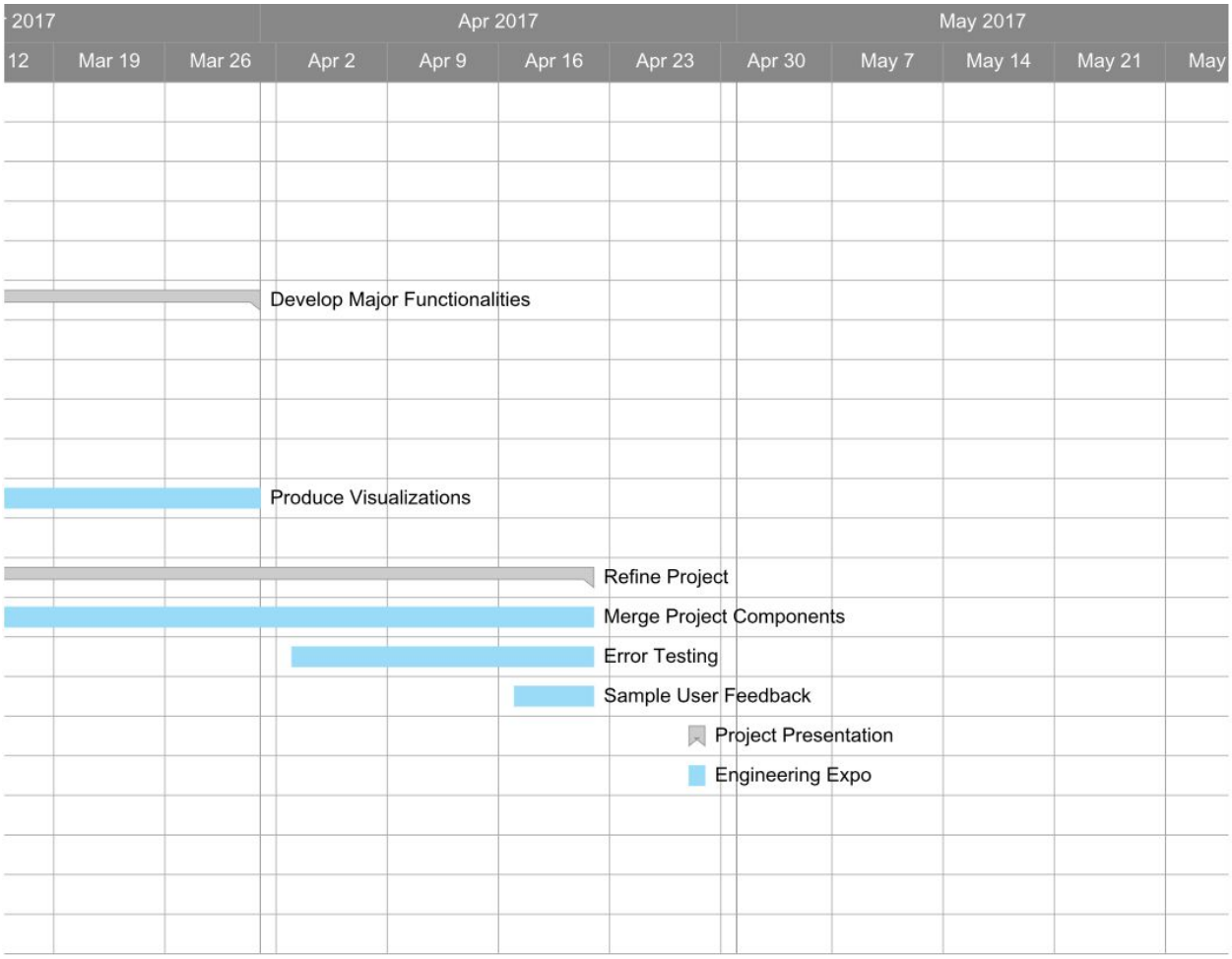
Geographic Heatmap Visualization



Timeline


Task
Sep 2016
Oct 2016
Nov 2016
Sep 4
Sep 11
Sep 18
Sep 25
Oct 2
Oct 9
Oct 16
Oct 23
Oct 30
Nov 6
Nov 13
Define Objectives
Determine Requirements
Develop Strategy
Assign Objectives
Develop Major Functionalities
Obtain Patent Data
Extract Relevant Data
Analyze Metadata
Analyze Patent Data
Produce Visualizations
Refine Project
Merge Project Components
Error Testing
Sample User Feedback
Project Presentation
Engineering Expo


Dec 2016
Jan 2017
Feb 2017
Nov 27
Dec 4
Dec 11
Dec 18
Dec 25
Jan 1
Jan 8
Jan 15
Jan 22
Jan 29
Feb 5
Feb 12
Feb 19
Feb 26
Mar 5
Patent Data
Extract Relevant Data
Analyze Metadata
Analyze Patent Data


2017
Apr 2017
May 2017
12
Mar 19
Mar 26
Apr 2
Apr 9
Apr 16
Apr 23
Apr 30
May 7
May 14
May 21
May
Develop Major Functionalities
Produce Visualizations
Refine Project
Merge Project Components
Error Testing
Sample User Feedback
Project Presentation
Engineering Expo