ITCS 6100 – Big Data for Competitive Advantage
Group Project Phase I
Data Extraction and Data Wrangling

Presidential Election and Patent Analysis using Hadoop

The project is intended to find insights from patent filed by the *Fortune 500* companies in the United States of America and how they influence the Presidential Elections in the country and vice versa. To achieve this, we require two main data.

- Patent grant information of the top Fortune 500 companies
- Presidential Election information for the past three terms.

Our team used the strategies explained below to collect and clean the unstructured data.

Patent Data Collection

The *unstructured data* of the United States patents are available in the server in *XML* format and the size of the data was around 330*GB*. Though *XML parsing* can be handled by Standard Java SAX or DOM parser, the size of the file is challenging for a simple *stand-alone system*. Therefore, handling the data through *distributed parallel* system like Hadoop is a possible solution.

The European patent data for past 10 years are collected from the *EPO Database* is also handled using the Hadoop *MapReduce* technique. Since the analysis is done on only several companies from the Fortune 500 list, the Japanese Patent Data is collected for only some companies.

Data Cleaning Technique

Patent data is cleansed using Hadoop MapReduce program written in Java Programming Language. A MapReduce job is initialized and it calls the mapper and reducer class over the patent XML file. The *mapper* reads every tag and its value from the file and it compares it with the contents of the properties file. The properties file contains a list of key-value pair for the fortune 500 companies where the key is a regular expression matching the company name as the value. Every time the mapper finds a match, it writes the value and one as the count to the context. The *reducer* reads the output from the context and combines the count for each companies and writes it to a file.

Other Data Collection

From the fortune list, we have decided three domains (Finance, Automobile and Technology) that are more popular in correlation with the election. We have selected two companies in each of this industries and have collected data related to these companies.

Though the patent count for the companies are collected using MapReduce, there are other information that are required for the analysis. The Election Data from 1950 to 2016 is collected along with the companies that funded for the political parties. It is also important to get the annual income of the fortune 500 companies as may rely on the election results.

We used *AlchemyAPI* to get the frequent words in the documentation of the target companies, then we used *Legiscan API* to find the bills passed in those name. Later these bills are compared with the list of bills passed in the whole year from government site.

Future Implementation

These data can be used to form a model that predict the outcome of the target companies based on the results of the 2016 election results. As the target companies that we have selected funded for some political party, the result may reflect on their patents they release this year and the revenue of their company.

The prediction model and other analysis on these data are reserved for the phase 2 of this project.

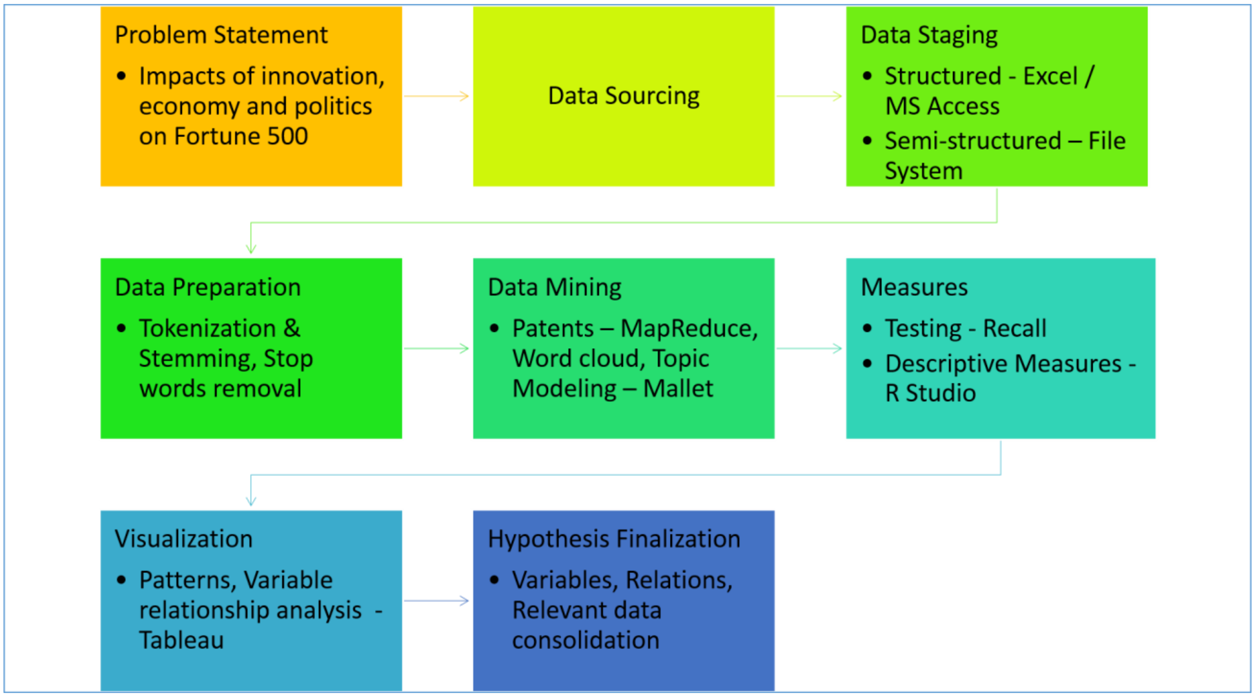
APPENDIX

I.	Fortune 500	It is an annual list published by Fortune magazine that ranks 500 of the largest U.S corporation by total revenue for an year.
II.	Unstructured Data	Data that is not in a standard format for processing.
III.	XML	Extensive Markup Language, it is used to store and transfer.
IV.	XML Parsing	Extensive Markup Language, it is used to store and transfer
V.	Stand-Alone System	Simple PC computer or a personal laptop.
VI.	Distributed System	A bunch of components connected and communicate with each other by passing messages.
VII.	EPO Database	European Patent Office Database containing european patent information.
VIII.	MapReduce	It is a programming model for processing and generating large dataset on a cluster.
IX.	Mapper	A function that apply certain logic to each input in the map.
X.	Reducer	A function that combines the similar values from list of input.
XI.	AlchemyAPI	AlchemyAPI helps developers and businesses build cognitive applications through text analysis and deep learning.
XII.	LegiscanAPI	An API used to get the list of bills for a specific word

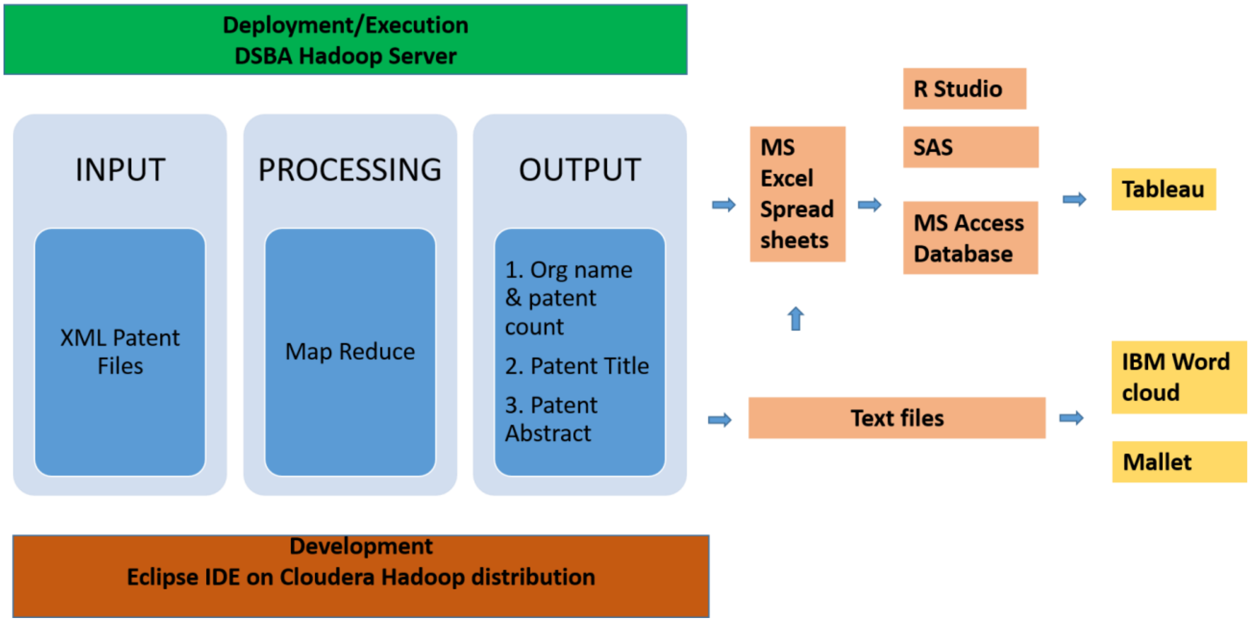
Dataset Assembled:

Data Set Description	**Data Source**	**File Format**	**Data collection method**	**Essential commands**	**Time Period**
Company revenue and social responsibility data & enrichment data	www.compustat.com	Excel	Direct download	VLOOKUP EXCEL VIEW SQL	1970 - 2015
Economy	http://www.bea.gov/national/index.htm#gdp	HTML	Manual text copy		1970 - 2015
Politics	http://www.fec.gov/data/CandidateDisbursement.do	CSV	Direct download	VLOOKUP	1970 - 2015
Fortune 500 Company - market performance data	http://www.hoovers.com/ UNCC Library access to Hoovers	HTML	Manual text copy	EXCEL VIEW SQL	2015
US Patent Data	/projects/class/dsba-6100/patentData2000_2015/patGrants2005_2015h1_ipgs	XML / ZIP	Linux Command	WGET UNZIP CP	2005 - 2015
US Patent Statistics	http://www.uspto.gov/	Excel	Direct Download		1969 - 2015
European Patent Data	PATSTAT Online- https://data.epo.org/expert-services/index-2-2-5.html	XML / ZIP	Direct Download		2005 - 2015
European Patent Statistics	http://worldwide.espacenet.com /advancedSearch?locale=en_EP	XML / ZIP	Direct Download		2005 - 2015
Bulk US Patent data	https://bulkdata.uspto.gov/	XML / ZIP	Direct Download		Need basis

High level design - Process flow diagram


Problem Statement
• Impacts of innovation, economy and politics on Fortune 500
Data Sourcing
Data Staging
• Structured - Excel / MS Access
• Semi-structured – File System
Data Preparation
• Tokenization & Stemming, Stop words removal
Data Mining
• Patents – MapReduce, Word cloud, Topic Modeling – Mallet
Measures
• Testing - Recall
• Descriptive Measures - R Studio
Visualization
• Patterns, Variable relationship analysis - Tableau
Hypothesis Finalization
• Variables, Relations, Relevant data consolidation

High level design - Architecture



Hadoop server

UNCC Research Computing has a 48-core Hadoop cluster (1 NameNode, 6 Slaves) available for use by faculty and graduate student researchers. Our Hadoop cluster also has a 10.75TB Hadoop Distributed File System (HDFS).

Cloudera Hadoop distribution

Open source platform distribution, including Apache Hadoop

High level design - Technology stack

a. Java
b. Eclipse IDE
c. SAS
d. MS Access
e. MS Excel
f. IBM Word Cloud
g. Mallet
h. Tableau
i. R-Studio
j. SPSS Statistics 23

Implementation Code:

XML Parsing of US and European Patent Data using Hadoop MapReduce in Java:

Main Method:

```
public static void main(String[] args) throws Exception {
            //Define a new configuration file
            Configuration conf = new Configuration();
            conf.set("xmlinput.start", "<us-patent-grant");
            conf.set("xmlinput.end", "</us-patent-grant>");
            // Keep properties file in classpath when executing in eclipse
            // Keep properties file in local path when executing in HDFS
            File file = new File("fortuneData.properties");
            Job job = Job.getInstance(conf);
            //Distributed Cache
            job.addCacheFile(file.toURI());
            job.setJarByClass(XMLParsing.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            job.setMapperClass(Map.class);
            job.setReducerClass(Reduce.class);
            job.setInputFormatClass(XmlInputFormat1.class);
            job.setOutputFormatClass(TextOutputFormat.class);
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
                  job.waitForCompletion(true);
      }
}
```

Custom Class:

```
public void initialize(InputSplit split, TaskAttemptContext context)
                  throws IOException, InterruptedException {
      Configuration conf = context.getConfiguration();
      this.startTag = conf.get("xmlinput.start").getBytes("utf-8");
      this.endTag = conf.get("xmlinput.end").getBytes("utf-8");
      FileSplit fileSplit = (FileSplit) split;
      this.start = fileSplit.getStart();
      this.end = (this.start + fileSplit.getLength());
      Path file = fileSplit.getPath();
      FileSystem fs = file.getFileSystem(conf);
      this.fsin = fs.open(fileSplit.getPath());
      this.fsin.seek(this.start);

}
```

Mapper Class:

Initializing Map Class:

```
public static class Map extends
                  Mapper<LongWritable, Text, Text, IntWritable> {
            private static final IntWritable one = new IntWritable(1);
            private Text word = new Text();
            //MultipleOutputs<Text, IntWritable> mos;
            static Set<Object> alist = new HashSet<Object>();
}
```

Call Setup Method for Every Map Instance:

Map Method to define Map Functions:

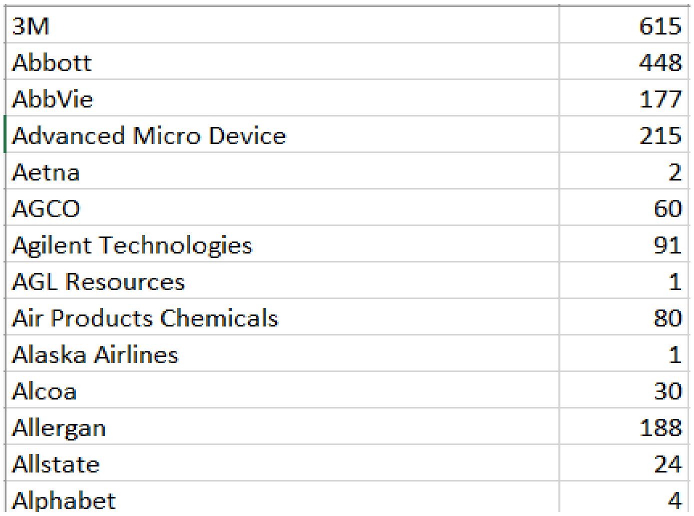
```
protected void setup(
                        Mapper<LongWritable, Text, Text,
IntWritable>.Context context)
                        throws IOException, InterruptedException {
                  URI[] path = context.getCacheFiles();
                  properties = new Properties();
                  InputStream stream = null;
                  super.setup(context);
                  File file = new File(path[0]);
                  stream = new FileInputStream(file);
                  properties.load(stream);
                  stream.close();
                  alist = properties.keySet();
            }
```

```
@SuppressWarnings({ "unchecked", "rawtypes" })
            protected void map(LongWritable key, Text value, Mapper.Context
context)
                        throws IOException, InterruptedException {
            String document = value.toString();
            document = document.replace("&#x26;", "dsba");
            Boolean flag = false;
            try {
      XMLStreamReader reader =
XMLInputFactory.newInstance().createXMLStreamReader(
            new ByteArrayInputStream(document.getBytes()));
            String currentElement = "";
            String propertyName = "";
            while (reader.hasNext()) {
            int code = reader.next();
            switch (code) {
            case 1:
            currentElement = reader.getLocalName();
                  break;
            case 4:
            if (currentElement.equalsIgnoreCase("assignees")) {
                        flag = true;
            }else if(currentElement.equalsIgnoreCase("orgname") &&
flag==true){
                  propertyName = reader.getText();
                  propertyName = propertyName.trim();
                  propertyName = propertyName.toLowerCase();
                  if(propertyName.contains("dsba")){
                  propertyName = propertyName.replace("dsba", "&");
                  }
                  Iterator iterator = alist.iterator();
                  while(iterator.hasNext()){
                  String temp = (String)iterator.next();
                  temp = temp.trim();
            if(propertyName.matches(temp)){
            this.word.set(properties.getProperty(temp));
            context.write(this.word, one);
                              }
                        }
                        flag = false;
                  }
                  break;
            }
      }
      reader.close();
      } catch (Exception e) {
            throw new IOException(e);
            }
      }
```

Reducer Class:

```
//Reducer begins //
public static class Reduce extends Reducer<Text, IntWritable, Text, 
IntWritable> {
            public void reduce(Text key, Iterable<IntWritable> values,
      Reducer<Text, IntWritable, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
            //Same as word count
            int sum = 0;
            for (IntWritable val : values) {
                  sum += val.get();
            }
            context.write(key, new IntWritable(sum));
      }
}
```

Sample Output of Mapreduce Program:

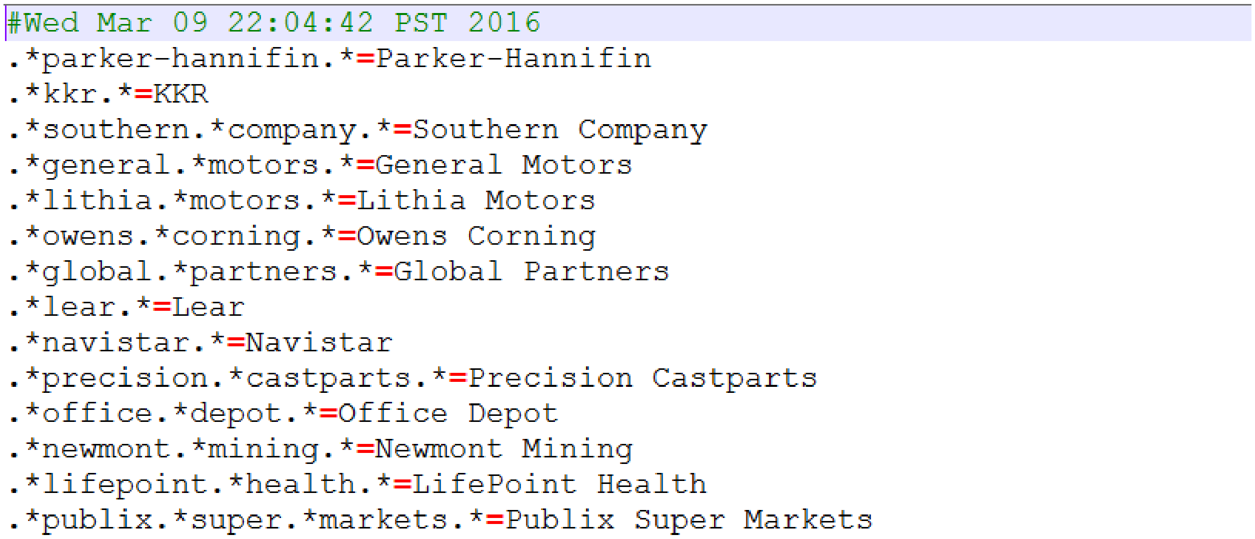


3M	615
Abbott	448
AbbVie	177
Advanced Micro Device	215
Aetna	2
AGCO	60
Agilent Technologies	91
AGL Resources	1
Air Products Chemicals	80
Alaska Airlines	1
Alcoa	30
Allergan	188
Allstate	24
Alphabet	4

Generating Properties File:

```
public static void main(String args[]) {
        Properties prop = new Properties();
        OutputStream output = null;
        ArrayList<String> alist = new ArrayList<String>();
        String filePath = System.getProperty("user.dir") + "/"
                                + "fortunecleaned.txt";
        BufferedReader inputStream = null;
        Pattern REPLACE = Pattern.compile("[][.,';?!#$%<>/(){}]]");
        try {
        output = new FileOutputStream("fortuneData.properties");
        try {
        inputStream = new BufferedReader(new FileReader(filePath));
        String lineContent = null;
        while ((lineContent = inputStream.readLine()) != null) {
          REPLACE.matcher(lineContent).replaceAll("");
          String[] rawSplit = lineContent.trim().toLowerCase().split(" ");
          alist.add(rawSplit[0]);
          StringBuilder builder = new StringBuilder(".*");
          for(int i=0;i<rawSplit.length;i++){
          builder.append(rawSplit[i]+".*");
          }
          prop.setProperty(builder.toString(),lineContent.trim());
        }
                File file = new File("firstword.txt");
                if(!file.exists()){
                        file.createNewFile();
                }
                FileWriter fw = new FileWriter(file.getAbsoluteFile());
                BufferedWriter bw = new BufferedWriter(fw);
                for(int i=0;i<alist.size();i++){
                bw.write(alist.get(i));
                bw.newLine();
                }bw.close();
        } finally {
                if (inputStream != null)
                        inputStream.close();
                        prop.store(output, null);
                        if (output != null) {
                        try {
                                output.close();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                }
        }
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

Sample Properties File:



```
#Wed Mar 09 22:04:42 PST 2016
.*parker-hannifin.*=Parker-Hannifin
.*kkr.*=KKR
.*southern.*company.*=Southern Company
.*general.*motors.*=General Motors
.*lithia.*motors.*=Lithia Motors
.*owens.*corning.*=Owens Corning
.*global.*partners.*=Global Partners
.*lear.*=Lear
.*navistar.*=Navistar
.*precision.*castparts.*=Precision Castparts
.*office.*depot.*=Office Depot
.*newmont.*mining.*=Newmont Mining
.*lifepoint.*health.*=LifePoint Health
.*publix.*super.*markets.*=Publix Super Markets
```

Alchemy API Implementation in Java:

```java
class KeywordTest {
    public static void main(String[] args) throws IOException, SAXException,
            ParserConfigurationException, XPathExpressionException {
        // Create an AlchemyAPI object.
        AlchemyAPI alchemyObj = AlchemyAPI.GetInstanceFromFile("api_key.txt");
        String htmlDoc = getFileContents("Microsoft.html");
        Document doc = alchemyObj.HTMLGetRankedKeywords(htmlDoc,
"https://en.wikipedia.org/wiki/Microsoft");
        System.out.println(getStringFromDocument(doc));
    }
   private static String getFileContents(String filename)
        throws IOException, FileNotFoundException
    {
        File file = new File(filename);
        StringBuilder contents = new StringBuilder();
        BufferedReader input = new BufferedReader(new FileReader(file));
        try {
            String line = null;
            while ((line = input.readLine()) != null) {
                contents.append(line);
                contents.append(System.getProperty("line.separator"));
            }
        } finally {
            input.close();
        }
        return contents.toString();
    }
```

Mallet Implementation:

1. Set the environment variable and give the path to the mallet directory

2. Make text files for different companies and save them in the mallet folder as follows:

    C drive->mallet->sample-data->web->create folder->save company.txt

3. Type : bin\mallet import-dir --input pathway\to\the\directory\with\the\files --output tutorial.mallet --keep-sequence --remove-stopwords

This runs the mallet program and removes all the stop words from the text files

4. At the command prompt in the MALLET directory, type:

```
bin\mallet train-topics  --input tutorial.mallet
```

This command opens your tutorial.mallet file, and runs the topic model routine on it using only the default settings. As it iterates through the routine, trying to find the best division of words into topics, your command prompt window will fill with output from each run

5. This command is used to save the output in the mallet directory

```
bin\mallet train-topics  --input tutorial.mallet --num-topics 20 --output-state topic-state.gz --output-topic-keys company_keys.txt --output-doc-topics company_compostion.txt
```

The output is a series of paragraphs containing the keywords

Semi structured Dataset:

1. **USPTO Patent.XML**

Business Element	**XML Element**
Company Name	<orgname></orgname>
Patent Title	<invention-title></invention-title>
Patent Abstract	<abstract></abstract>

2. **European Patent XML**

Business Element	**XML Element**
Company Name	<B731><snm>
Patent Title	<B542><p>
Patent Abstract	Not used

Implementation - Data Preparation

Tokenization

Removal of special characters like (,),+, leading spaces in organization names

Example: (Proctor + Gamble)

Hyphenated words in organization names. Example: (Wal-mart)

Stemming

Some company names were URIs. Example: (Amazon.com)

Stop words removal

List of stop words such as should, could, must etc are provided at the end of this document.

Patent counts – Output of MapReduce program

Extract: IBM – International Business Machines

Name	**Patent counts 2011**
international business machiness corporation	2
international business machines	5
international business machines coroporation	1
international business machines incorporated	2
international business machines corporatio	1
international business machines corproation	2
international business machines corporation	6111
oy international business machines ab	1
international business machines coporation	1
Total (we have taken the Max count from the above)	6111

Patent counts – USPTO Statistics – 6148

Name	**Patent counts 2011**
International business machines corporation	6148

Reporting & Visualization

a. Tableau

Tableau was downloaded from www.tableau.com/

Data source was selected to point to Access Database. The individual tables were selected and joined to perform visualizations of relationships between variables.

Dataset Source:

US Patent Data:
https://www.google.com/googlebooks/uspto-patents-grants-text.html
European Patent Data:
https://data.epo.org/expert-services/index-2-2-5.html
Japanese Patent Data:
http://guides.library.uncc.edu/c.php?g=173110&p=1141972
Election Related Data:
https://www.congress.gov/legislation?pageSize=250&q=%7B%22congress%22%3A%5B%22108%22%5D%7D
https://legiscan.com/gaits/search
http://www.cpds-data.org/index.php/data
Company Information:
http://www.crsp.com/products/research-products/crspcompustat-merged-database
Other Data:
http://fortune.com/2013/03/21/the-50-greatest-business-rivalries-of-all-time/
opensecrets.org