The Corporate-Political Nexus

A thesis submitted in partial fulfillment of the requirements for

M.TECH MAJOR PROJECT
by

ABHISHEK AGARWAL

Entry No. 2014MCS2114

AMARTYA CHAUDHURI

Entry No. 2014MCS2117

Under the guidance of

Dr. AADITESHWAR SETH

Department of Computer Science and Engineering,
Indian Institute of Technology Delhi.
January 2016.

Certificate

This is to certify that the thesis titled **The Corporate-Political Nexus** being submitted by **ABHISHEK AGARWAL** & **AMARTYA CHAUDHURI** for the partial fulfilment of **M.Tech Major Project** in **Computer Science & Engineering** is a record of bona fide work carried out by them under my guidance and supervision at the **Department of Computer Science & Engineering**. The work presented in this thesis has not been submitted elsewhere either in part or full, for the award of any other degree or diploma.

Dr. AADITESHWAR SETH
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Abstract

In this project we try to facilitate the study of distribution of power across and within various Indian power institutions by analysing their inter-linkages, family trees, timelines, transactions, etc. to make more sense of the big political and corporate handshakes. In that direction, we envision to construct a system to collect data in this regard from various sources and integrate all of it into a single data store for others to use.

Our aim is to disseminate these findings to the mass society using a crowd-sourced system, and use mass language for such a flow of information. and in this way, enhance governmental accountability and transparency using the notion of Open data and crowdsourcing.

Acknowledgments

We would like to express our heartiest gratitude to our supervisor Dr. Aaditeshwar Seth for guiding this work with utmost interest and scientific rigor. We thank him for setting high standards, giving us freedom to explore multiple facets of the problem and teaching us value of analytical thinking and hard work. We are also grateful to Manoj Kumar, Anirban Sen, and Dipanjan Chakraborty who have helped a great deal by providing technical guidance and support during difficult times and whose suggestions went a long way in making this work a reality. We would also like to thank our family and friends for their love and support.

ABHISHEK AGARWAL
AMARTYA CHAUDHURI

Contents

List of Figures

List of Tables

Chapter 1

Introduction

By common opinion, a democratic society in modern world is a misnomer - an illusion often given to the ”ruled” party in a nation. Two centuries after Lincoln’s definition of democracy, a government-for-the-people sadly is nothing but a collection of handful of population in control of basic resources of a country (natural or artificial). Leaders in military, politics, businesses, sports and arts thus become the actual voice of a country rather than the common mass. The distribution of power in these domains are often hereditary (among close ties in family) instead of elected representatives. Interactions among the persons of these important fields are also common for their effective functioning (or often to safeguard their self benefits.)

In this regard, accountability and transparency in government is one of the key requirements in order to obtain an ideal democratic society. Unfortunately the lack of proper knowledge about the politicians and corporates has led to new forms of ”collective” dictatorship where public rights or voices are rendered ineffective.

To account for this growing problem of opacity we have tried to study the social network of Indian politicians and corporates and disemminate the information to the common public.

Problem Statement- Problem of forming the social networks of Indian Politicians and Corporates, visualizing and analysing them.

1.1 Objective

We intended to complete following tasks through our project-

- To **collect data from semantic web** (and other sources) to form a database (henceforth referred to as **”knowledge base”** / **”knowledge graph”**)

- To create **a neat, structured minimal error data collection** which otherwise is scattered at respective sources.
- To provide **a data mashup from different fields** to further help the academecians, journalists etc.
- To **monitor the top players in Indian society** - mainly in the spheres of politics and businesses in India.
- To **disseminate information to public** which brings about accountability and transparency.
- To seek answers to questions like -
  - *who were the big players in Indian politics and businesses?*
  - *Is there any influence (or possibility of it) of political field by a person in corporate field?*
  - *How important is one politician in a network of politicians (or a businessperson in a business network)?*
  - *Whom does actual power reside in a democracy?*

We believe that through our work, we will be able to show how such system of inter-disciplinary data helps to spread information and find patterns and discover more knowledge.

1.2 Motivation & Related Work

In his book **The Power Elite** [13], C. Wright Mills calls attention to the interwoven interests of the leaders of the military, corporate, and political elements of society and suggests that the ordinary citizen is a relatively powerless subject of manipulation by those entities. His book deals with the power elite in US. But the hierarchy he proposes is more or less the same across all countries. Power rests with the top one percent in an economy. We plan to create a watchdog for that one percent. One interesting list to accompany this direction could be the Forbes list [3] of 147 companies that control everything.

French economist Thomas Piketty in his famous work **Capital in the Twenty-First Century** [14] focuses on wealth and income inequality in Europe and the United States since the beginning of the industrial revolution. He proposes a global system of progressive wealth taxes to help reduce inequality and avoid the vast majority of wealth coming under the control of a tiny minority. We plan to collect, integrate, visualize and open such data for Indian terrain to let data fanatics carry out sych works to understand this inequality.

British writer and historian Patrick French, in his book **India: A Potrait** [10] has stated many such interesting patterns in Indian politics where he argues that almost all of the young Indian politicians in the Indian Parliament are hereditary. Infact, patterns similar to this can be seen over the entire political Indian scene. One can find interesting overlaps, family ties, social links within these power houses. In a survey, **Who owns your media?** [7], we find that even the media is an entity of importance and most of the politicians tend to try pull their strings in this domain. Another interesting case could be Jayant Sinha's family tree [9] and their business holdings. He is the Minister of State for Finance and a Member of Indian Parliament and has links to lot of powerful companies.

Research along the area have been prominent across countries. **Sastry** [15] shows how crime and money play important role in Indian elections. In a related work **Vaishnav** [16] explain why do Indian parties elect criminal candidates and why they win. **Kapur** [11] connects the hidden relationships between politicians and builders. He argues that where elections are costly but accountability mechanisms are weak, politicians often turn to private firms for illicit election finance and that where firms are highly regulated, politicians can exchange policy discretion or regulatory forbearance for bribes and monetary transfers from firms

Works like what we propose have already been done for countries like USA, UK, Chile etc. We have examples like **LittleSis** [4], **Poderopedia** [8] where journalists, developers, analysts came together to put up profiles of important entities, institutions of the society and highlighted the connections between them. Littlesis (opposite of Big brother) in one hand exists in USA from the political and economical data available there. Poderopedia is a similar site in Chile. These sites feature separate pages of people in power in USA, their connections to different institutions and other entities , work history, visualizations of the connections to educate masses etc. Other than producing awareness to people about the corporate- political connections, these sites also allow public to register and collaborate in data entry processes and has an API system to promote further use of their data for research purposes.

Such system in absence of digital data/ structured data and other human factors is difficult in India. But various local and national initiatives have been started. **Association for democratic Reforms** [1] for example has sites like **Myneta** [6] to disseminate information about political leaders of India.

Our vision is to produce a system similar in lines to the websites embedded with the power to query interesting connections, find interesting visualizations, and help raise suspicious issues.

1.3 Thesis Overview

The rest of the thesis has been divided into following -

1. **Social Network Creation. -**
2. **Design of Power Elites Web App.**
3. **Pattern Analysis**

Chapter 2

System Design

2.1 High Level Design

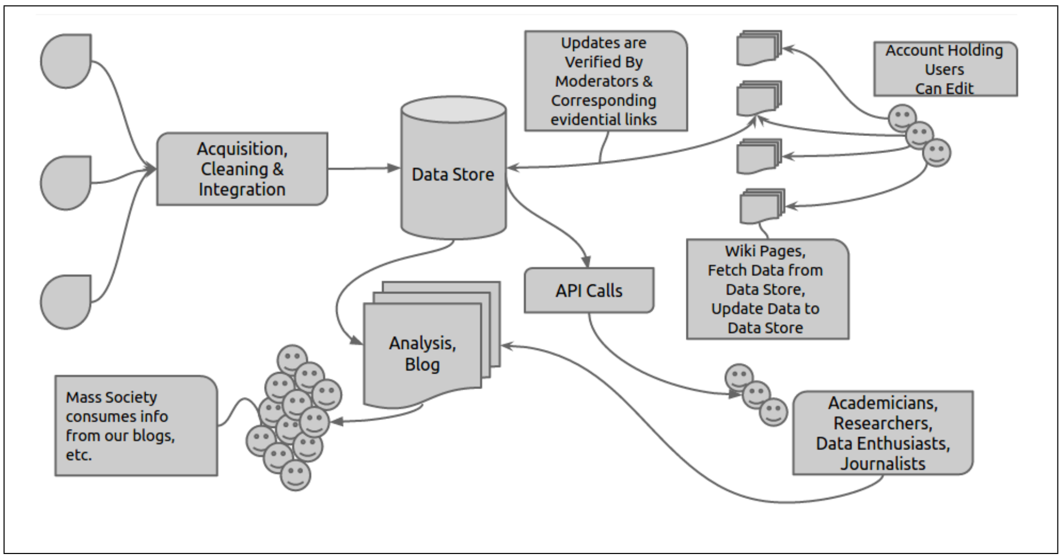

Updates are Verified By Moderators & Corresponding evidential links
Account Holding Users Can Edit
Acquisition, Cleaning & Integration
Data Store
Wiki Pages, Fetch Data from Data Store, Update Data to Data Store
API Calls
Analysis, Blog
Mass Society consumes info from our blogs, etc.
Academicians, Researchers, Data Enthusiasts, Journalists

Figure 2.1: Big Picture

Above is the generic high level diagram of the entire system as we deduced from our study of the websites above. We have assumed three main types of users of here:

1. **General public** - Use the system for information and news. May also contribute towards data entry.
2. **Researchers & Academicians** - Use the site for using social network studies.
3. **Journalists (Media persons)** - Use the data to frame their news stories.

And keeping these in mind, the system accounts for following functions:

- **Data acquisition** - As getting structured data is often difficult system should have a mechanism to automate collection of data from various sources over internet.
- **Data Store** - There should be a central repository for whatever data collected. This repo will store the data in structured form. Care should be taken to keep its integrity, durability and non-redundancy.
- **Data Verification** - As the data is sensitive and important, provisions for verification for the input data has to be taken care of. This involves manual labor and system should incorporate this in the entire process.
- **Data Visualization** - A portal for the public display of data (in tables, visualizations etc.).
- **APIs** - To provide our data for use with other applications.

2.2 Technical design

Accordingly we have incorporated the functionalities in the following way:

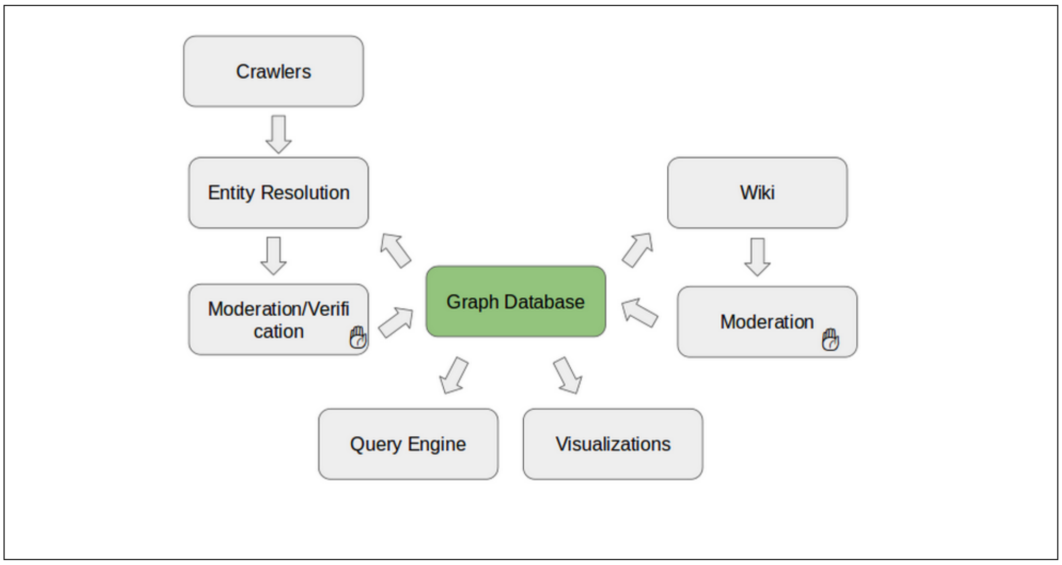


Figure 2.2: Proposed System

Components include:

- **Crawlers-Cleaner-Resolver** - As a part of the data acquisition module, the crawler crawls specific websites and scrapes data in format specified in a static file. Cleaner and resolver works on the scraped data. The cleaner makes the data more structured by keeping all data in specific format, discarding missing values etc. The resolver acts on the data to remove all duplicate entries (and merge similar entries) so to keep the data non redundant as possible.
- **Verifier**- The data coming from the crawler after resolution is verified by human moderator. Any to-be-updated information is first human moderated. Any new data is then fed to the graph database.
- **Graph database** - Acts as the data store for the system storing structured data with nodes as entities and edges as relationships.
- **Web portal(Query Engine, Visualization, Wiki)** - This is the final product that is directly visible to the users. All visualizations (graphical, tabular) are done here. It also has an wiki interface for an end user to add more entities/relations to the graph database. The portal also includes a query/search engine to show results as per specific user queries.
- **API** - Registered users can use this to read in data from the datastore in their application.

Chapter 3

System Pipeline

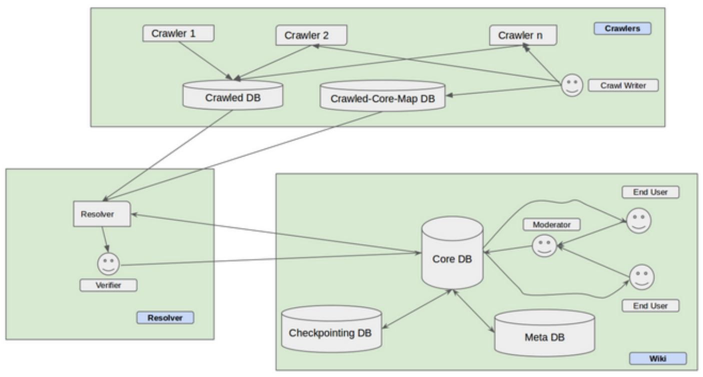


Figure 3.1: System pipeline

The total pipeline of the system can be seen in figure above. Overall, we have decomposed the system into 3 cohesive modules. In order of the data flow, they are -

Crawlers → Resolver → Wiki

3.1 Crawlers

This is the data acquisition module we have designed. Internally this module consists of the following components:

3.1.1 Crawler(s)

- These are basically python scrapy/beautiful soup/selenium scripts to crawl particular websites and srape data from them. The nature of the data (i.e - data types, labels) is mentioned in the ***Crawled-Core-Map*** database.

3.1.2 Crawled-Core-Map database

- Contains the type of the data, its name(or label) to be scraped from a website. This mapping information is used by crawlers to produce output data of specific type,label. This map along with the specific crawlers are written by the developers (***Crawl-Writers***)

3.1.3 Crawled database

- The raw output from the crawlers is written here. The schema of this database is largely dictated by the ***Crawled-Core-Map*** database. The data is then read from this database into the ***Resolver*** module.

3.1.4 Crawl Writers

- They are the human components of this module. These persons are developers who write specific scripts to scrape desired websites. They are also responsible for maintaining the data schema in ***Crawled-Core-Map*** database.

3.2 Resolver

This module works to fine tune the data scraped by the crawlers. All data collected thus far are mostly unformatted with duplicate or missing values. The function of this module is to process such data into a form suitable for the database. Its constituents are:

3.2.1 Resolver

- It takes the data from the Crawled database as input. The resolver can be further divided into two parts -

- **Cleaner** - The cleaner does the text normalization before the actual process for resolving begins. This includes out-of-format data, capitalization, missing-value issues which if not dealt with may cause the resolver to give low accuracy.
- **Resolver/Duplicator** - Function of resolver is two-fold. Firstly, it checks any duplicate records in the incoming data (i.e. - data from the crawled database) and removes if any. Then, it resolves the entries from the crawled data with that of the core data. And outputs all possible matchings to the verifier for verification/tagging.

3.2.2 Verifier

The authenticity of the data should be checked before it can be inserted into the ***Core Database***. The function of the verifier module and thus the human moderator is to tag the records outputted by the resolver module. These tagged records are the ones which are finally added to the ***Core Database***.

3.3 Wiki

This module is the web portal that is directly accessible to the end users. It consists of the interfaces that allow user to view, add, delete entities and their relationships interactively.

3.3.1 Core Database

This is the main data store of the system. Care has been taken to ensure that whatever data goes in is redundant, free of noise and authenticated. **Neo4j GraphDB** [12] is used in the backend for this.

- **Why Graph database preferred here?**
  Lot of brainstorming went in deciding to use Neo4j graph database. This is because a graph database stores the relations of records in the physical layer (unlike relational database) which makes faster retrieval

of connections of entities without joins. And most of the analysis in social networks involves reading in the relationships/connections between entities. Hence query results can be produced faster here without any complicated joins.

3.3.2 Meta Database

This contains the description of the data(format, source, type) being inserted in the database. This is especially useful when the data is authenticated against real-world information, so that every ounce of data in the core database is accounted for.

3.3.3 Checkpointing Database

Without a checkpoint, a Wiki is un-achievable. With every new update, a log of all changes is stored in this database. This is later used to roll back to a previous state to undo any new updates that occured.

3.3.4 Web Server

Background server that hosts the web application currently has 3 major functions.

- **Wiki + Visualization** - The basic functionality of the web application is to provide the users with profiles and connections of organizations and personas which they can edit. The server also maintains a mechanism for checkpointing any data received.
- **Query Engine** - To support the queries required for analysis of the network the server implements a query engine which takes queries of specific pattern and return results in tabular or visual format. This is achievable by Cypher query language which facilitates inquiring graph-like queries over Neo4j. These queries would go on the lines like: How are two entities connected? What is the shortest path between two entities? How far an influence of an entity goes over the graph?

Name	Age	Sex

Table 3.1: Sample data formats

Entity no.	Graph label	Graph props	Mysql props	Graph resolve props	Mysql resolve props
1	person	name,age,sex	name,age,sex	name	name

Table 3.2: Sample record of Crawled-Core-map database

- **Read API** - External applications can give GET requests to read data in json format. This aids research and analysis by end users.

3.3.5 Others

Human components in the system - *Users* and *Moderators.* Users include the wiki-users who edit the Wiki to enter any new/updated info they have. They have to provide an evidential link for any information they commit.
The job of moderators is to verify records entered by users in the Wiki. They can be experts in their domain, they need to cross-verify from trusted sources.

3.4 Example

Here we describe how the Verifier and Resolver actually work with a working example.

3.4.1 Verifier + Resolver

Let us say that the crawler crawls data and saves it in a format like in Table - 3.1 Also the crawler has an accompanying map-data information which looks like - 3.2
graph-props column have one to one order wise mapping with *mysql-props* column.
graph-resolve-props column have one to one order wise mapping with *mysql-resolve-props* column.

So, basically in the crawled data, each row represents a node with label :person and attributes (name, age, sex) in the core database.

1. Now, when a verifier logs-in a row from the 3.1 is fetched, then 3.2 is used to frame a query to search an entity with the name like in the row.
2. Matching nodes are suggested after the query to the verifier.
3. If verifier selects one of the propsed nodes, the resolved node is updated.
4. Else if the verifier doesn't find any matching node, a new node correspoding to the selected row is created and inserted in the core database.

3.4.2 Wiki + Visualizations

Profiles

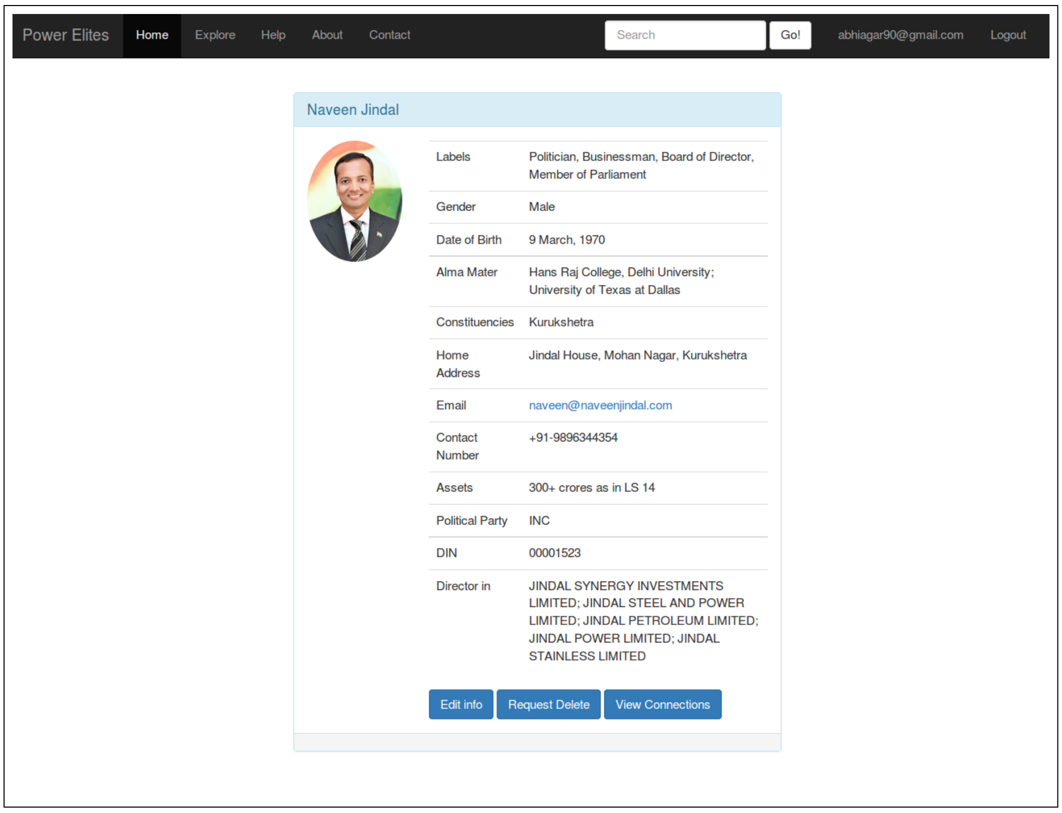
Following is a typical profile in the Wiki-

The figure above shows the Wiki page for industrialist Naveen Jindal containing information about him. It also contains interfaces for any registered user to edit as happens in a Wiki. It also allows the user to show connections of the person.

Connections (Visualizations)

The figure above shows the connections for Minister of State for Finance Jayant Sinha. His connections include corporates firms like the **Aditya Birla Group** and **Pacific Paradigm Advisors** .

Other popular influence networks that our system shows is that of *Gandhi Family and their Corporate linkups*, *Jaydev Galla with his company with a large asset value*, *Kamal Nath with Moser Baer*, *Ravi Shankar Prasad with News 24 channel.*



Naveen Jindal

Labels	Politician, Businessman, Board of Director, Member of Parliament
Gender	Male
Date of Birth	9 March, 1970
Alma Mater	Hans Raj College, Delhi University; University of Texas at Dallas
Constituencies	Kurukshetra
Home Address	Jindal House, Mohan Nagar, Kurukshetra
Email	naveen@naveenjindal.com
Contact Number	+91-9896344354
Assets	300+ crores as in LS 14
Political Party	INC
DIN	00001523
Director in	JINDAL SYNERGY INVESTMENTS LIMITED; JINDAL STEEL AND POWER LIMITED; JINDAL PETROLEUM LIMITED; JINDAL POWER LIMITED; JINDAL STAINLESS LIMITED

Figure 3.2: Naveen Jindal Profile

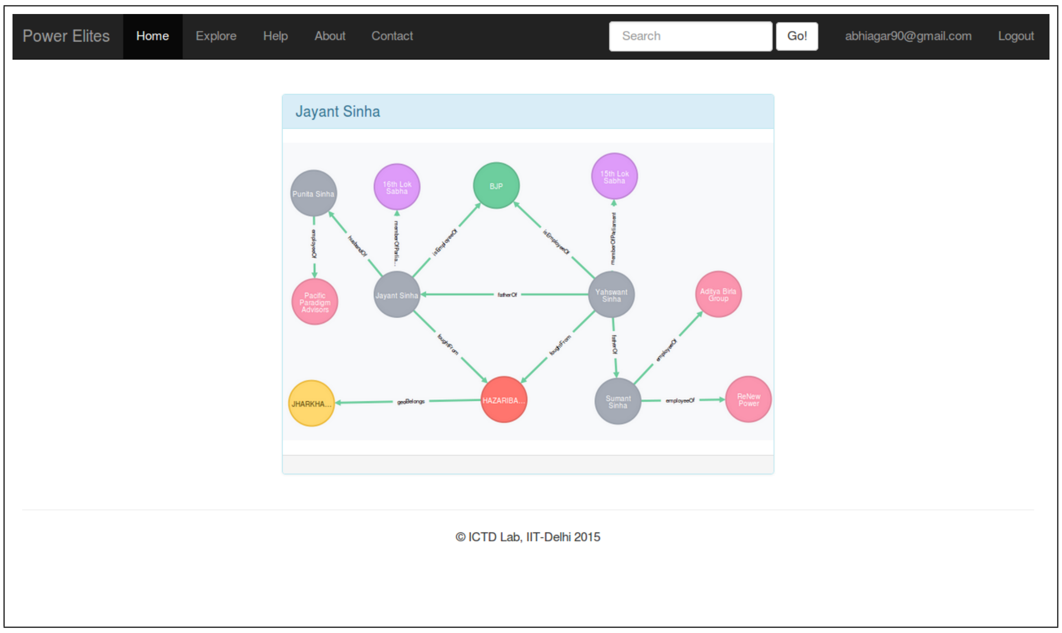


Figure 3.3: Jayant Sinha Connections

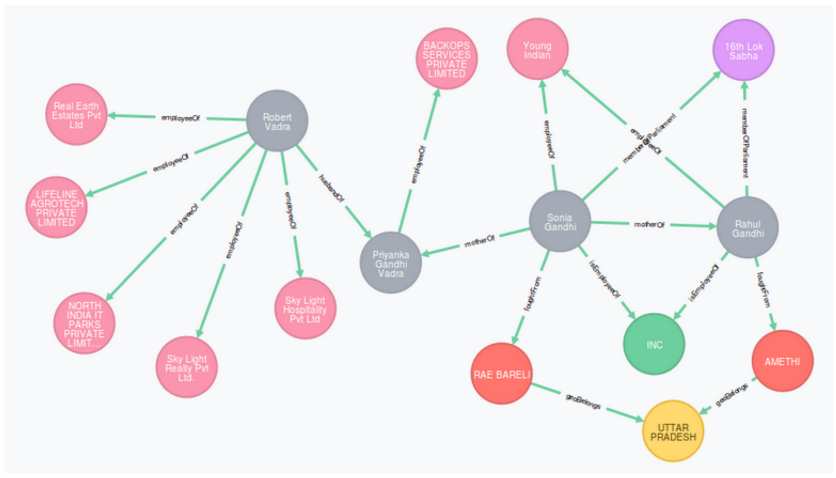


Figure 3.4: Gandhi family

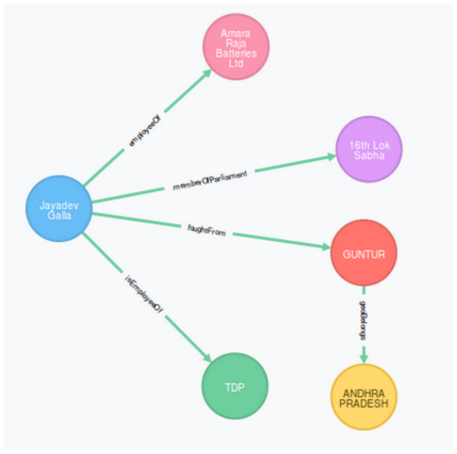


Figure 3.5: Jaydev Galla Connections

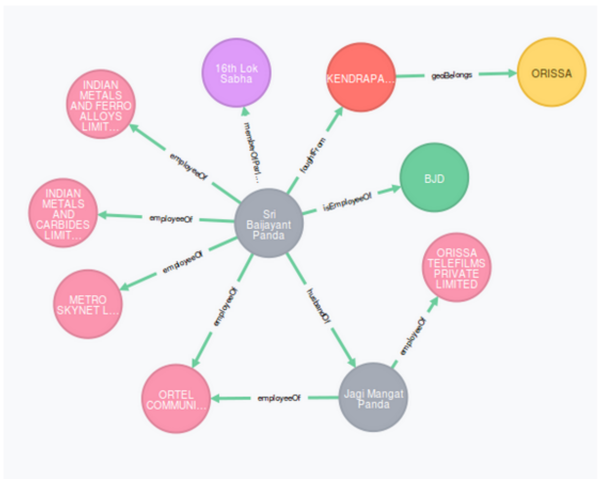


Figure 3.6: Jay Panda Connections

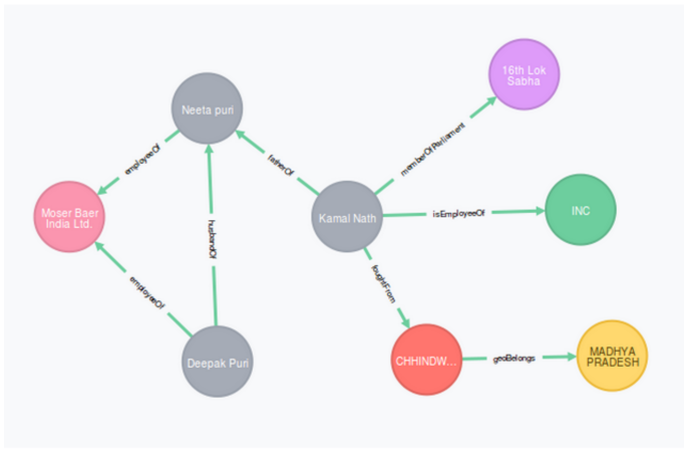


Figure 3.7: Kamal Nath Connections

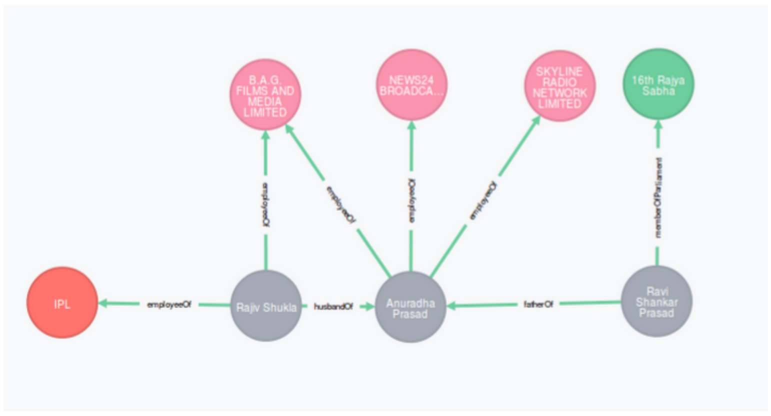


Figure 3.8: Ravi Shankar Connections

Chapter 4

Challenges

4.1 Datasets acquisition

Biggest challenge in mining political, bureaucratic, corporate data is the difficulty in getting relevant sources and structured information.

For our initial use cases we began by using political data from **MyNeta** [6]. We plan to enrich this data from LokSabha and Rajya Sabha archives. For corporate data, we used publicly available information from **CompanyWiki** [2] and **Ministry of Corporate Affairs** [5]. As the system progresses we plan to include retired IAS/IPS officers, PPP data, family data of politicians, donations in our growing dataset. We already have about 1000+ politicians, 60000+ business people, 90000+ companies in our current core DB.

4.2 Data Modelling

After getting some data the challenge was to model the data for the graph database. This would ensure listing of all possible entities and relationships of the system. Care was taken to form such relationships so that in the long run, queries supported by the system take optimum time to produce results. The prominent entites(nodes) and relationships(links) modelled so far:

Nodes/Entities-
(**Person** (uuid, name, address-location, DOB))
(**Politician** (uuid, name, mynetaid-electionname-year, constituency))
(**Alias** (uuid,name,context))
(**Party**(uuid,name,type:*"national,state,regional"*,start-year,HQ))
(**Company**(uuid,cin,name,location,income,expenditure,profit,value))

(**IAS** (uuid,name,DOJ,year,IAS-ID,posting-location))
(**Employee**(uuid,din,name,designation))
(**Govt-Body**(uuid,name,location))

Relationships/Links-
(**Politician**)→(**member-of**(id,type,years)→(**Party**)
(**Politician**)→(**member-of**(id,type,years)→(**Govt-Body**)
(**IAS**)→(**member-of**(id,type,years)→(**Govt-Body**)
(**Employee**)→employee-of(id,designation,years)→(**Company**)
(**Company**)→donated(id,cin,party,amt,year)→(**Party**)
(**Person**)→family(id,is-biological→(**Person**)
(**Person**)→profession(id,type)→(**Politician**)
(**Person**)→profession(id,type)→(**Corporate**)
(**Person**)→profession(id,type)→(**IAS**)
(**Person**)→aka(id)→(**Alias**)

4.3 Latency and Optimizations

Although implementation of basic system is ready, its performance suffers when single GET request to the Web server calls for multiple read requests to the core database. We are planning to alleviate this problem by indexing the database to optimize search time, forking multiple threads.

Chapter 5

Epilogue

5.1 Future Work

The constant aim would be to push as much data as possible and stablize the system as soon as possible. The query engine is very basic as of now. The further plan is to give users a platform to give structured queries to the system. We also plan to extend our sources of data, use information extraction algorithms to extract relationships from blogs, newspaper articles where the data is totally unstructured. The UI/accessibility of the system needs to be improved so that the verifiers, moderators, end-users can access the related pages/info readliy.

5.2 Conclusion

The system should be able to answer such questions as we move along:

- Preference between corporate donations to political parties.
- Preference between donation source locations and political parties.
- Ex-bureaucrats in corporate jobs and their political ties.
- Identification of shell companies/fraud companies in donation transactions and investments.
- Family trees of political candidates, corporate giants.

We are still at a very nascent stage. Trying to learn in our work and hopeful to create something beneficial for the society in future.

Chapter 6

Constructing the Social network

The fuel of any social network or a knowledge base is the data it represents. The problem in constructing a linked-data system such as this one is always the data and the entropy it brings with itself. The challenges are always the usual ones - unavailability of data, noise in the collected data, no authentic source, and many-a-times no structure in the data. Even if we are successful in collecting and cleaning the linked data we want, the way we go about integrating all this variety of information in a single data store is itself an another challenge.

The choice of data store matters here the most because one would like to query the data and unearth interesting relations between participating entities. Henceforth, a person or an organization will be called an entity. These entities would be linked to each other by certain edges/links just like in a graph which we will normally call relations. All these challenges are elevated manifold when one wishes to resolve entities from different datasets into a single unified entity.

Here we describe in order our choice of data storage, our core data model for the linked data, data collection practices and data sources, data integration methodology, and finally the necessary evil in such a system entity resolution.

6.1 Everything is a graph

We begin by describing how we are going to actually store any kind of linked data we get and why our approach is a sensible one. We describe here how our core data and core data model looks like. It has to be stated at the onset that care has been taken to ensure that whatever data goes in our core data modal is non-redundant, free of noise and verified. As already stated, our goal is to construct a social network between politicians, companies, entrepreneurs, military personnel, bureaucrats, political parties, universities, movie actors, and any important entity one can think of in the usual power hierarchy. Also, we want to be able to model all kinds of relationships that can exist between these entities with ease: family-links, donation-links, director-links, ownership-links, subsidiary-links, etc.

Practically anything connected can be represented by a graph(or a hypergraph to be exact, refer our LIMITATIONS section). We live in a connected world. There are no isolated pieces of information, but rich, connected domains all around us. Thus, it makes sense to model our core data as a large inter-connected graph where every node is an entity and every edge represents a link between two of them.

A graph is composed of two elements: a node and a relationship. Each node represents an entity (a person, place, thing, category or other piece of data), and each relationship represents how two nodes are associated. This general-purpose structure allows you to model all kinds of scenarios from a system of roads, to a network of devices, to a populations medical history or anything else defined by relationships. [https://neo4j.com/developer/graph-database/]

Before beginning to describe how we achieve the above, it is important to understand that even traditional SQL tables are a connected piece of information, and can be modeled using a graph.

What we provide is therefore a graph to the user where he fits any relevant connected data he has. The choice of the data model is clear, but two problems remain - how we are going to store our graph's interconnected data, and how do we query it efficiently for digging out interesting relationships. Our next two sections discuss the same.

6.1.1 Neo4j

Neo4j [https://neo4j.com/] is our choice of data storage. It is one of the leading JVM based NoSQL graph databases. We build our knowledge base on it as a graph and thus Neo4j is at the center of our entire system. We chose Neo4j because of the following reasons[https://neo4j.com/developer/graph-database/], and we have found it be a non-separable asset to our use cases.

1. Only a database that embraces relationships as a core aspect of its data model is able to store, process, and query connections efficiently. While other databases compute relationships expensively at query time, a graph database stores connections as first class citizens, readily available for any join-like navigation operation. Accessing those already persistent connections is an efficient, constant-time operation and allows you to quickly traverse millions of connections per second per core. [https://neo4j.com/developer/graph-database/]

2. Graph databases are designed to mimic the most natural way we tend to model data the same way you would map it all out on a whiteboard. Your collection of circles, boxes, lines and arrows is in essence already a graph. [https://neo4j.com/blog/graph-data-modeling-success/]

3. In Neo4j, everything is stored in form of either an edge, a node or an attribute. Each node and edge can have any number of attributes. Both the nodes and edges can be labeled. Labels can be used to narrow searches. [http://neo4j.com/docs/1.8.3/indexing.html] Neo4j is very easy to learn and adapt. It's object property model is very intuitive and anything can be modeled on a white board in the form of nodes and edges.

4. Constant time traversals for relationships in the graph both in depth and in breadth due to efficient representation of nodes and relationships [https://neo4j.com/developer/graph-database/]

5. All relationships in Neo4j are equally important and fast, making it possible to materialize and use new relationships later on to shortcut and

speed up the domain data when new needs arise [https://neo4j.com/developer/graph-database/]

6. Compact storage and memory caching for graphs, resulting in efficient scale-up and billions of nodes in one database on moderate hardware [https://neo4j.com/developer/graph-database/]

7. It's NoSQL help us in modeling the varied data from different sources we have collected.

8. The cypher query language provided by Neo4j helps in querying the connected data very easily and is very powerful. Also, the Neo4j browser client provided by Neo4j is vary good for visualizing the results of the cypher queries.

That said, it is important to note that Google uses Cayley - an open source graph database - to power it's google's knowledge graph. [citation: https://github.com/goog

6.1.2 Property Graph Model Explained

Let us dive into some examples which explain how me model our core data in Neo4j using it's property graph model. [https://neo4j.com/developer/graph-database/]

1. The property graph contains connected entities (the nodes) which can hold any number of attributes (key-value-pairs). What this means for us is: a person node can have different attributes his date-of-birth, address, email, sex, etc.

2. Nodes can be tagged with labels representing their different roles in our domain. That said, this unique thing about Neo4j helps us in specifying IS-A relationships via multiple labels on a single Neo4j node. Thus, a node can be labeled as a person, politician, businessman at the same time. The same information is very hard to model in traditional SQL databases.

3. In addition to contextualizing node and relationship properties, labels may also serve to attach meta-data index or constraint information to certain nodes.

4. Relationships provide directed, named semantically relevant connections between two node-entities. A relationship always has a direction, a type, a start node, and an end node. This model is exactly the same as directed edges in a traditional graph structure. Although they are directed, relationships can always be navigated regardless of direction.

5. A relationship can also be labeled by a single label specifying what kind of a relationship exists between two nodes. Thus, if a politician is 'son-of' a businessman we can connect the two nodes by an edge attributed with such a label.

6. Just like a node, a relationship can also hold any number of attributes(key-value pairs). In most cases, relationships have quantitative properties, such as weights, costs, distances, ratings, time intervals, or strengths. But for our use case, one can think of an example relationship as : a particular 'person'(node) 'donated'(relationship-type) 'x amount of money at y date'(relationship properties) to(directed edge) a particular 'politician'(node).

7. As relationships are stored efficiently, two nodes can share any number or type of relationships without sacrificing performance.

An example could be the image [image 1]

6.1.3 Cypher

Cypher is a declarative graph query language for the graph database Neo4j that allows for expressive and efficient querying and updating of the graph store. Cypher is a relatively simple but still very powerful language. Very complicated database queries can easily be expressed through Cypher. This allows users to focus on their domain instead of getting lost in database access.[http://docs.neo4j.org/chunked/stable/cypher-introduction.html]

A node is represented like this: (:person:politician name:'Narendra Modi',sex:'M')
A relation is represented like this: (startnode)-[:relatedto kind:'childof']-¿(endnode) The structure here is self explanatory and can be further explored by reading Neo4j manual. A sample image[2] here describes the image for cypher.

Querying in cypher is as easy as thinking about how you traverse a graph. Here is a simple query to return all the relationships that start from or end at node 'Naveen Jindal' of type politician.

MATCH (jindal:politician name:'Naveen Jindal')-[anyrelation]-(anynode) RETURN anyrelation

Interested readers can be direct here to read more about cypher in Neo4j's manual: [https://neo4j.com/docs/developer-manual/current/]

6.1.4 Our restricted property graph model

We had to specify some ground rules to model our data for imposing uniformity on varied data that is being fed into the system. Thus, our property data model follows the rules stated below.

1. As allowed by Neo4j - a node can have more than one label, a relation always has only one label.
2. Every node and a relation has a unique uuid/relid
3. All nodes are labeled 'entity' by default. To make anonymous queries on nodes easier.
4. All living or dead people are labeled 'person'.
5. Any person connected to a company as a director/owner is labeled 'businessperson'.
6. All operating units are labeled as 'organization'.

7. All companies is labeled as 'company'
8. A political party is labeled as both 'organization' and 'company' alongside 'entity'
9. Other self-explanatory labels for nodes in current core data are: city, state, constituency.
10. Every 'entity' will have to have a name property. An aliases property - a neo4j array - helps in keeping track of different names of an entity.
11. An 'entity' can have any number of properties, but these properties if present are validated: 'startdate'(int), 'enddate'(int), 'iscurrent'(boolean). For a person a 'startdate' represents his date-of-birth, for a company 'startdate' represents it's incorporation-date. The 'iscurrent' property can help us in tracking if the 'person' is dead or the 'company' is not active.
12. startdate and enddate are timestamps since epoch - so any date-of-birth or company's incorporation-date will have to be converted to a particular format before pushing to the system.
13. All relationships have to have a property bidirectional - to explicitly specify if the relationship goes both ways. This had to be done as Neo4j edges are always directed, though they can be queries without directions.
14. Some labels for relationships in current core data are: relatedto, worksin, geoBelongs.
15. An entity can have any number of properties, but these properties if present are validated: startdate(int), enddate(int), iscurrent(boolean). For a person a startdate represents his date-of-birth, for a company startdate represents it's incorporation-date. The iscurrent property can help us in tracking if the person is dead or the company is not active.
16. All the other meta-info will be stored in a separate sql database that will help in mapping any data point change to its source and the user

who allowed that change. this is better explained in the provenance section in system design chapter.

We list some images here that better describe the present data model. [all the model images]

6.2 Data collection and description

Since the aim was to to build a social network for all the Indian power houses, we needed to identify and collect as much varied data as possible. It is only when different kind of data mingle with each other in the system that we can hope to see some hidden patterns or dig out interesting insights. All the visualizations that we obtained from this data is shown in the analysis section. [citation]

We needed to search for alternate data sources that might help the ongoing work. We have looked for datasets spanning company details, board of directors, Lok-Sabha MPs, subsidiaries, banks, and any other relevant political-corporate data.

6.2.1 Challenges

1. The biggest challenge in data collection is that data for Indian context is not easily available. Open data initiatives like data.gov.in have initiated hope for data enthusiasts but there is still a long way to go. [citation]

2. No central place for any category of target data, so one would have to look for multiple sources. Instead, in a way, we are striving to create such a central place in the present work.

3. Since data has to be integrated from variety of source points, the credibility of a data item listed can be established strongly but with the added evil of entity resolution - which has been explained in a later section [CITATion section]

4. Mostly, no unique identifiers for entities appear in datasets that are publicly available. There is a possibility of duplication of data, this is where the ER work comes in handy. [citation]

5. Most data contained noise, or erroneous details at times. This sometimes is intentional on the data entry operator's part, sometimes it's just a naive mistake. For example: name for some people was wrongly spelt in some publicly available websites.

6. Data was missing for some datasets. For example, in the affidavit that the politicians have to fill before elections, or in the data collected from companywiki.

7. Names are the biggest problems in such a work. People use titles like Mr., Shri, Shriman, Late, Lt., Mrs., Kumari, Kumar which again add more complexity to the problem.

8. There is no standard data format for dates even across government departments. Similarly for the address - there is no standard format to extract out relevant information.

9. Most of the government sites maintain significant data on-line but hidden in a complex web of links and they love uploading data in PDFs!

6.3 Tools and Methods

We describe here tools that we picked up along the way for data collection and data cleaning. It is important to note that no single tool can be a cure for all the data-sources. In fact, what we have learned in due course of time is that different kind of data sources require different approach.

[todo]

1. A naive method that we experimented during the initial work is to fetch a source url through python's urllib library, and then process it using BeautifulSoup.

2. selenium
3. scrapy
4. manual at times: if data less and useful for connections.
5. data collected is preprocessed using pandas library, nameparser, wote our own algorithms for converting date-of-birth to startdate format (int) - timestamp epoch

6.4 Corporate data

The challenges in collecting any corporate data are:

1. Getting government provided unique identification numbers for the company
2. Getting government provided director identification numbers for their directors
3. Linking these two IDS.
4. Getting the list of all companies over the time with their Ids of course.
5. Getting the list of subsidiaries for each company/group. (The toughest job)

[todo] We have been able to tackle the first four challenges to a commendable extent but the last one. A related work is Parag's [citation]. We first describe the sources we have encountered, and finally how we collected our corpus of 60000 directors and 90000 companies.

MCA

[citation mca]

Being the official government source, this is the most authentic data source we have found for obtaining the list of companies and LLPs (Limited Liability Partnerships) operating in India. It gives us all the companies registered year-wise before and after 1980 till date in the form of pdfs with their CIN/LLPIN (Corporate Identity Number/Limited Liability Partnership Identification Number) which is unique to a company/LLP. The only downside as always with any pdf data is a lengthy parsing job.

MCA, with its search engine, is amazing! Armed with the CIN, we can obtain a lot more information for a company from the MCA site itself. We get the type of the company, the category of the company, the headquarters, its market capital, date of incorporation and the charges it is facing. Not only that we also obtain the signatories of a company from the MCA site as well i.e. their board of their directors, with their unique DIN(Director Identity Number). This can again be useful in entity resolution. (This is a new feature they introduced).

[todo: mca problems crawl: why not]

capitaline database

[ciation cpaitaline] One of the main challenges was to figure out the subsidiaries of a firm. This would complete the picture of the company network. Alas, finding the subsidiaries is not an easy task. We could find just one source after running around all over the internet: Capitaline. However, the data seems to be spotty at best. We could not find details for a lot of the listed subsidiaries.

The problem with capital line is multifold: No CIN, LLPIN. Very bad interface to crawl but nonetheless crawlable. It has no single source of the companies. And subsidiaries information extraction would just reveal names of subsidiaries, no unique IDs! There is no hyperlink from the subsidiaries name to any company page.

Parag has already collectd this data [citation] todo

6.4.1 companywiki

[citation compnywiki] [todo] advantages linked data reverse search from both directions seed parag dataset three days crawl selenium dataset stats linked data our core base

6.5 Political data

ls crawl from ls site - images too -

Our task in this domain was to find personal information of MPs and their dependents. So we went looking in Wikipedia of various political members and their families. But Wikipedia is highly unstructured and crawling, parsing Wikipedia even with professional extractors is a tedious task.

We started with these problem statements in our mind:

How to get info on all 16 Lok Sabhas MPs? How to get the family information of MPs?

MyNeta

they have done a tremendous work in digitizing this data courtesy data description problems - missing fields, name format info about constituency and states info about assets and liabilities info about criminal cases mynetaid for a particular election helps in resolution from myneta dataset, kept as internal prop ls 2009 and ls 2014 data

Lok Sabha Official data

describe how we got the data name and address problem why we choose the data gain when we had myneta- family links of politicians

Rajya Sabha Official data

describe how we got the data name and address problem why we choose the data family trees of politicians

6.6 Family data

why to see if somehow the powerhouses are connected known examples already include naveen jindal, jayadev galla, etc. chidambram and his family in news already, etc [citation: https://en.wikipedia.org/wiki/Category:Business_families_of_Inc
[citation: https://en.wikipedia.org/wiki/Political_families_of_India]

why integration crawl db desscription —– diff type of data – so json — diff labelled of data– examples— amshups —diff type of connected data —- why json — ao all follow the same langauge we speak...many crawlers ..many data sources...format of the json...to show connections...nodes and relationships between them...the description of api and validations in system section....

data collection should be able to update as and when required–cleaning names etc. indian data, open data, sources, sometime manual, scripts - selenium, beuatiful soup, scrapy...comparsion among the three....(for a side study - application – article collection) mca, loksbha, rajyasbha, manual connections from wikipedia, cin, din what all fields in data, etc. parag dataset, seed concept, intersting insights can be seen in the analysis section - after and vefore mashups... quickcompany and other sources...images of politicians

current core dataset contains nodes, and rels, all the agg info in analysis section

Bibliography

[1] Association of democratic reforms - non-governmental organization for electoral and political reforms. http://www.adrindia.org/. 2015.

[2] Companywiki - find information about indian companies. http://www.companywiki.in/. 2015.

[3] Forbes - 147 companies that control everything. http://bit.ly/1O4BIgZ. 2015.

[4] Littlesis - profiling the powers that be. http://littlesis.org/. 2015.

[5] Ministry of corporate affairs - gateway to all services, guidance, and other corporate affairs related information in india. http://www.mca.gov.in/. 2015.

[6] Myneta - criminal and financial open data on politcians. http://myneta.info/. 2015.

[7] Newslaundry - who owns your media? http://bit.ly/1eY5Bj2. 2015.

[8] Poderopedia - collaborative platform that helps understand the relationships among important people, companies and organizations for chile. http://www.poderopedia.org/. 2015.

[9] Sinha family tree. http://bit.ly/1O4BIgZ. 2015.

[10] Patrick French. *India: A portrait.* Vintage, 2011.

[11] Devesh Kapur and Milan Vaishnav. Quid pro quo: Builders, politicians, and election finance in india. *Center for Global Development Working Paper*, (276), 2011.

[12] Justin J Miller. Graph database applications and concepts with neo4j. In *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA March 23rd-24th*, 2013.

[13] C Wright Mills. *The power elite.* Oxford University Press, 1999.

[14] Thomas Piketty. Capital in the 21st century. *Cambridge: Harvard Uni*, 2014.

[15] Trilochan Sastry. Towards decriminalisation of elections and politics. *Economic and Political Weekly*, 4, 2014.

[16] Milan Vaishnav. *The Merits of Money and Muscle: Essays on Criminality, Elections and Democracy in India*. PhD thesis, Columbia University, 2012.