SKEWER: SENTIMENT KNOWLEDGE EXTRACTION WITH ENTITY RECOGNITION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Christopher Wu

June 2016

COMMITTEE MEMBERSHIP

TITLE:	SKEWER: Sentiment Knowledge Extraction With Entity Recognition
AUTHOR:	Christopher Wu
DATE SUBMITTED:	June 2016
COMMITTEE CHAIR:	Alexander Dekhtyar, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Foaad Khosmood, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Franz Kurfess, Ph.D. Professor of Computer Science

ABSTRACT

SKEWER: Sentiment Knowledge Extraction With Entity Recognition

Christopher Wu

The California state legislature introduces approximately 5,000 new bills each legislative session. While the legislative hearings are recorded on video, the recordings are not easily accessible to the public. The lack of official transcripts or summaries also increases the effort required to gain meaningful insight from those recordings. Therefore, the news media and the general population are largely oblivious to what transpires during legislative sessions.

Digital Democracy, a project started by the Cal Poly Institute for Advanced Technology and Public Policy, is an online platform created to bring transparency to the California legislature. It features a searchable database of state legislative committee hearings, with each hearing accompanied by a transcript that was generated by an internal transcription tool.

This thesis presents SKEWER, a pipeline for building a spoken-word knowledge graph from those transcripts. SKEWER utilizes a number of natural language processing tools to extract named entities, phrases, and sentiments from the transcript texts and aggregates the results of those tools into a graph database. The resulting graph can be queried to discover knowledge regarding the positions of legislators, lobbyists, and the general public towards specific bills or topics, and how those positions are expressed in committee hearings. Several case studies are presented to illustrate the new knowledge that can be acquired from the knowledge graph.

ACKNOWLEDGMENTS

Thanks to:

- My advisor, Alexander Dekhtyar, for his invaluable support and guidance
- My committee members, Foaad Khosmood and Franz Kurfess, for their indispensable advice and feedback
- Carmen Dang, for her excellent work in integrating AlchemyAPI with SKEWER
- Andrew Wang, for letting me use his TONGS sentiment engine
- My family, for their unwavering love and support

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

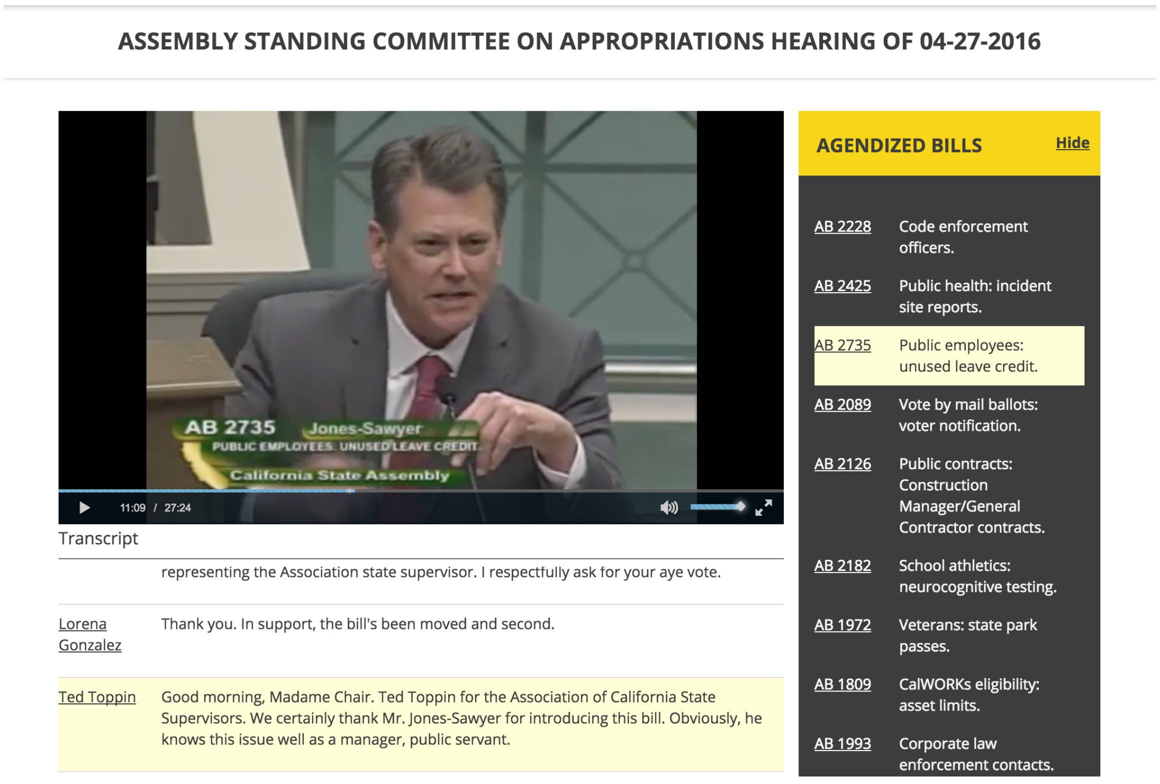
INTRODUCTION

Legislative hearings within the federal government are readily available to the general public through network channels like C-SPAN. The same cannot be said for the equally important hearings in state governments, whose laws have as much of an impact on citizens as federal laws. The record-keeping for state legislative hearings consists of raw video recordings that are not easily accessible to the general public. Also missing from these recordings is supplemental data so that the public can readily understand what is happening in these recordings. This results in a harmful lack of awareness regarding the state legislature. Virtually all aspects of the state legislature, from laws to representatives' dealings, are lost to the general public.

The Institute for Advanced Technology and Public Policy (IATPP), founded in 2012 by former California State Senator Sam Blakeslee, was created to remedy this problem. It has garnered the formal support of several distinguished individuals including the Honorable Gavin Newsom, former Secretary of State George Schultz, and Silicon Valley Leadership Group CEO Carl Guardino. Its purpose is to use advanced technology to inform and advance public policy, and create solutions for publically relevant societal issues.

One of IATPP's goals is to create online resources for American citizens to view and understand the proceedings of their state legislatures. The efforts to achieve this goal include outlining state legislative bills, surfacing relevant information on elected members of the state legislature as well as third-party influencers likes lobbyists, and providing transcripts for the aforementioned video recordings so viewers know exactly what is being said at any desired timestamp in a hearing and who is saying it. These transcripts in particular are an entirely new dataset previously unavailable to the public. The progress of these efforts is seen in the IATPP's Digital Democracy website (http://www.digitaldemocracy.org).

The Digital Democracy website is a searchable database of the California state legislature's committee hearings, with search capabilities on keywords, topics, speakers, organizations, and dates. When a user queries a desired committee hearing, the website displays its video recording, transcript, and additional data regarding the hearing such as the legislative bills on its agenda, as shown in Figure 1.1.


ASSEMBLY STANDING COMMITTEE ON APPROPRIATIONS HEARING OF 04-27-2016

AB 2735 Jones-Sawyer
PUBLIC EMPLOYEES: UNUSED LEAVE CREDIT
California State Assembly
11:09 / 27:24

Transcript

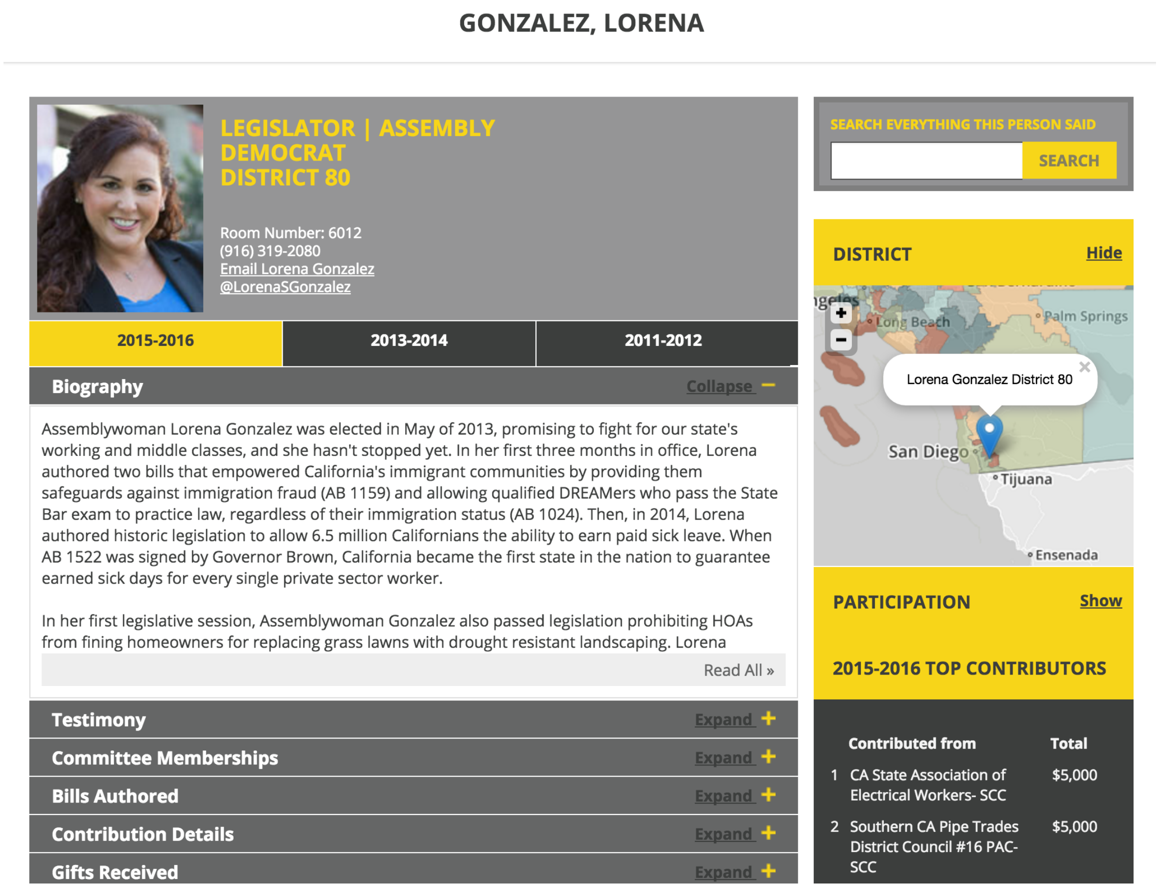
	representing the Association state supervisor. I respectfully ask for your aye vote.
Lorena Gonzalez	Thank you. In support, the bill's been moved and second.
Ted Toppin	Good morning, Madame Chair. Ted Toppin for the Association of California State Supervisors. We certainly thank Mr. Jones-Sawyer for introducing this bill. Obviously, he knows this issue well as a manager, public servant.

AGENDIZED BILLS Hide

AB 2228	Code enforcement officers.
AB 2425	Public health: incident site reports.
AB 2735	Public employees: unused leave credit.
AB 2089	Vote by mail ballots: voter notification.
AB 2126	Public contracts: Construction Manager/General Contractor contracts.
AB 2182	School athletics: neurocognitive testing.
AB 1972	Veterans: state park passes.
AB 1809	CalWORKs eligibility: asset limits.
AB 1993	Corporate law enforcement contacts.

Figure 1.1: Screenshot of a committee hearing page on the Digital Democracy website.

The website also contains information on individual legislators, lobbyists, and any person that has testified in a committee hearing. These speakers can be queried by a name search or by links in a committee hearing transcript. When a user queries a legislator, the website surfaces a breadth of information including biographical data, committee memberships, legislative bill authorships, testimonies, and received donations, as shown in Figure 1.2. When a user queries a lobbyist, the website surfaces a list of his employers and clients, and his testimonies, as shown in Figure 1.3.



GONZALEZ, LORENA

LEGISLATOR | ASSEMBLY
DEMOCRAT
DISTRICT 80

Room Number: 6012
(916) 319-2080
Email Lorena Gonzalez
@LorenaSGonzalez

2015-2016 | 2013-2014 | 2011-2012

Biography — Collapse −

Assemblywoman Lorena Gonzalez was elected in May of 2013, promising to fight for our state's working and middle classes, and she hasn't stopped yet. In her first three months in office, Lorena authored two bills that empowered California's immigrant communities by providing them safeguards against immigration fraud (AB 1159) and allowing qualified DREAMers who pass the State Bar exam to practice law, regardless of their immigration status (AB 1024). Then, in 2014, Lorena authored historic legislation to allow 6.5 million Californians the ability to earn paid sick leave. When AB 1522 was signed by Governor Brown, California became the first state in the nation to guarantee earned sick days for every single private sector worker.

In her first legislative session, Assemblywoman Gonzalez also passed legislation prohibiting HOAs from fining homeowners for replacing grass lawns with drought resistant landscaping. Lorena

Read All »

Testimony — Expand +
Committee Memberships — Expand +
Bills Authored — Expand +
Contribution Details — Expand +
Gifts Received — Expand +

SEARCH EVERYTHING THIS PERSON SAID
SEARCH

DISTRICT — Hide

PARTICIPATION — Show

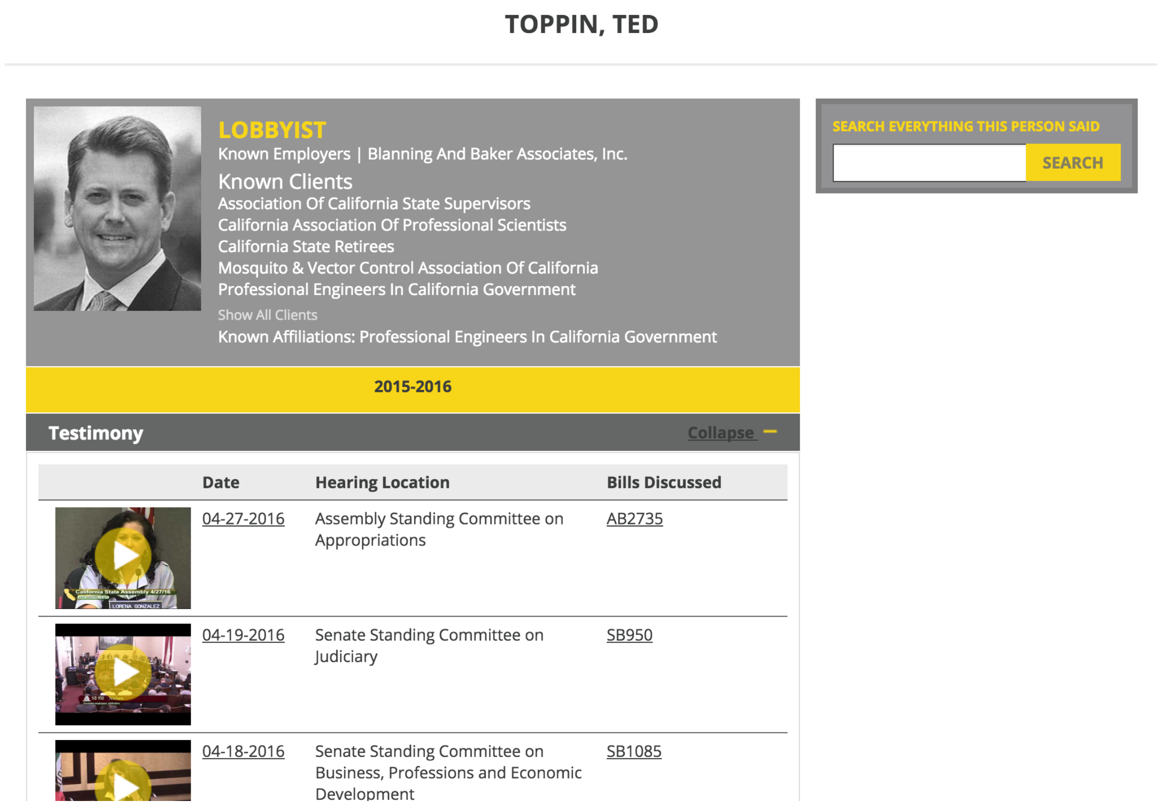
2015-2016 TOP CONTRIBUTORS

	Contributed from	Total
1	CA State Association of Electrical Workers- SCC	$5,000
2	Southern CA Pipe Trades District Council #16 PAC-SCC	$5,000

Figure 1.2: Screenshot of a legislator page on the Digital Democracy website.

The Digital Democracy project obtains data on the state legislature from a variety of public sources. Video recordings of committee hearings, information on legislative bills, information on legislators and lobbyists, and records of political contributions are all obtained from official public databases and websites. Transcripts and additional metadata, such as positions of speakers and organizations on specific bills, are created by a suite of internal tools.

Aside from the video transcriptions, these efforts in their current state are focused on surfacing information that was not readily understandable or popular, but was already accessible through public sources. In this thesis, we extend on the success



TOPPIN, TED

LOBBYIST

Known Employers | Blanning And Baker Associates, Inc.

Known Clients

Association Of California State Supervisors
California Association Of Professional Scientists
California State Retirees
Mosquito & Vector Control Association Of California
Professional Engineers In California Government

Show All Clients

Known Affiliations: Professional Engineers In California Government

SEARCH EVERYTHING THIS PERSON SAID

SEARCH

2015-2016

Testimony

Collapse

	Date	Hearing Location	Bills Discussed
	04-27-2016	Assembly Standing Committee on Appropriations	AB2735
	04-19-2016	Senate Standing Committee on Judiciary	SB950
	04-18-2016	Senate Standing Committee on Business, Professions and Economic Development	SB1085

Figure 1.3: Screenshot of a lobbyist page on the Digital Democracy website.

of the video transcriptions by creating and surfacing new semantic knowledge from the IATPP's transcripts. This process is done in three stages. First, we analyze the transcripts of committee hearings using NLP tools such as AlchemyAPI's entity extraction system, locating and annotating important named entities in the transcript text[6]. Types of entities include, but are not limited to, person names, organizations, locations, legislative bills, and monetary figures. Additionally, the text is processed through a sentiment analysis engine to identify the emotion of spoken words and sentences[67]. These annotations are formatted alongside the raw transcript text and speaker names into XML documents for the next stage of analysis.

The second stage of this process involves organizing the annotations. Each annotation contains a speaker's name and a sequence of entities that reflect actions, statements, and questions from that speaker. Each annotation represents a simple sentence of knowledge about a particular speaker. The complete set of annotations regarding one individual speaker provides a summary of his or her major contributions to the legislative process. These annotations are populated into a knowledge graph using the popular Neo4j graph database system[63], interconnecting the annotated entities and providing a basis for generating new semantic knowledge.

To display the usefulness of the knowledge graph, we conduct several case studies, querying the graph on specific bills, speakers, and topics. These case studies provide validation of the knowledge graph's capability to surface information not readily obtained from the raw transcript text or from the existing Digital Democracy database.

Finally, we provide a prototype client interface for users to easily access the new information from this knowledge graph. In addition to the information already gathered from public sources on speakers, organizations, and bills, users can view novel knowledge regarding those parties and unknown to and unobtainable from any other source.

The system presented in this thesis was designed and implemented specifically for the Digital Democracy initiative, but can be easily adaptable for many other use cases, including any dataset that includes text transcriptions of audio or video recordings, or any formal proceeding documented by transcribers. Court proceedings are a viable use case for this system.

The contributions of this thesis are the following:

- **A method for annotating Digital Democracy's textual transcripts into XML documents using NLP tools:** The transcripts from Digital Democracy are currently unstructured data that are only represented as supplemental information for committee hearings. This thesis presents a pipeline for running a combination of NLP tools on the individual sentences of these transcripts to extract named entities, noun phrases, and emotions, and outputs these results as XML documents.

- **A method for populating a knowledge graph from annotated transcript XML documents:** This thesis presents the design and population of a knowledge graph from the newly created annotations of the Digital Democracy transcripts. This knowledge graph represents the testimonies spoken by legislators, lobbyists, and the general population.

- **Case studies using the knowledge graph to generate insights regarding committee hearing testimonies:** This thesis presents a number of case studies for which the knowledge graph is queried to provide analysis. This thesis presents the following case studies:

  - Locations of residency for members of the general population opposed to vaccination requirements.
  - Sentiments of legislators toward Bay Area companies and cities.
  - Sentiments of legislators with regards to line item budgets.
  - General sentiments of legislators along party lines.

- **Prototype client for querying the knowledge graph by person name or topic:** Finally, this thesis presents a prototype client for users to analyze sentiment values for a particular person or topic.

Further background on the IATPP and the Digital Democracy initiative is provided in Chapter 2. Chapters 3 and 4 describe the design and implementation of the knowledge graph pipeline, respectively. Validation and the prototype client are discussed in Chapter 5. Chapter 6 provides areas of future work and concludes this thesis.

Chapter 2

BACKGROUND

2.1 The California State Legistature

The 2015–2016 enacted budget for the State of California is $167.6 billion dollars[13]. This budget is put into use by 120 full-time lawmakers spread over 130 committees, supplemented by over 2,000 full-time staff, and influenced by 1,100 state-registered lobbyists[62]. These politicians introduce approximately 5,000 bills each legislative session, each bill affecting California's population of 38 million[62, 43].

However, the general population and the news media have little to no awareness of the state's legislative proceedings. The government does not produce transcripts or meeting minutes for any of its sessions. All legislative hearings are recorded and publicly accessible, however their lack of accessibility and searchability render them all but useless to the public and to the media. Therefore, the only easily obtainable data consists of bill texts, committee recommendations, and vote counts, which only gives a trivial outline of what actually transpires in the legislative process. There is no public access to more nuanced information such as who attached amendments to bills, the language and revisions of those amendments, how debates and negotiations unfolded, whether or not committee members have been influenced by lobbyists, or any number of other factors that define the legislative system. All this adds to a jarring lack of transparency, and more disturbingly, lack of accountability, for a state government which oversees the eight largest economy in the world at $2.3 trillion[62, 19].

The Ralph M. Brown Act was passed in 1953 with the intention to create transparency in the California government by declaring the right of public access to meetings held by public servants[15]. Since its creation, the 686-word text has grown to its current word count of 12,845. The problem lies in enforcement of this text. Unless a court intervenes directly, public servants can virtually ignore this law. Even the Brown Act's minimal standard of meeting minutes goes unnoticed. It is clear that the government alone cannot be relied on to provide transparency and accessibility for its proceedings[62, 15].

2.2 Digital Democracy

One of the IATTP's initiatives is the Digital Democracy online platform. Digital Democracy provides a database of California's state legislative committee hearings, with commitee recordings searchable by keyword, topic, speaker, or date. Its novel contribution is the transcription of every committee hearing. Users can read a hearing's transcript while following along with the video recording, and they can use the transcript to jump to its associated timestamp in the video. Metadata attached to the transcripts enables easy analysis of trends and relationships regarding the speakers and committee members. Users can explore individual participants' testimony, committee positions, and donation and gift histories[62, 26].

The Digital Democracy platform also provides access to campaign contribution data to reveal the exchange of money between special interests and lawmakers as legislation is crafted and voted on. This financial data is pulled in from trusted organizations such as MapLight, the National Institute on Money in State Politics, the SunLight Foundation, state databases of registered lobbyists, and government ethics agencies[62].

The focus of the Digital Democracy project is not just to provide access to this information, but to present data such that it can be meaningfully interpreted and acted upon. Users can use built-in clipping tools to save selected portions of commmittee hearing videos. These clips can be shared on social media platforms and utilized by grassroots organizations, media outlets, bloggers, and other organizations for their own purposes. These tools are designed to be simple enough to be used by the general population yet sosphicated enough for journalists, researchers, and other professionals[62].

2.2.1 Target Users

Lawmakers and their staff can use Digital Democracy to track their activities and those of their colleagues in committee hearings and floor sessions, as well as analyze the work of individual committees and legislative bodies on specific issues. Digital Democracy benefits lawmakers by enabling them to keep their colleagues honest. The committee hearing transcripts and metadata can be used to ensure commitments made during negotiations are kept. Lawmakers are also empowered to discover misrepresentations and dishonest dealings. When discussing Digital Democracy, Dr. Blakeslee recalled several occasions during his tenure as State Senator where he was asked to oppose a bill because the author had broken a commitment to add certain language and he was unable to find out where and when this commitment was made, as well as the specific wording of the commitment. The Digital Democracy project encourages public servants to ensure that their peers act in good faith[62, 26].

Community organizations represent crucial but understated constituencies such as foster children, seniors, the disabled, and the homeless. These organizations do not have the resources to fully voice themselves in Sacramento, and do not enjoy the special access and privileges given to corporate special interest groups. Digital Democracy seeks to balance the scales by providing public access into all committees. These community orgazniations can easily monitor arguments made by legislators and stakeholders and keep up to date with the drafing and editing of bills[62].

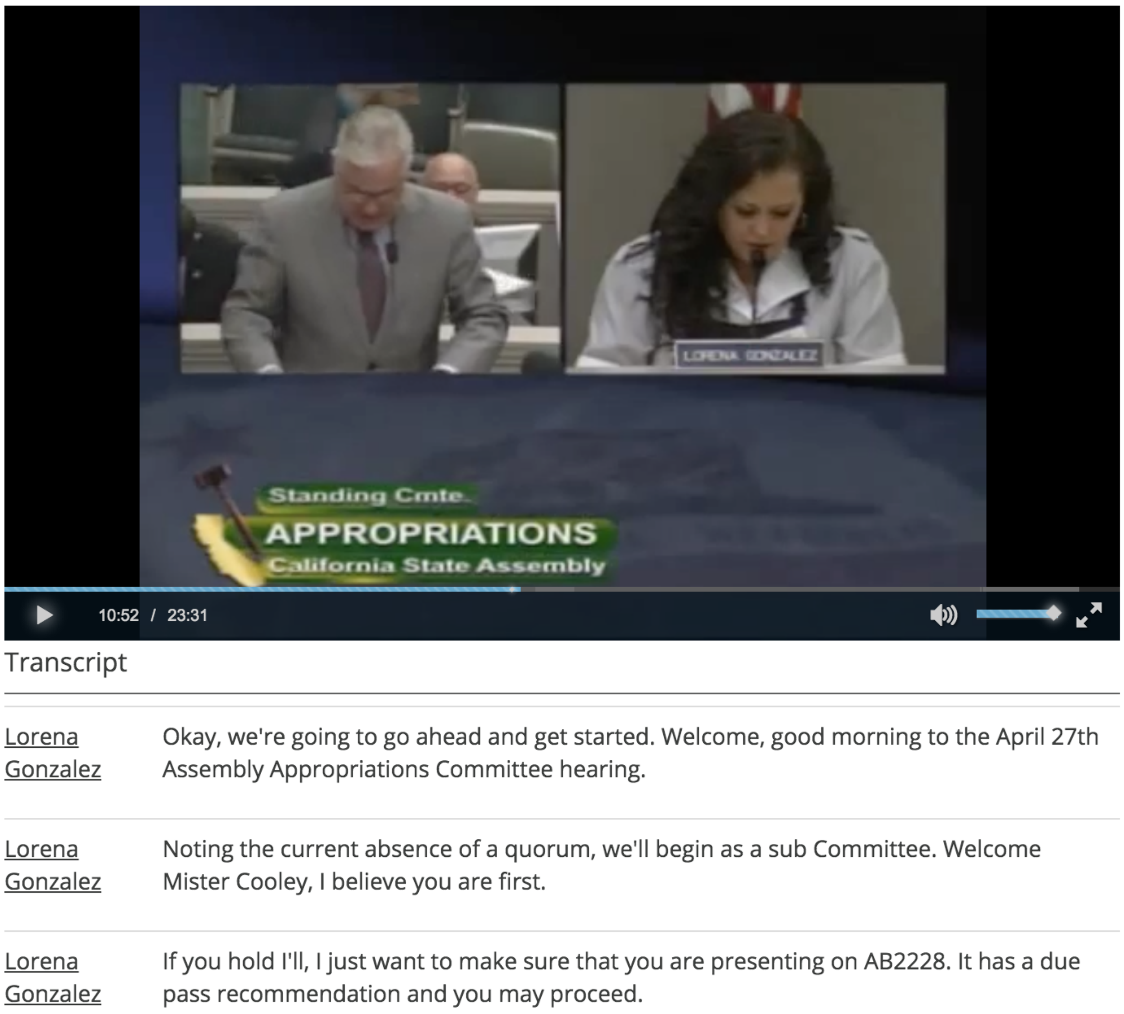
Smaller **media outlets** currently have to piggyback on larger outlets for news on state issues because they face a similar problem as the community organizations: fewer resources. Eighty-six percent of local TV news stations and over two-thirds of U.S. newspapers don't even assign a single reporter to the state capitol. Digital Democracy allows these media organizations to research specific committee hearings, the histories of legislative bills, and the involvement of special interest groups. Digital Democracy revitalizes investigative journalism, enabling quicker discovery of newsworthy moments[62, 28].

Organizations such as MapLight, California Common Sense, SunLight Foundation, and National Institute for Money in State Politics are developing increasingly sophisticated databases and visualization tools that provide better access and insight into the legislative process. Digital Democracy introduces a new dataset in the form of committee hearing transcripts, helping these organizations provide more data on the wide range of factors influencing policy decisions[62].

2.2.2 Data

The Digital Democracy platform integrates and surfaces its own uniquely generated data along with data from a number of publicly accessible sources. The datasets unique to the Digital Democracy platform are listed below[28]:

- Videos of California Legislature committee hearings. These videos are accessible outside of Digital Democracy, but a nontrivial amount of effort is required to access and obtain actionable data from them.
- Transcripts of California Legislature committee hearings. These transcripts are annotated with attribution of all direct speech to individual speakers. They are divided into "utterances", which are individual statements of thought given by a particular speaker. These utterances range from a single sentence to a short paragraph, but are usually no more than a few sentences.



Lorena Gonzalez	Okay, we're going to go ahead and get started. Welcome, good morning to the April 27th Assembly Appropriations Committee hearing.
Lorena Gonzalez	Noting the current absence of a quorum, we'll begin as a sub Committee. Welcome Mister Cooley, I believe you are first.
Lorena Gonzalez	If you hold I'll, I just want to make sure that you are presenting on AB2228. It has a due pass recommendation and you may proceed.

Figure 2.1: Screenshot of the Digital Democracy website displaying a committee hearing video and transcript.

By incorporating data from public sources, Digital Democracy is able to surface contextual information on these committee hearings, including metadata on individual legislators and other speakers, analysis of legislative bills and their histories, and publicly available information on third-party organizations, lobbyists, and campaign contributions. Digital Democracy incorporates the following public data sources[28]:

- LegInfo — As the official website for the California state legislature, LegInfo contains information on the California State Senate and the State Assembly. Digital Democracy imports the following data from LegInfo[20]:

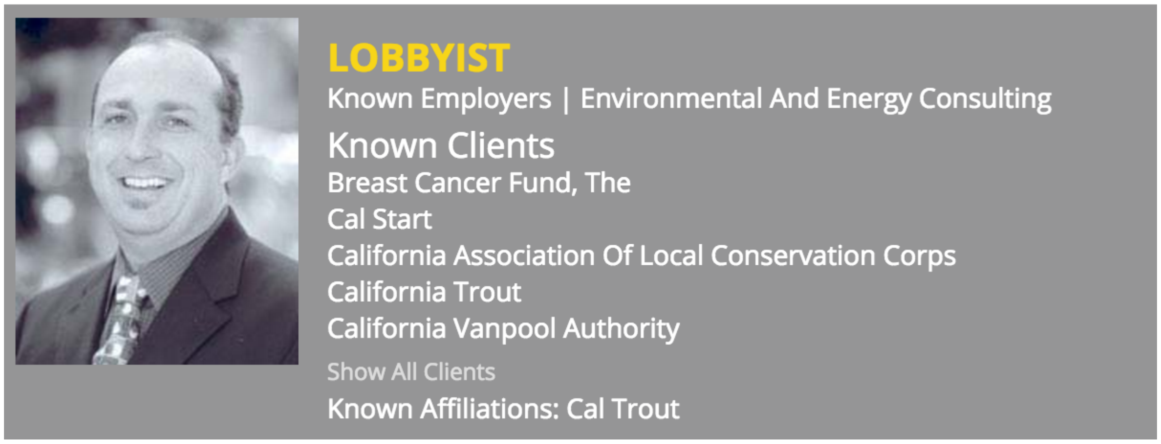
  - Information on the individual members of the State Senate and State Assembly, including past and current biographical text and contact information.
  - Information on the State Senate and State Assembly committee leadership and membership for current and prior legislative terms.
  - Information on legislative bills, including their text, authorship, revisions, the committee hearings and floor sessions they are discussed in, and the committee and floor votes on those bills.

  An example of LegInfo data surfaced on the Digital Democracy website is shown in Figure 2.1.

- California Secretary of State database of registered lobbyists — All registered lobbyists are recorded by the office of the California Secretary of State. Digital Democracy uses this database to gather information on individual lobbyists, lobbying firms, and their employments with other organizations. These lobbyists can then be identified in the committee hearings that they testify in[45]. An example of this data surfaced on the Digital Democracy website is shown in Figure 2.2.

- MapLight — The MapLight website contains information on political donations to Presidential, Congressional, and State Legislative candidates. Digital Democracy uses MapLight's California site to gather information on donations to political and electoral campaigns, including but not limited to those of California's State Senate and Assembly members[37]. An example of MapLight data surfaced on the Digital Democracy website is shown in Figure 2.3.

The Digital Democracy platform collects and stores this data in a single relational database. This database is automatically updated with new data by a number of


LOBBYIST

Known Employers | Environmental And Energy Consulting

Known Clients

Breast Cancer Fund, The

Cal Start

California Association Of Local Conservation Corps

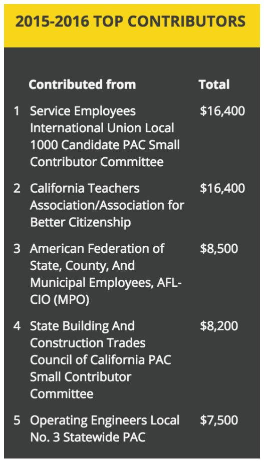
California Trout

California Vanpool Authority

Show All Clients

Known Affiliations: Cal Trout

Figure 2.2: Screenshot of the Digital Democracy website displaying information on Reed Addis from the California database of lobbyists.


2015-2016 TOP CONTRIBUTORS

	Contributed from	Total
1	Service Employees International Union Local 1000 Candidate PAC Small Contributor Committee	$16,400
2	California Teachers Association/Association for Better Citizenship	$16,400
3	American Federation of State, County, And Municipal Employees, AFL-CIO (MPO)	$8,500
4	State Building And Construction Trades Council of California PAC Small Contributor Committee	$8,200
5	Operating Engineers Local No. 3 Statewide PAC	$7,500

Figure 2.3: Screenshot of the Digital Democracy website displaying political donations and gifts given to State Senator Richard Pan. Data obtained from MapLight.

scripts that query the above public datasources. The tables relevant to this thesis are outlined in Table 2.1.

Table 2.1: Relevant Digital Democracy SQL tables

Table Name(s)	Content
`Person`	Name of every person who has spoken in a committee hearing or floor session, including state legislators, lobbyists, and members of the general public.
`servesOn`	Committee membership, including rank, for every state legislator.
`Committee`	Name and type of every committee in the state legislature.
`Bill`	Name, current legislative status, and other metadata on every bill.
`BillVersion`	Subject, text, and status of every version of very bill.
`BillVoteDetail`, `BillVoteSummary`	Record of every vote cast by a state legislator for a given bill at a given committee hearing or floor session.
`Organizations`	Name of every company and third-party organization that has contributed to the legislative process, from testifying at a hearing through lobbyists to contributing financially to a legislator.
`Lobbyist`	Identifier for every person in the "Person" table that is registered as a lobbyist.

`LobbyistRepresentation`	Record of every instance where a lobbyist has representated an organization at a committee hearing.
`CommitteeHearings`, `Hearing`	Every committee hearing and floor session.
`Video`	Metadata for every video recording of a committee hearing or floor session.
`BillDiscussion`	Start and end video timestamps during which a given bill is discussed.
`Utterance`	Every utterance spoken in every committee hearing and floor session, the timestamp of the utterance in the corresponding video recording, and the speaker's position on the bill being discussed.

2.3 Graph Databases

After running a suite of NLP libraries to annotate the committee hearing transcripts, SKEWER aggregates the resulting annotations into a graph database, creating edges connecting individual speakers to the entities they mentioned.

2.3.1 Graph Database Model

A graph database model uses graphs to represent data and/or the schema of said data[11]. The degree to which graphs are used varies on the model in question. The data model of GraphDB, intended for modeling graphs in object-oriented databases, represents the entire database as a single graph[23]. The hypernode model uses a labeled directed graph as the single underlying data structure, called a hypernode, and the database consists of a single directed graph of hypernodes. This model allows for the representation of much more complex objects and entities than data models like GraphDB can usually accommodate[34]. The Gram data model uses a directed weakly connected labeled multigraph as a schema, in which each node is labeled with a type, and each type is associated with a domain of values. Similarly, each edge has a label representing a relation between two types. The database in this model is simply an instance of this schema[8]. An example of a Gram database is shown in Figure 2.4.

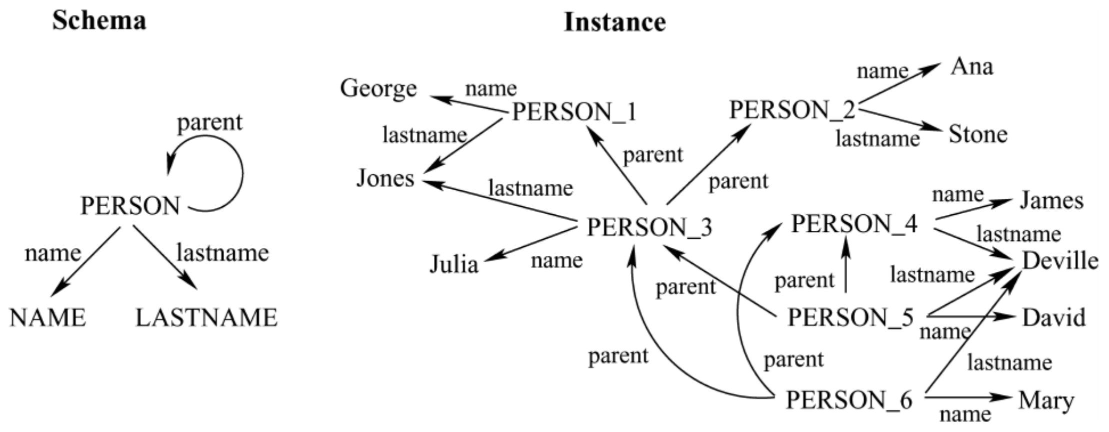


Figure 2.4: A small family tree represented using the Gram data model[11].

Data manipulation in a graph database model can be expressed by graph transformations, as demonstrated in the Graph-Oriented Object Database Model (GOOD)[11]. Queries and data manipulation can also be expressed by operations whose main primitives are on graph features like paths, neighborhoods, subgraphs, graph patterns, connectivity, and statistical values like diameter and centrality[24]. The GraphDB model uses a small number of powerful graph manipulation primitives to express all possible queries. For example, Figure 2.5 shows a GraphDB query to find the titles of all books written by Hopcroft in 1983.

```
on person wrote book

where person.name = "Hopcroft" and book.year = 1983

derive book.title
```

Figure 2.5: A sample GraphDB query to find book titles for a given author and year of writing

The first clause states that each pair of *person* and *book* nodes connected by a *wrote* edge should be considered. This combination of two nodes with a connecting edge is called a “triple”. This collection of triples is filtered using the two equality comparison statements in the where-clause. Finally, derive-clause creates a result object for each selected triple, where each object as a single attribute *title* whose value is taken from the attribute *title* of the *book* object in the triple[23].

To ensure data consistency, graph database models enforce integrity constraints. Constraints can be categorized into schema-instance consistency, identity and referential integrity, and functional and inclusion dependencies[11]. Common constraints are labels with unique names, typing constraints on nodes, functional dependencies similar to those of relational data models, and domain and range of properties like in Resource Description Frameworks (RDFs)[21, 33, 31, 32].

2.3.2 Advantages

Graph data models are often used in applications where information about data interconnectivity and/or topology is as important, if not more important, than the data itself. These applications usually give equal value to data and relations among data. Graph models give a layer of abstraction that allows for a more natural modeling of this type of data, and also allow a more natural of handling or querying such data[11, 10]. Graphs are able to store all information about an entity in a single node, while related information is shown by edges connected to that node[47]. Graph structures can also explicitly define portions of a larger database, as demonstrated in the hypernode model, allowing encapsulation and context definition[11, 34].

Specially designed query languages and operators allow for direct querying of these graph structures. Query languages can include complex graph-specific operations, such as finding shortest paths and retrieving subgraphs in GraphDB[23]. The high level of abstraction given by these query languages relieves the burden on the user to have full knowledge of the underlying graph structures[11].

When compared to traditional relational database models, graph models enjoy greater flexibility and extensibility. The relational model favors simple record-type data, where the data structures are static and known in advance. The modification or expansion of a relation schema is often a major undertaking. Additionally, relational query languages are not well-suited for traversing the implicit graph of relationships among the different relational tables[11].

Graph database models are now grouped into the broader category of non-relational models, termed NoSQL database models. The other types of NoSQL models—key-value stores, column stores, and document stores—store data as sets of disconnected aggregates, which are difficult to use for connected data and graphs. The common way of representing data relations in these models is to embed an aggregate's unique identifier as a field in another aggregate, essentially creating a foreign key. However, this requires that the aggregates be joined at the application layer rather than the database layer, which easily becomes an expensive operation[52]. Graph database models naturally represent data relationships as edges, so the joining of objects is done at the database layer and without the need of additional object properties.

2.3.3 Complex Networks

A set of data that naturally takes the form of a graph can be called a *network*. There has been a major shift in research with regard to these networks, in large part due to the availability of computers and communication networks that allow researchers to collect and analyze data on a massive scale. Whereas early research used networks with tens or hundreds of vertices and mainly studied the properties of individual vertices and edges, today's researchers analyze networks with millions or billions of vertices and focus on the large-scale statistical properties of these networks. The size of these networks is such that humans cannot simply "eyeball" graph visualizations to gain meaningful insights. Such networks are coined "complex networks", and are the perfect use-case for graph database models[42, 3, 17].

Complex networks can be divided into four categories[11, 42]:

- Social networks — Nodes represent people or groups, and edges show relationships among nodes. Examples of social networks include personal relationships like Facebook, business relationships like LinkedIn, research networks showing collaboration and coauthorship, communication records such as telephone calls and emails, computer networks, and networks for national security[55].
- Information networks — These networks model relations representing the flow of information. Examples are citations among academic papers, the World Wide Web, and peer-to-peer networks.
- Technological networks — These man-made networks facilitate distribution of a commodity or resource. Examples are the Internet, electric power grids, airline routes, telephone networks, mail delivery networks, and geographic information systems.

- Biological networks — These networks represent biological data whose volume, management, and analysis has grown beyond the traditional scope due to the recent automation of data gathering. Examples include networks in the field of genetics, food webs, and biological neural networks.

2.4 Knowledge Graphs

Explicit knowledge—knowledge that has been formulated and can be readily retrieved and articulated—can be represented in several ways. Among the oldest forms of explicit knowledge are written or recited texts, which can be obscure. It is uncommon for a written text to present the entire problem area or full scope of knowledge, so the reader does not get a complete overview. Knowledge that is structured in schemes, such as hierarchical tree structures and arrow diagrams, can present a complete overview[49]. However, two problems are still prevalent. First, language is largely ambiguous, resulting in a variety of ways to interpret any single concept[48]. Second, knowledge is dynamic, while text is a static representation of knowledge. The discovery of new knowledge or modification of existing knowledge requires the corresponding text to be rewritten[49].

The two high-level approaches to knowledge representation are logical formalism and structured representation[50]. Predicate logic is an example of logical formalism. Semantic networks—graph structures for representing knowledge through labeled nodes and edges—are an example of structured representation[57]. A knowledge graph is a type of semantic network that only uses a small number of relationship types, and allows for the addition of new knowledge which can be integrated with the graph's existing knowledge[27].

The first step to building a knowledge graph is text analysis, which is the extraction of information from texts. Text analysis results in a list of concepts. A concept is the basic unit for the content of what it refers to[48]. An “author graph” is constructed with each node corresponding to a specific concept, and with each link of one of the allowed relationship types[29]. The specific techniques for marking concepts include classic text mining techniques like named-entity recognition, while NLP techniques such as coreferencing and part-of-speech tagging help identify the links. The most important type of link between concepts is the causal relation[49].

The next step is concept identification, or the compilation of the author graphs into a single graph by combining their corresponding nodes. Due to the ambiguity of language, nodes with different labels may represent synonyms for a single concept. To remove this ambiguity, neighborhoods of nodes are compared to identify potentially identical pairs. A similar technique is also used to distinguish homonyms, such as a chair that someone sits in, versus a committee chair or chair of a department. The resulting graph is free of linguistic ambiguities. This graph is further refined by alternating between concept integration, in which interesting substructures are identified, and link integration, in which new links are inferred for the existing ones. This now “integrated graph” can be further refined with additional iterations of concept integration and link integration[29].

Originally, only three relation types were allowed—CAU (causal relation), PAR (is part of), and AKO (is a kind of). Some early research allows the related inverse relations as well: CBY (is caused by), HAK (has as kind), and HAP (has as part)[60]. Current research defines two types of concepts: tokens and types. Tokens are like variables in logic, while types are generic concepts determined by their attribute sets, and can be thought of as schema information. For example, “dog” is a type, and “Pluto” is a token. The only relation between token and type is ALI (alike), while there are seven relations between types. In addition to the original three, relations between types also include ORD (ordering), ASS (symmetric association), EQU (equal or symmetric), and DIS (distinct)[49].

2.5 Named-Entity Recognition

Named-entity recognition (NER), also referred to as entity identification or entity extraction, is a subtask of text analysis that locates and classifies “named entities”, or units of information, in a, typically unstructured, text. Types of named entities include, but are not limited to, names of physical concepts including persons, organizations, and geographical locations, and numerical expressions including times, dates, monetary figures, and percentages. The input of a typical NER system is a text, and the output consists of the locations and types of named entities found in the text[41].

The accuracy of an NER system is measured by its F_1 score, also referred to as F-measure in this thesis. In the field of information retrieval, “precision” for a given input is defined as the fraction of results that are correct, or the ratio of true positives (tp) to all predicted positives (tp + fp). “Recall” is defined as the fraction of an input’s actual results that are found by the system, or the ratio of true positives to all actual positives (tp + fn). The F_1 score, shown in equation 2.3, is the harmonic mean of the precision and recall, with a ceiling of 1 and a floor of 0.

$$p = \frac{tp}{tp + fp} \tag{2.1}$$

$$r = \frac{tp}{tp + fn} \tag{2.2}$$

$$F_1 = 2\frac{pr}{p + r} \tag{2.3}$$

The most accurate form of NER is manual annotation. The highest F-measure from a human annotator at the seventh Message Understanding Conference (MUC-7) in 1997 was 0.9760, with both precision and recall at 0.98[38]. However, because human labor is involved, this technique does not scale well into large numbers of text documents or texts of nontrivial length. Computerized NER systems can be classified based on their high-level machine-learning styles, or lack thereof. Handmade rule-based systems use rules given by expert linguists in their processing model. Systems based on supervised learning train themselves on a set of fully labeled input data to improve their accuracy[14, 41]. Such systems often use Conditional Random Fields (CRFs) in their statistical modeling. In 2012, Tkachenko and Simanovsky developed a CRF-based supervised learning NER system that achieved an F-measure of 0.9102 on the popular CoNLL 2003 dataset[65]. Both handmade rule-based systems and supervised learning systems require the input training data to be manually labeled by human experts, which limits the effectiveness and usability of large-scale NER systems[41].

Systems based on unsupervised learning base their rulesets on the data structures deduced from the input data, without the use of pre-labeled items or external resources[14, 41]. Because unsupervised NER systems do not require human labor, they are only limited in scale by their implementations. Recent research has produced unsupervised learning systems that can achieve F-measures above 0.8. In 2012, Munro and Manning developed an unsupervised system that jointly learns to identify entities in parallel text.[1] This system achieved an F-measure of 0.8610 on the CoNLL 2003 dataset[40].

Semi-supervised learning systems use a combination of labeled and unlabeled data. By requiring only a small amount of labeled data, these systems avoid the labor costs of manually creating large labeled datasets. More effort is often needed to design effective models and similarity functions. Because labeled data is so scarce, especially in comparison to unlabeled data, semi-supervised learning methods make strong model assumptions. Bad matching of problem structure with model assump-

[1]Parallel text is a text formatted alongside one or more of its translations.

tion can non-trivially reduce system performance and accuracy[69, 16]. However, when done correctly, semi-supervised learning systems can achieve an accuracy comparable to supervised systems. These systems are therefore better suited than fully supervised systems for most real world applications, which often don't allow for the time or resources to acquire such large amounts of labeled training data. Ando and Zhang developed a semi-supervised system that achieved an F-measure of 0.8931 on the CoNLL 2003 dataset from a labeled dataset of 204K words and 27M unlabeled words[9]. Suzuki and Isozaki created a similar system that achieved an F-measure of 0.8992 on the CoNLL dataset from the same amount of labeled data as Ando and Zhang's and from one billion words of unlabeled data[61].

2.6 Natural Language Processing Tools

The NLP tools used in the annotation of committee hearing transcripts are discussed in detail in Chapter 4. AlchemyAPI is a SaaS-based collection of cognitive APIs developed by IBM and is a subset of the Watson Developer Cloud[4]. spaCy is an industrial-strength NLP library featuring word tokenization, part-of-speech tagging, NER, and syntactic dependency parsing[58]. TONGS (TLDR; Online Narrative Generating System) uses the Natural Language Toolkit to determine the emotional value of sentences[67]. Finally, the Stanford Parser is a probabilistic natural language parser used for calculating the grammatical structure of sentences[22].

Chapter 3

DESIGN

The transcripts of committee hearings are a new set of unstructured textual data. They serve not only as a companion to the user for watching legislative committee hearings, but also as a searchable index into the raw spoken word of every single legislator and speaker that has ever testified in any of those hearings. However, it is difficult to analyze the transcripts in their given state for pattern identification and insight, due to the sheer amount of text. The ultimate goal of this thesis is to provide a method for modeling the textual data to allow queries of various targets—person, company, legislative topic, etc.—for the discovery of new knowledge and insight regarding the parties involved in the state legislative process. The method proposed by this thesis uses natural language processing tools to parse and annotate the transcripts, and collects the annotation results into a knowledge graph.

The knowledge graph proposed by this thesis incorporates data from the relational MySQL database with the new data generated from the named entity recognition and sentiment pipeline described later in this thesis. Section 3.1 describes the data model of the imported MySQL data in the knowledge graph. Section 3.2 describes the model of the data created through the named entity recognition pipeline.

The goal of the knowledge graph is to provide new semantic information based on the transcriptions of committee hearings that reveals interesting statistics and patterns regarding testimonies from all parties involved in the legislative process. The graph should contain sufficient data to accomplish this goal without requiring the user to make supplemental queries to the existing MySQL database. However, the graph should also not contain an abundance of extraneous data, which inevitably consumes both memory and disk space and slows the performance of graph queries, especially as the Digital Democracy platform expands to include other states.

3.1 Existing Data

3.1.1 Node Types

The backbone of the knowledge graph is the `Person` node. A `Person` node represents a single person and has four node properties: a unique integer *uid*, *first* name, *last* name, and an optional *middle* name. Each node is also one of three subtypes: `Legislator`, `Lobbyist`, or `Public`. The `Legislator` subtype designates the person as a member of the California state legislature. Each Legislator node is also one of two sub-subtypes—`Senator` or `Assemblymember`—designating the house that the legislator is a member of. The `Lobbyist` subtype indicates that the person is a state-registered lobbyist. The `Public` subtype indicates that the person is a member of the general public and has testified in one or more committee hearings.

A `Committee` node represents a Senate or Assembly committee. In addition to a unique integer *cid*, each `Committee` node has four other required properties. The *house* property indicates the legislative house of the committee—either "Senate" or "Assembly". The *type* property indicates the official type of the committee: Standing, Sub, Select, Joint, Extraordinary, or Other. The *name* property indicates the name of the committee, and is usually of the form "<house> <type> Committee on <Topic>", where <Topic> is a legislative issue like "Agriculture" or "Education". There also exists a `Committee` node to represent the entire Senate body, and likewise for the entire Assembly body.

A `Hearing` node represents a committee hearing or floor session. Each node has an unique integer *hid* and a string *date* property.

A `Bill` node represents a proposed piece of legislation that is currently being or has been considered by the legislature. Each node has a unique *bid*, a *billState*, a *subject*, and a *name* property, where the name contains a acronym and a number, such as "SB 277". The *billState* property indicates the current status of the bill. Examples of *billStates* are "Proposed", "Amended", and "Vetoed".

An `Organization` node represents a company or other organization that has participated in the legislative political process, whether by contributing to a legislator or testifying in a committee hearing. Each node as a unique *oid* and a *name* property.

3.1.2 Edge Types

An edge from a `Legislator` node to a `Committee` node indicates that the legislator is a member of that committee. These edges are one of two types—`Chair` or `Member`—indicating the legislator's position on that committee. Each edge has a *year* property indicating the year for which the legislator was on that committee. Each `Committee` node can have multiple inbound `Member` edges, but only one `Chair` edge for any given year.

A `HeldHearing` edge from a `Committee` node to a `Hearing` node indicates that that committee was the organizer of that hearing. These edges have no other properties.

A `Vote` edge from a `Legislator` node to a `Bill` node represents the legislator's vote for the given bill. The *result* property of the edge indicates the vote selection: AYE (yes), NOE (no), or ABS (abstain). Because a bill can go through any number of committees and revisions, a legislator may vote on a single bill multiple times. For simplicity, only the most recent, or final, vote from a legislator for a bill is represented in the knowledge graph. This also has the benefit of clearly establishing the legislator's current or final position on the bill, as a legislator may switch sides any number of times as the bill goes through the legislative process.

A `LobbiesFor` edge from a `Lobbyist` node to an `Organization` node indicates the lobbyist is or has been employed by that organization, possibly through an intermediary lobbying firm. Each edge has a *bid* property, indicating the bill for which the lobbyist was testifying on behalf of the target organization.

A `Testified` edge from a `Person` node to a `Hearing` node indicates that the person testified at the given committee hearing. These edges have no other properties.

Figure 3.1 shows a diagram of the edges described in this section and their start and end nodes.

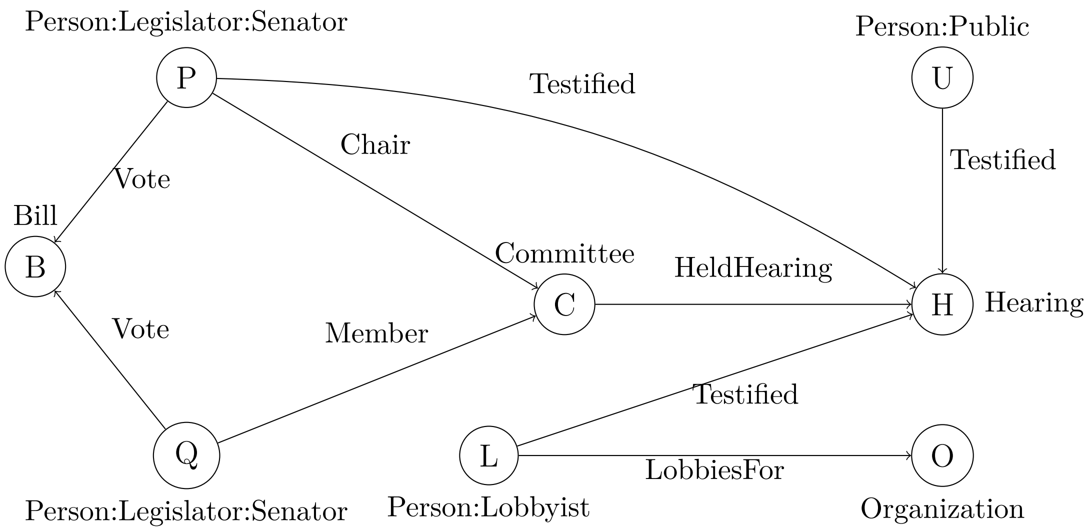


Figure 3.1: Sample diagram of knowledge graph edges created from existing relational data.

3.2 New Data

The nodes and edges described in this section are created from the transcript utterance annotations and sentiment values outputted by the tools introduced in Section 2.6.

3.2.1 Node Types

Named entities are represented as nodes in the knowledge graph. Each node has a name property, representing the textual name of the "thing" represented by that node. The type of the node indicates the classification of the represented named entity. The most common types are listed below, in alphabetical order:

- City
- Event
- Facility (a physical building)
- JobTitle
- Location
- Money
- Norp (Nationality or religious or political group)
- Organization (company or other business)
- Percent
- PersonName
- Religious
- Time (includes dates)
- Topic

A noun phrase is a word or group of words that functions in a sentence as a subject, object, or prepositional object[1]. Noun phrases are represented in the knowledge graph as Phrase nodes. Each Phrase node has a name property, similar to named entity nodes, representing the text of the phrase.

3.2.2 Edge Types

A `Said` edge from a `Person` node to a named entity node or a `Phrase` node indicates that the person has said that entity or phrase at least once in a committee hearing or floor session. The edge's *bid* property is the integer id of the legislative bill that was being discussed at the time, and can be used to query the corresponding `Bill` node. The *sentiment* property is a numerical value representing the sentiment score of the utterance in which the entity or phrase was spoken. Positive values indicate positive sentiments, and negative values indicate negative sentiments. The *words* property is a list of words in the utterance that contributed, positively or negatively, to the numerical sentiment value. The optional *align* property indicates whether the person was determined to be "for" or "against" that bill based on the utterance in which the entity or phrase was spoken.

A `Said` edge can also connect two `Person` nodes, indicating that the person represented by the edge's source node said the name of the person represented by the edge's destination node. If the mentioned person's name is unknown to the graph, a `PersonName` named entity node is used as the edge's destination.

A `Sentiment` edge from a `Person` node to a `Bill` node represents the sentiment of a single utterance spoken by the person about that bill. The *val* property is a numerical value representing the sentiment score. The *words* property is a list of words in the utterance that contributed to the numerical sentiment value. The optional *align* property indicates whether the person is "for" or "against" that bill based on the utterance. These properties are identical to those of `Said` edges, but the purpose of `Said` edges is to gain information on a person's opinion towards an entity such as a company, city, or another person. The purpose of `sentiment` edges is to gain information on a person's opinion towards a legislative bill.

Figure 3.2 shows a diagram of the edges described in this section and their start and end nodes.

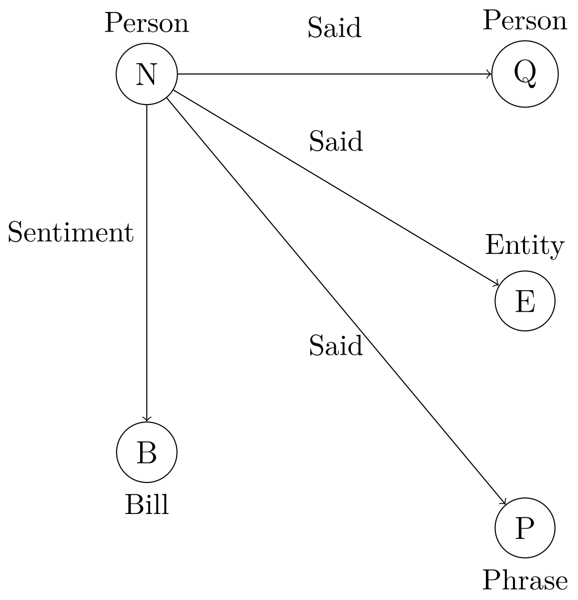


Figure 3.2: Sample diagram of knowledge graph edges created from the knowledge graph population pipeline.

Chapter 4

IMPLEMENTATION

This thesis uses Neo4j, a highly scalable, enterprise-strength, ACID compliant, native graph database as its knowledge graph implementation[63]. Further discussion of Neo4j's capabilities is in Section 4.1.

The first step of the Digital Democracy knowledge graph pipeline is the pre-population of data from the existing MySQL database to create the nodes and edges of the knowledge graph as described in Section 3.1. Second is the annotation of committee hearing transcripts. These hearing transcripts are formatted into XML files, with named entities tagged as elements based on their classification. Finally, the XML files are processed to create the nodes and edges of the knowledge graph as described in Section 3.2.

4.1 Neo4j

Neo4j is the world's most popular graph database according to Forrester Research and the only graph database on Gartner's Magic Quadrant for Operational Database Management Systems, as seen in Figure 4.1[2, 18]. It has 200 enterprise subscription customers, including at least fifty in Forbes's Global 2000 list of public companies. The International Consortium of Investigative Journalists used Neo4j to model and analyze the data from the Panama Papers leak in 2016[25].

Neo4j implements a native labeled property graph model. The property graph contains connected nodes, which can hold any number of key-value pair attributes, or properties. Nodes can also be marked with labels representing their various roles in the domain of the graph. Relationships, or edges, can have properties and labels

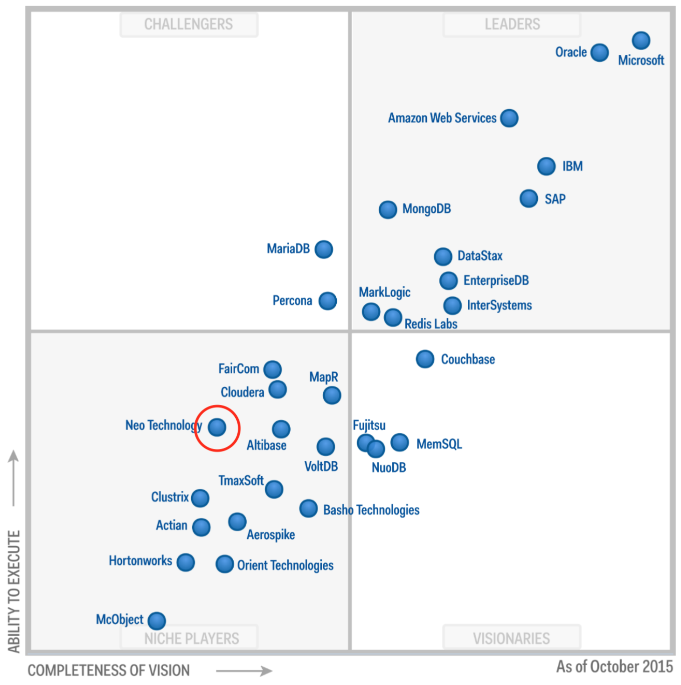


Figure 4.1: Gartner's Magic Quadrant for Operational Database Management Systems[18]. Neo4j (Neo Technology) is the only graph database to be included.

as well. While all relationships are directed, they can be navigated in both directions without affecting performance. The number of relationships between two nodes also does not affect performance. Uniqueness constraints can be used to enforce data integrity on node properties, and enterprise Neo4j versions can create property existence constraints on both nodes and relationships.

Neo4j utilizes index-free adjacency, where each node maintains direct references to its adjacent nodes. Each node acts as an index of other nearby nodes, which not only consumes less disk space than global indexes, but also means that query performance is not proportional to the total size of the graph. To support index-free adjacency, Neo4j uses native graph storage. The graph data is stored in a number of *store files*, where each store contains data for a specific component of the graph: nodes, relationships, labels, and properties. Each node record in the node store file contains pointers to its related data in the other store files, supporting extremely fast traversals. Similarly, each relationship record in the relationship store file stores pointers to the next and previous relationship records for both its start and end nodes. Both stores use fixed-size records, so the location of a node or relationship is easily computed[52].

The declarative graph query language used in Neo4j is called *Cypher*. It is a SQL-like language designed for efficient querying and updating of the graph store, and builds upon a number of established practices in related fields. For instance, many Cypher keywords are inspired by SQL, and Cypher's pattern matching syntax is inspired by SPARQL expressions[64]. Figure 4.2 shows an example Cypher query that creates an edge between two nodes.

```
MATCH (p:Object {id:100}), (q:Object {id:101})
CREATE (p)-[:related {property1:'value', property2:3.5}]->(q)
```

Figure 4.2: Example Neo4j Cypher query to create an edge between 2 nodes

4.2 Knowledge Graph Pre-population from Relational Database

The first task of creating the knowledge graph is inserting data from the existing Digital Democracy relational database. The steps to this task are outlined in Figure 4.3.

`Person` nodes are the first nodes inserted into the knowledge graph. The `servesOn` and `Lobbyist` SQL tables are queried to identify which persons are legislators and lobbyists, respectively. Then for each record in the `Person` table, a node with first, middle, and last name properties is inserted into the knowledge graph. Each node has a minimum of two labels, one being a `Person` label. The second label is either

1. Create `Legislator`, `Lobbyist`, and `Public` nodes.
2. Create `Committee` and `Hearing` nodes.
3. Create `Chair` and `Member` edges from `Legislator` nodes to `Committee` nodes.
4. Create `Bill` nodes.
5. Create `Vote` edges from `Legislator` nodes to `Bill` nodes.
6. Create `Organization` nodes.
7. Create `LobbiesFor` edges from `Lobbyist` nodes to `Organization` nodes.

Figure 4.3: Outline of knowledge graph pre-population from Digital Democracy database

`Legislator`, `Lobbyist`, or `Public`, depending on the records of the `servesOn` and `Lobbyist` tables. If the node has a `Legislator` label, it also has a third label, either `Senator` or `Assemblymember`, as determined from the `servesOn` table.

Next, legislative committees and their hearings are inserted into the knowledge graph. Each record in the `Committee` table is translated into a `Committee`-labeled node, with the corresponding properties as described in Section 3.1, in the knowledge graph. The `CommitteeHearings` table is then queried to identify which committee is responsible for each hearing in the `Hearing` table. These two tables are translated into `Hearing`-labeled nodes and `HeldHearing`-labeled edges from the `Committee` nodes to those `Hearing` nodes.

The `servesOn` table is queried to identify committee memberships in the graph. Each record in `servesOn` is translated into either a `Chair`-labeled or `Member`-labeled edge from a `Legislator` node to a `Committee` node. If the position column in the table is NULL for a particular row, it is assumed that a `Member`-labeled edge should be created.

The `Bill` and `BillVersion` tables is queried to create the corresponding `Bill` nodes in the graph. The node properties are described in Section 3.1.

The `BillVoteSummary` and `BillVoteDetail` tables are translated into `Vote`-labeled edges in the graph. For each record in `BillVoteDetail`, a `Vote` edge from the corresponding Legislator node to the target `Bill` node is either created or updated. If the edge already exists, its *result* property is updated to the result column value of the `BillVoteDetail` row. The rows are sorted by ascending date, so that when all the rows have been processed, each `Vote` edge represents the legislators' final votes for the corresponding bills.

The `Organization` table is translated into `Organization`-labeled nodes in the graph. Similarly to the `Bill` table-node translation, nothing noteworthy is involved in this step.

The `BillDiscussion` and `LobbyistRepresentation` tables are queried to create `LobbiesFor`-labeled edges in the graph. The `BillDiscussion` table is used to fill in the *bid* properties of the edges.

The *pid* and *bid* properties of `Person` nodes and `Bill` nodes, respectively, are indexed to improve graph query performance. Specific bills are often the main targets of queries directed at the knowledge graph, so indexing by their unique identifiers decreases the time to locate a targeted `Bill` node in the graph. While it possible for specific persons to be targeted by user queries, the main reasoning behind indexing *pids* in `Person` nodes is to reduce the runtime of creating the edges described in Section 3.2. All of those edges have `Person` nodes as their sources, so by indexing on *pid*, the time to locate those nodes is reduced significantly.

4.3 Transcript Annotation

The transcript annotation pipeline incorporates four major third-party tools: AlchemyAPI and spaCy for named entity recognition, the Stanford Shift-Reduce Constituency Parser for noun-phrase extraction, and the TONGS Amazon Review analyzer for sentiment analysis. These tools are discussed in detail in the following subsections.

4.3.1 AlchemyAPI

AlchemyAPI is an IBM-based company that provides NLP services for processing unstructured textual and image data into information[4]. Its main service is AlchemyLanguage, a set of 12 API functions for different forms of text analysis[5]. This thesis uses the Entity Extraction function for identifying people, companies, organizations, cities, geographic features, and other typed named entities from the transcripts[7].

Although AlchemyAPI does not provide any accuracy analysis for its Entity Extraction function, outside researchers have tested its accuracy with mixed results. Using Holocaust survivor testimonies as well as newsletters written for the crew of the H.M.S. Kelly in 1939 as input data, and targeting only person names, organizations, and locations, the Entity Extraction function performed with a F-measure of 0.48[53]. Using news articles from BBC, CNN, The New York Times, and Yahoo! News as input data, the Entity Extraction function performed with a precision of 0.74[51].

The use of AlchemyAPI's services requires a paid license. Our license allows a maximum of 30,000 requests per day. This means that AlchemyAPI may not always be a usable resource for this thesis.

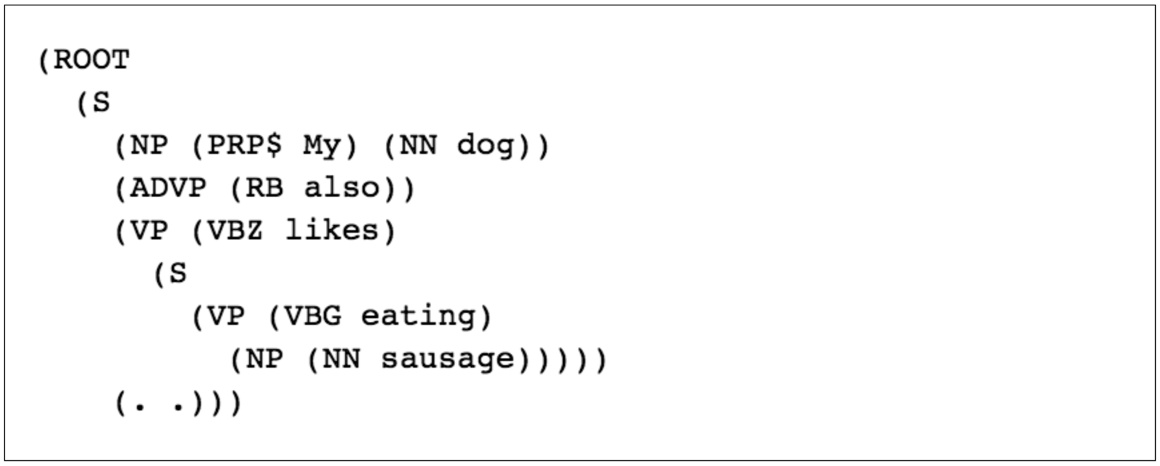
4.3.2 spaCy

spaCy is an open-source commercial Python NLP library featuring a high performance tokenizer, part-of-speech tagger, named entity recognizer, and a syntactic dependency parser[58]. It is released under the MIT license and purposed for production use[59]. This thesis uses spaCy to complement AlchemyAPI for named entity recognition. spaCy claims an NER "accuracy" of 82.6% on the OntoNotes 5 corpus, although the data behind that claim is not public[59]. Our own anecdotal analysis finds spaCy to be on par with AlchemyAPI for the most common types of entities—people, organizations, and locations—and performs better than AlchemyAPI for certain numerical expressions such as percentages. Additionally, because spaCy is a library and not a paid API, it does not have the rate limitations that AlchemyAPI has and can be considered a viable backup solution if AlchemyAPI is no longer available to us in the future.

4.3.3 Stanford Parser

A natural language parser is an application that determines the grammatical structure of sentences, such as which words comprise phrases and which words are subjects or objects of verbs. A probabilistic natural language parser uses knowledge from a set of hand-parsed sentences to produce the most likely analysis of new input sentences[22].

The Stanford Natural Language Processing Group has developed a number of probabilistic natural language parsers. Its featured version is a lexicalized parser that implements a factored product model of an unlexicalized PCFG parser and a lexicalized dependency parser, combining their preferences by efficient exact inference using an A* algorithm. This model achieved an F-measure of 0.867 on the Penn Treebank parsing test[30, 44]. In 2013, Stanford also released a Compositional Vector Grammar parser with an F-measure of 0.904 while running 20% faster than the PCFG parser[56] For this thesis, we are only concerned with the parser's constituency output. Figure 4.4 shows the constituency tree output of the parser given in the input sentence: "My dog also likes eating sausage."



```
(ROOT
  (S
    (NP (PRP$ My) (NN dog))
    (ADVP (RB also))
    (VP (VBZ likes)
      (S
        (VP (VBG eating)
          (NP (NN sausage)))))
    (. .)))
```

Figure 4.4: Tree-structured output of the Stanford lexical parser

In 2014, the Stanford NLP Group released a shift-reduce constituency parser that achieved an F-measure of 0.913 on Wall Street Journal articles from the TIPSTER corpus—better than the lexicalized version—while running several orders of magnitude faster than any other parser, including the 2007 state-of-the-art Berkeley parser and the Stanford PCFG parser. The shift-reduce parser works by keeping a state of the current parsed tree, with the sentence's words in a queue and partially completed trees on a stack. It applies transitions—shift, unary reduce, binary reduce, or finalize—to the state until the queue is empty and the stack contains only the finished tree[68]. This thesis uses the shift-reduce parser to parse the phrase structure trees of transcript sentences.

4.3.4 TONGS

TONGS[67] is an Amazon review analysis system that builds upon the SPORK text summarization pipeline[36]. After summarizing Amazon reviews through an optimized version of SPORK, TONGS uses the NLTK Python module to determine their overall sentiment, thus allowing users to quickly make purchasing decisions on the reviewed products. This thesis uses the sentiment engine of TONGS to determine the sentiment of testimonies in committee hearings[67, 36].

Unlike other sentiment engines, TONGS outputs a numerical sentiment value for each word in a sentence that it tokenizes, and if the sentence has been negated by words such as “not”, “don’t”, and “won’t”. It uses a sentiment dictionary containing lists of positive words and negative words given by Dr. Bing Liu of the University of Illinois at Chicago[35]. Users can also override this default dictionary with their own[67].

The final output of TONGS’s sentiment engine for a single sentence is the following[67]:

- The part-of-speech for each word
- The numerical sentiment value of each word
- Highest word sentiment value in the sentence
- Lowest word sentiment value in the sentence
- Overall sentiment value of the sentence
- Whether or not the sentence is negated

This detailed output helps users determine if TONGS is calculating results correctly.

4.3.5 Annotation Pipeline

The previous four subsections described the natural language processing tools used in this thesis to analyze committee hearing transcripts. This section describes how those tools are incorporated into the annotation pipeline that processes unstructured transcript text into XML documents with annotated elements that will be populated into the knowledge graph. Figure 4.5 shows an architecture diagram of the annotation pipeline.

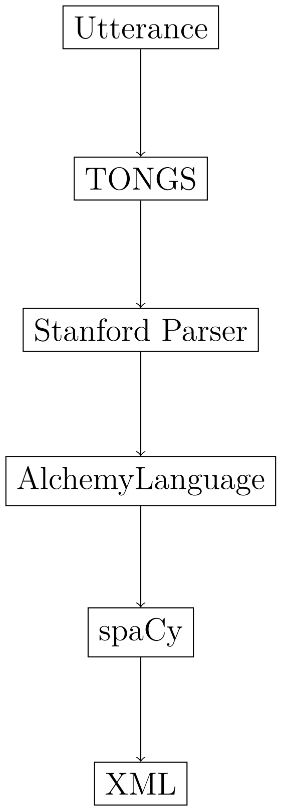


Figure 4.5: Diagram of the annotation pipeline.

To understand the annotation pipeline, consider a single hearing H. This hearing is associated with a sequence of utterances that were spoken during that hearing. Each utterance includes a unique identifier uid, a pid that refers to the id of the speaker in the `Person` table in the DDDB, the spoken text itself, and an optional alignment enum indicating whether the speaker supports or opposes the bill being discussed at the hearing.

The annotation pipeline first queries the `Hearing` table for H and creates a root

XML element tagged "hearing". Three attributes are added to the root element: a unique identifier hid corresponding to the hid from the SQL table, the date of the hearing in the format "YYYY-MM-DD", and the name of the hearing in the format "<house> <type> Committee on <topic>".

The utterances associated with H are then queried from the `Utterance` SQL table and sorted by ascending timestamp. Consider a single utterance U. A corresponding child element E, tagged "utterance", is added to the root XML element. The unique identifier uid, the speaker's `Person` identifier pid, the id of the legislative bill being discussed, and the name of the speaker are added as attributes to the child element. The text of U is copied as the text of E.

U is then processed through TONGS for sentiment analysis. U can contain multiple sentences, so TONGS parses U's text into individual sentences. For each sentence, TONGS outputs an integer sentiment value, S, and a list of words in the sentence, W, that contributed to the sentiment value. Figure 4.6 shows the output of TONGS for a sentence in an utterance spoken by Senator Carol Liu. Two additional attributes, *sentiment* and *words*, are added to E. The sentiment attribute is a list of S's for the given utterance, and the words attribute is a list of W's for the given utterance, i.e. a list of word lists. If for a given sentence, W is empty, S and W are not added to the sentiment and words attributes. However, if W contains at least one word, even if S is 0, those values are represented in the attributes.

Each sentence in U is also processed through the Stanford Shift-Reduce parser. The resulting phrase structure tree is analyzed to find noun phrases in the sentence. If a noun phrase contains a proper noun, an adjective, or a cardinal expression, the phrase is considered as important. Figure 4.7 shows the phrases detected by the parser for a different sentence in the same utterance used in Figure 4.6. The list of

```
Input: So, I wholeheartedly recommend that you approve and send his appointment
to the Senate Floor for confirmation.
Output: Sentiment: 3
        Words: ['wholeheartedly', 'recommend', 'approve']
```

Figure 4.6: TONGS output for a given sentence

important phrases for all the sentences in U is added as a phrases attribute to E.

```
Input: Obviously, he has earned the respect of his Board of Governors Colleagues
who elected him recently as Vice Chair.
Output: the respect of his Board of Governors Colleagues who elected him recently
as Vice Chair
```

Figure 4.7: Noun phrase detected by the Stanford Shift-Reduce parser for a given sentence

While this thesis is almost exclusively implemented in Python, the Shift-Reduce parser is only available in Java. It also takes a nontrivial amount of time to load the parser model, approximately 10 seconds on a 2013 Macbook Pro with a 2.3 GHz Core i7. Therefore, the annotation pipeline starts the Java Shift-Reduce parser process upon startup and allows it time to load the parser and tagger models. The annotation pipeline keeps the process running as it goes through each hearing, using the Python subprocess module to input sentences into the parser and retrieve the resulting phrase structure trees. The parser process is shut down when the pipeline has finished annotating its final hearing.

The utterance text is then run through the named entity recognizers. When a named entity is identified, the word or words comprising the entity are tagged as a child element of E, where the tag name is the classification of the entity as listed in

Section 3.2. First, the pipeline itself looks for and tags legislative bill names and religious keywords. The religious keywords are scraped from the Association of Religion Data Archives[12]. Next, the utterance is processed through AlchemyLanguage using code written by Carmen Dang, and finally the spaCy named entity recognizer. Some entities may be tagged twice, once by Alchemy and once by spaCy. This is acceptable, since the knowledge graph population pipeline will only consider the outermost tags. Figure 4.8 shows the named entities found by Alchemy and spaCy for the utterance used in the previous two examples. At this point, the utterance XML element E is finished, and the pipeline can go to the next utterance. Once all the utterances associated with H have been annotated, H has finished the annotation pipeline and its root XML element can be written to file. Figure 4.9 shows the final XML of an annotated hearing with a single utterance, the same one used in the previous examples.

Input: Obviously, he has earned the respect of his Board of Governors Colleagues who elected him recently as Vice Chair. So, I wholeheartedly recommend that you approve and send his appointment to the Senate Floor for confirmation.

Output: Board of Governors – Organization

Vice Chair – JobTitle

Figure 4.8: Named entities detected by AlchemyLanguage and spaCy for an example utterance.

As mentioned in Section 4.3.1, our key for AlchemyAPI has a rate limit of 30,000 requests per day. A single hearing usually has at least 1,000 utterances. So if one API call is made per utterance, only thirty hearings at most can be processed per day. Given the number of hearings in the SQL database, at least eight hundred as of this thesis's writing, this is an unacceptable stopgap. The solution is to concatenate the text of several utterances and make one API call per utterance group. The pipeline combines utterances into groups of 25, an arbitrarily chosen number.

```
<hearing hid="1" name="Senate Standing Committee on Senate
   Standing Committee on Rules" date="2015-01-07">
  <utterance pid="63" bid="CA_201520160AB717" name="Carol
     Liu" uid="73" sentiment="[0, 3]" words="[['respect', '
     Vice'], ['wholeheartedly', 'recommend', 'approve']]"
     phrases="['the respect of his Board of Governors
     Colleagues who elected him recently as Vice Chair', '
     the Senate Floor for confirmation']">Obviously, he has
      earned the respect of his <Organization>Board of
     Governors</Organization> Colleagues who elected him
     recently as <JobTitle>Vice Chair</JobTitle>. So, I
     wholeheartedly recommend that you approve and send his
      appointment to <Organization>the Senate Floor</
     Organization> for confirmation. Thank you for your
     consideration.</utterance>
</hearing>
```

Figure 4.9: An annotated hearing XML document with a single utterance.

4.3.6 Limitations

The initial design of the annotation pipeline included the use of Python's multiprocessing module. The plan was to spawn multiple concurrent processes, with each process annotating a different hearing. However, this module breaks when used in conjunction with the NLTK module, which is used by TONGS, and the urllib module, which is used to query AlchemyAPI. This issue has been acknowledged by NLTK's authors for over a year, but there has been no apparent progress to fixing this issue[46]. The current implementation of the annotation pipeline is limited to serial execution.

4.4 Knowledge Graph Population

Once a transcript has been fully processed by the annotation pipeline, the population pipeline can read it to populate the knowledge graph with new information. The population pipeline's first step is to gather the pids of all the speakers who testified in that hearing and create the corresponding `Testified` edges from Person nodes to the Hearing node.

Now the population pipeline can process the annotated utterances one by one. Consider a single child XML element E, representing utterance U in hearing H. Recall the attributes of E produced from TONGS: sentiment and words. Also recall that E has a *pid* attribute identifying the speaker, a *bid* attribute identifying the bill being discussed, and an *align* attribute representing the speaker's position on the bill being discussed. Using these attributes, the pipeline creates a `Sentiment` edge from the `Person` node representing the speaker to the discussed `Bill` node. The Cypher query to create this edge is shown in Figure 4.10.

```
MATCH (n:Person {pid:|pid|}), (b:Bill {bid:|bid|})
CREATE (n)-[:sentiment {val:|sentiment|, words:|words|, align:|align|}]->(b)
```

Figure 4.10: Neo4j Cypher query to create a Sentiment edge

After the `Sentiment` edge for E is created, the population pipeline looks for tagged child elements, the named entities, in E's text. For each entity, the pipeline creates a `Said` edge from the `Person` node representing the speaker to the node representing that entity, creating the latter node if it does not already exist, and using the same attributes of E that were used to create `Sentiment` edges. The same is done for noun phrases in E's phrases attribute. If a named entity is the name of a person, the pipeline instead tries to find the corresponding Person node in the graph, and if found, uses that node as the target node for the edge. Figure 4.11 shows the corresponding Cypher query for creating a `Said` edge to a named entity.

```
MERGE (e:|entityTag| {name:|entity|})


MATCH (n:Person {pid:|pid|}), (e:|entityTag| {name:|entity|})
CREATE (n)-[r:said {align:|align|, bid:|bid|, sentiment:|sentiment|,
                words:|words|}]->(e)
```

Figure 4.11: Neo4j Cypher query to create a Said edge for a named entity or noun phrase

Chapter 5

VALIDATION

Validation of this thesis is done in two stages. The first stage evaluates the turnaround time of creating and updating the knowledge graph. We measure the performance of the knowledge graph pre-population, the annotation of all the transcripts from the existing database, and the population of the knowledge graph from those annotations. We then examine the overall structure of the graph, counting the numbers of each node and edge type. Finally, we evaluate the turnaround time of annotating a single day's worth of hearings and updating the knowledge graph from those results.

In the second stage of validation, we conduct several case studies to demonstrate the type of information that can be retrieved from the knowledge graph. A case study consists of an analytical question involving the testimonies in committee hearings, and the data that is queried from the knowledge graph to answer this question. For each case study, we show the specific Neo4j Cypher statement used to query the knowledge graph, measure the performance time of that query, and format the data in a chart or image.[1] Our case studies cover a variety of speakers and topics, and cannot be answered from just the existing relational database. Similarly, the knowledge graph data queried for these case studies is not present in the relational database. We present the following case studies:

- Locations of residency for members of the general population opposed to vaccination requirements. We gather the cities of origin for individuals in the anti-vaccine community and plot the results on a map of California.

- Sentiments of legislators toward Bay Area companies and cities. We plot

[1]These images were made in color; information may be lost when printed in black and white.

the sentiment scores of individual companies and cities per legislator onto a heatmap.

- Sentiments of legislators with regards to line item budgets. We plot the sentiment scores of legislators discussing line item budgets regarding a specific topic onto a bar chart.
- General sentiments of legislators along party lines. We plot the overall sentiment of each legislator given everything he's ever said in a committee hearing onto a bar chart.

5.1 Environment

The tests in this section were run on a Google Compute Engine virtual machine running Debian 8 with 2 vCPUs, 16 GB of RAM, and a 15 GB SSD. Neo4j was given 2 GB of heap space and 2GB of page cache memory for mapping the store files discussed in Section 4.1.

5.2 Turnaround Time

In this section, we analyze the turnaround time of creating the knowledge graph from scratch and of updating the knowledge graph with one day of new committee hearings. The results of this evaluation are summarized in Table 5.1. The graph prepopulation from the existing database only takes one minute and sixteen seconds. The most time-consuming step is the transcript annotation. When this evaluation was run, the Digital Democracy database contained 1009 committee hearings spanning from the beginning of 2015 to April 19, 2016. The total time to process all 1009 hearings into annotated XML documents is 246 minutes 27 seconds, or just over 4 hours. This means each hearing takes, on average, 14.65 seconds to be annotated.

Table 5.1: Turnaround time for creating the knowledge graph from scratch

Step	Time (min)
Pre-population	1.26
Transcript Annotation	246.46
Population	54.30
Total	302.02

The graph population from all the transcripts took 54 minutes 18 seconds, or 3.23 seconds per hearing. These processing times add up to just over 5 hours. This means the knowledge graph can be easily created from scratch overnight.

This evaluation is important because of the many possible reasons for re-annotating the hearing transcripts. NLP libraries such as AlchemyLanguage and spaCy may be frequently updated, and major updates may significantly affect their output, and by extension, the named entities that are found. Depending on the importance of the update, this may require a full reprocessing of all transcripts, and therefore a complete rebuilding of the knowledge graph. The 5 hour turnaround time makes it feasible for the knowledge graph to be completely rebuilt from scratch overnight.

Once the knowledge graph is populated, we can gather some statistical data regarding the nodes and edges of the graph. Table 5.2 and 5.3 show the numbers of nodes and edges in the graph. The final size of the knowledge graph on disk is 1.3 GB.

A single day in the California state legislature usually holds a single-digit number of committee hearings. The highest number of hearings held in a day during 2015 was 12. Using the processing times above 12 hearings takes 175.8 seconds, or just under 3 minutes. Populating the graph with those hearings takes 38.76 seconds. This leads to a total turnaround time of approximately 3.5 minutes for a day's worth of

Table 5.2: **Number of nodes in the knowledge graph by type. Node counts for named entity types that are less than 1000 are not shown, but included in the Total count.**

Node Type	Count
Public	16166
Committee	346
Assemblymember	240
City	3374
Phrase	819180
JobTitle	4218
Hearing	1124
PersonName	16169
Topic	2075
Senator	112
Location	2865
Quantity	3140
Facility	3339
GeographicFeature	1386
Organization	44975
Lobbyist	2669
Money	5045
Time	11554
Bill	31649
Total	975085

Table 5.3: Number of edges in the knowledge graph by type.

Edge Type	Count
HeldHearing	1124
Sentiment	206158
LobbiesFor	2895
CoChair	2
ViceChair	18
Testified	43481
Vote	274410
Member	3033
Chair	172
Said	1471463
Total	2002855

hearings. These results are summarized in table 5.4.

Note that 12 was the highest number of hearings for a single day in 2015. The average number of hearings per day is 5, leading to a total turnaround time of 89.4 seconds. This means the day-to-day updating of the knowledge graph can be incorporated into the daily Digital Democracy update system without significantly increasing its turnaround time.

Table 5.4: Turnaround time for updating the knowledge graph with a single day of committee hearings.

Step	Time (s)
Transcript Annotation	175.80
Population	38.76
Total	214.56

5.3 Case Studies

To determine if the knowledge graph provides useful information, we performed a number of case studies targeting various persons, bills, and topics.

5.3.1 Legislative Bill SB 277

In 2015, the California State Senate passed SB 277, a bill that removed the personal belief exemption for vaccination requirements required by public schools and daycare centers. When the bill was in committee, the anti-vaccine community of California traveled to Sacramento to voice their opposition. Individuals were required to state their name, where they are from, and whether they support or oppose the bill. Those names and locations are annotated and populated into the knowledge graph, so we can retrieve those locations using the Cypher query in Figure 5.1, which took 1536 ms to execute. These locations are then converted into a geographical heatmap of California, shown in Figure 5.2. As expected, most of the represented anti-vaccine community came from large urban cities like San Francisco and Los Angeles.

```
MATCH (n:Public)-[r:said {bid:'CA_201520160SB277', align:'against'}]->(e)
WHERE e:City or e:Location
RETURN e.name
```

Figure 5.1: Neo4j Cypher query to retrieve the places of origin of the general population opposed to SB 277

In this case study, we analyzed the testimony of the general public with regards to a controversial vaccination bill and extracted the names of the cities that individual testifying speakers reside in, and plotted those cities onto a geographical heatmap.

This pattern analysis can be altered to target any desired bill.

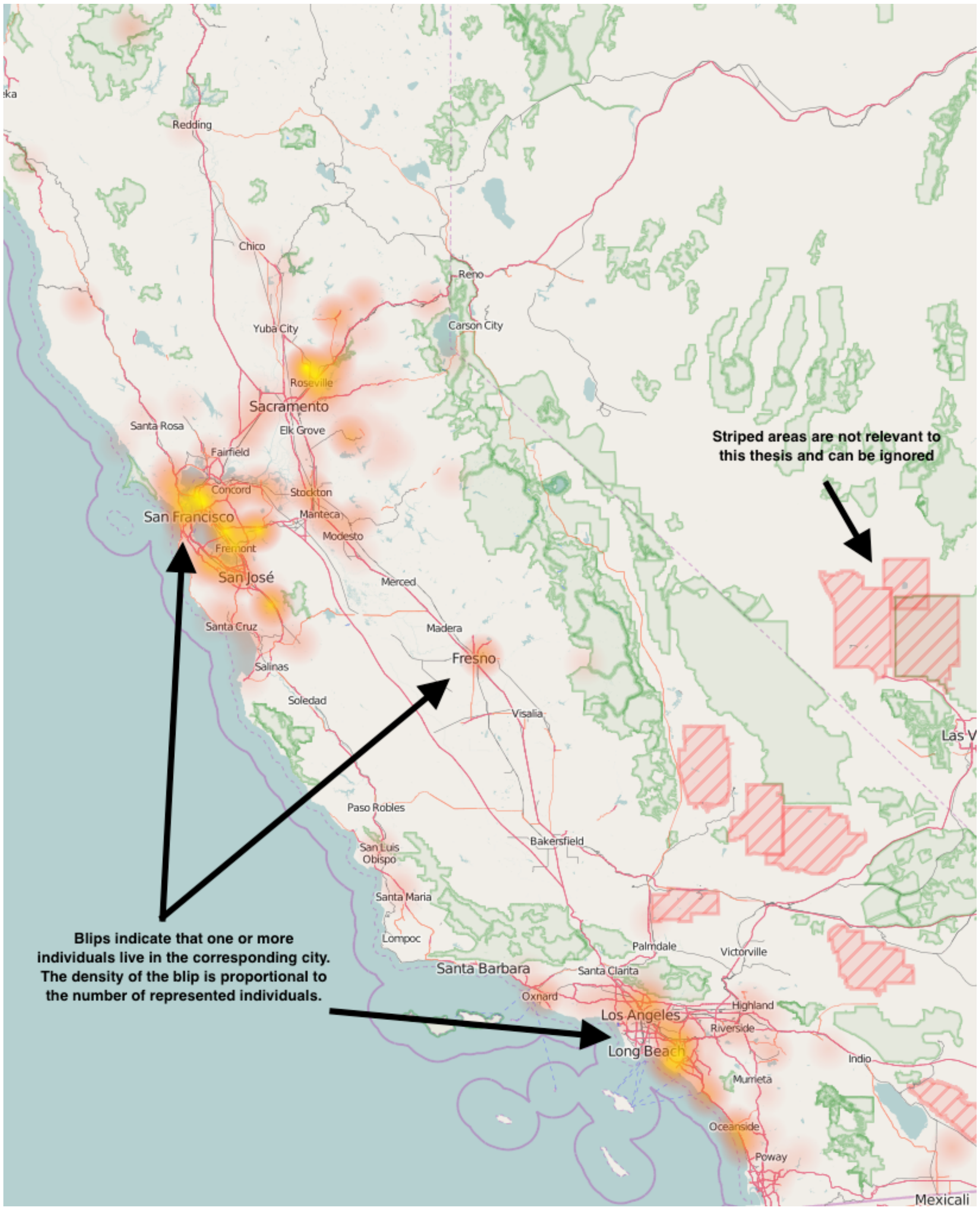


Figure 5.2: Heatmap of places of origin of the general population opposed to SB 277

5.3.2 Sentiments of legislators toward Bay Area cities

```
MATCH (n:Legislator)-[r:said]->(e:City {name:|cityname|})
RETURN n.first, n.last, e.name, r.sentiment
```

Figure 5.3: Neo4j Cypher query to retrieve the sentiments of legislators toward a given city

This case study analyzes the sentiments of legislators towards cities in the Bay Area. The Cypher query in Figure 5.3 is run for each Bay Area city, with each query taking approximately 800-1000 ms to execute, and the results are aggregated into the heatmap shown in Figure 5.4. A small sample of the heatmap is shown in Figure 5.5. Green slashed squares represent positive sentiments, and red squares with x's represent negative sentiments.

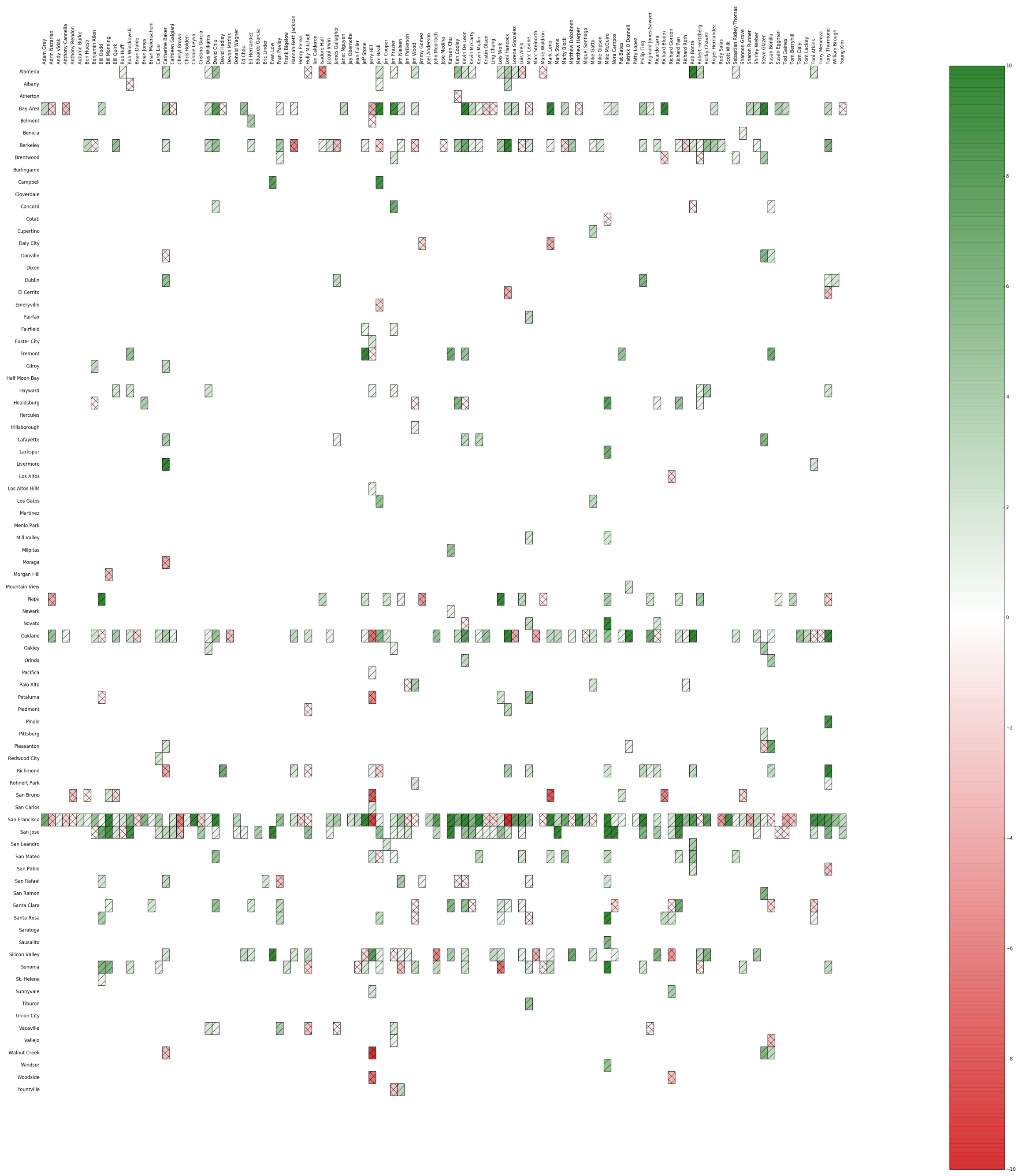


Figure 5.4: Heatmap of sentiments of legislators towards Bay Area cities

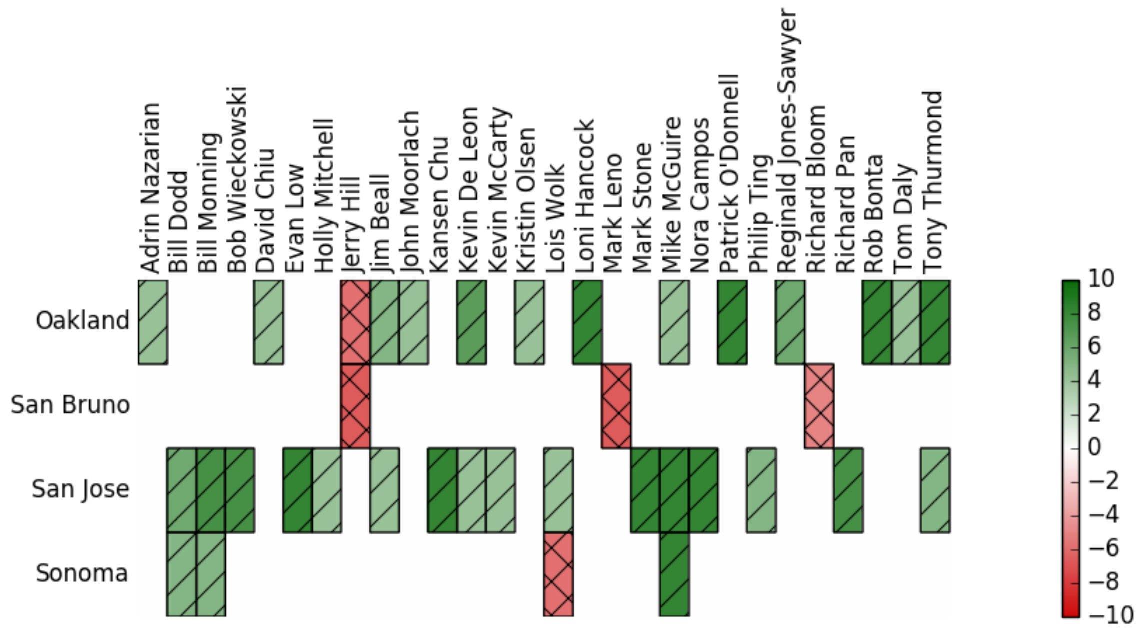


Figure 5.5: A small sample of the heatmap of legislator sentiments towards Bay Area cities

While this case study was meant to provide insight into the sentiments individual legislators, it also provides insight to into overall sentiments toward specific cities. For example, the sentiment toward San Bruno is largely negative. By querying the noun phrases spoken by these legislators towards San Bruno, we see that they were talking about the 2010 San Bruno PG&E pipeline explosion. Figure 5.6 and Table 5.5 show the Cypher query and results for Senator Jerry Hill's sentiment towards San Bruno.

```
MATCH (n:Senator {pid:66})-[r:said]->(e:Phrase)
WHERE e.name contains 'San Bruno'
RETURN r.sentiment, e.name
```

Figure 5.6: Neo4j Cypher query to retrieve the noun phrases spoken by Senator Jerry Hill regarding the city of San Bruno.

Table 5.5: Phrases spoken by Senator Jerry Hill regarding San Bruno.

Sentiment	Phrase
-2	the 2010 natural gas explosion in the city of San Bruno
-3	the grandfather clause concerning how to set a maximum pipeline pressure more than four years after the San Bruno disaster
-1	the 2010 natural gas explosion in my district in San Bruno
-1	a safety policy after the San Bruno explosion
1	the San Bruno explosion
1	the 2010 gas pipeline explosion in San Bruno
-2	the 2010 explosion in San Bruno that killed eight
1	the incineration of eight individuals living in San Bruno
-3	if the Franchise Tax Board attempts to deny PG and E's deduction of the San Bruno penalty, PG and E
2	the explosion in San Bruno

Each square in the heatmap is the sum of the sentiments of all utterances spoken by a legislator that contain the given city. Therefore, darker shades of green and red can either mean that the legislator has spoken multiple times about the city in question, or the legislator has spoken very strongly about that city. This ambiguity reduces the confidence of this case study, and perhaps averages would have resulted in a more accurate analysis instead of summations. Nevertheless, the heatmap is evidence of useful, actionable data retrieved from the knowledge graph.

5.3.3 Sentiments of legislators toward Bay Area companies

This case study is very similar to the previous case study, except instead of targeting cities, it targets companies. The resulting heatmap is shown in Figure 5.7 and 5.8. While the heatmap suggests that the overall sentiment towards Bay Area companies is largely positive, further analysis shows that this isn't the case. The heatmap says that Mike McGuire has a very positive sentiment towards Airbnb, however Mr. McGuire has actually been advancing legislation to require Airbnb and similar vacation-rental companies to pay their fair share of taxes and abide by local ordinances[54]. This suggests that some of the sentiment values given by TONGS in this case may be inaccurate.

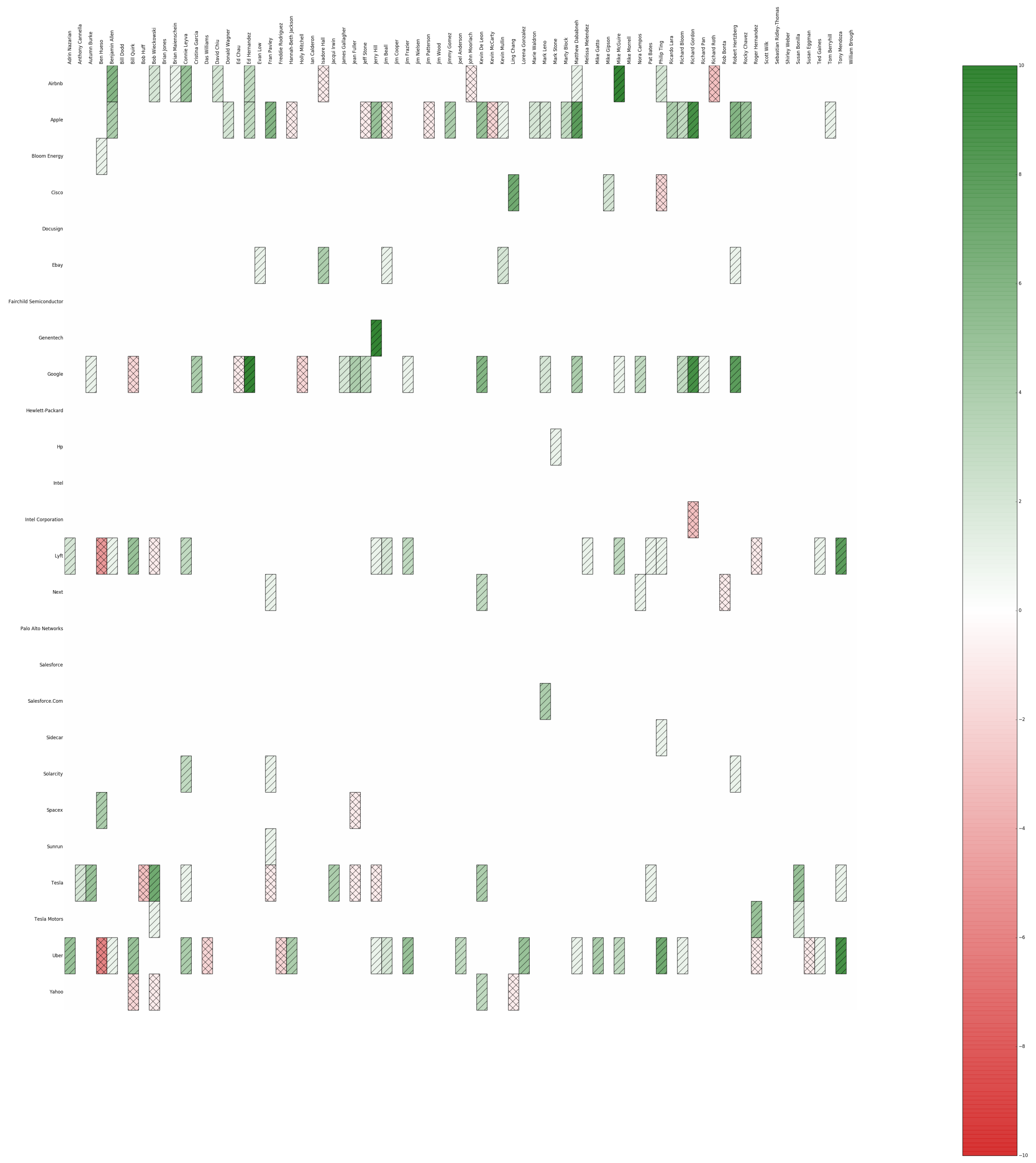


Figure 5.7: Heatmap of sentiments of legislators towards Bay Area companies

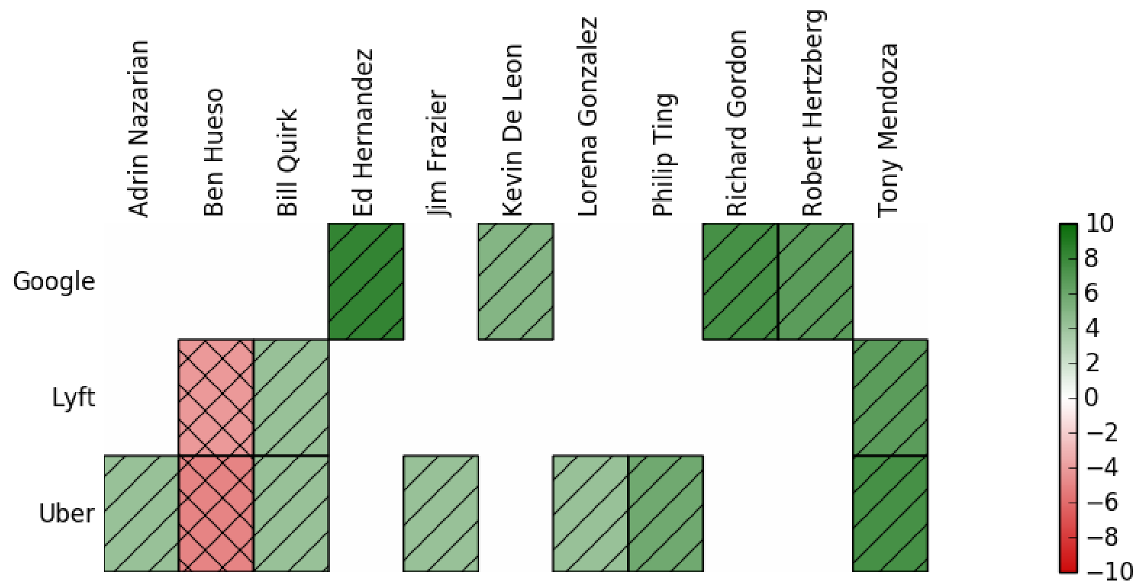
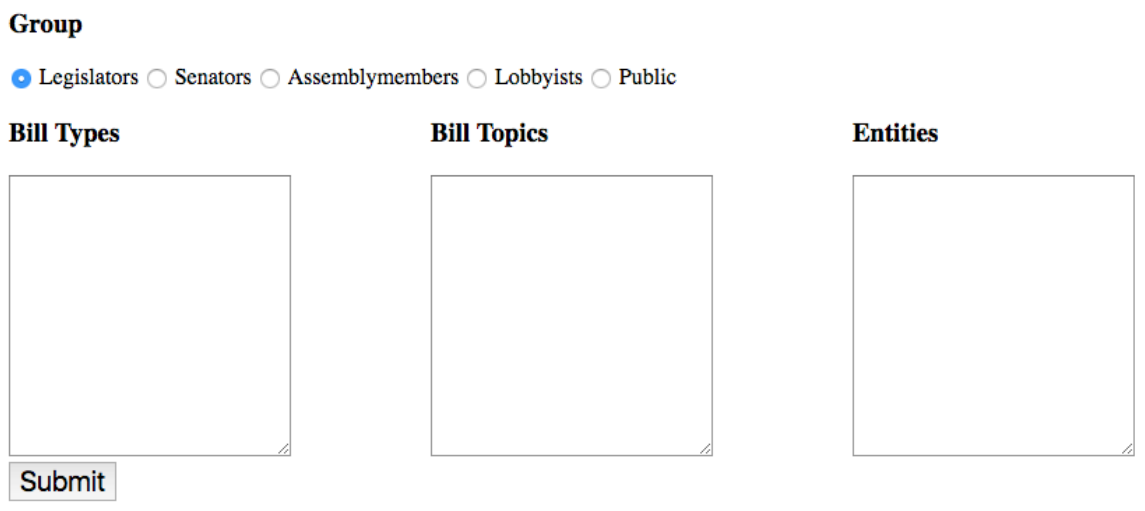


Figure 5.8: Heatmap sample of sentiments of legislators towards Bay Area companies

5.3.4 Prototype Client


Group

[✓ Legislators] Senators Assemblymembers Lobbyists Public

Bill Types **Bill Topics** **Entities**

Figure 5.9: Screenshot of first section of prototype client

For the following case studies, we created a prototype web client that queried the knowledge graph using parameters set by the user and displayed results in either a heatmap, a line graph, or plain text, depending on the query.

The client is split into two sections. The first section, shown in Figure 5.9, takes the following inputs:

- Which type of speaker to query: legislators, lobbyists, etc.
- Types of bills to query: AB for Assembly bills, SB for Senate bills, etc.
- Bill topics to query: Housing, University, etc. The client looks at the *subject* property of `Bill` nodes for these topics.
- Optional list of entities, or keywords, to look for: San Jose, John Smith, etc.

Once the user fills in those inputs and clicks Submit, the client first queries the graph for the bills with the specified bill types and topics. Figure 5.10 shows the Cypher query for retrieving bills with the type AB or SB and with the topics University or Education. Then, the client queries for `Said` edges whose destination nodes are entities with names matching the entities specified by the user. Figure 5.11 shows an example with the entities 'Ford' and 'Tesla'. Once the edges are retrieved, the client processes only the edges with a *bid* property that was found in the first query, in Figure 5.10. Sentiment values are summed and organized by speaker and entity.

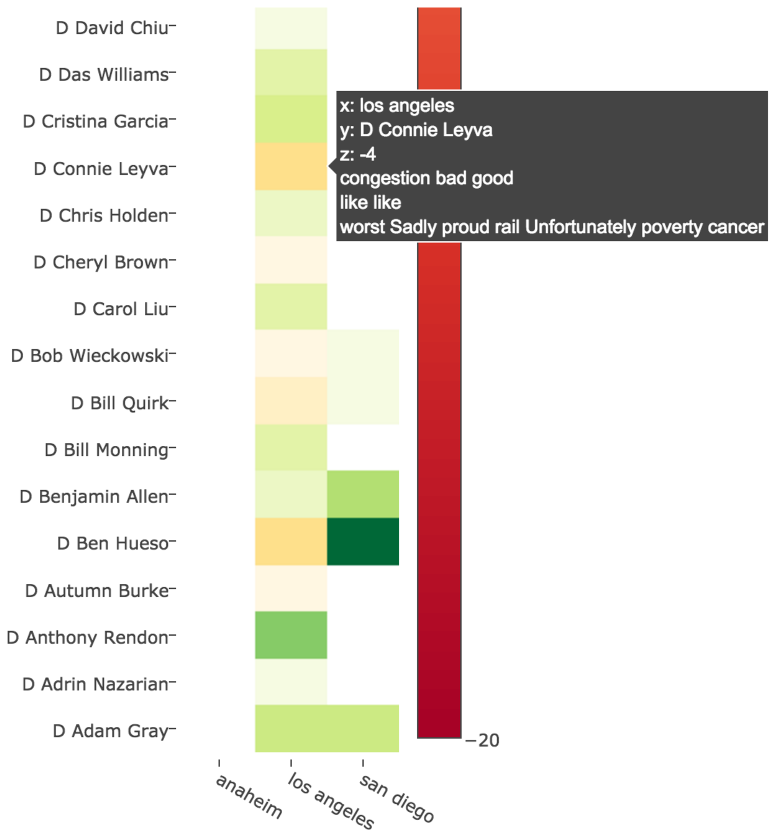
```
MATCH (b:Bill)
WHERE (b.name starts with 'AB' OR b.name starts with 'SB')
    and (b.subject contains 'University' or b.subject contains 'Education')
RETURN b
```

Figure 5.10: Neo4j Cypher query used by first section of client to retrieve bills

```
MATCH (n:Senator)-[r:said]->(e)
WHERE not e:Phrase and e.name in ['Ford', 'Tesla']
RETURN n, r, e
```

Figure 5.11: Neo4j Cypher query used by first section of client to retrieve speaker sentiments toward given entities

Finally, the client displays the results in a heatmap similar to those in the Bay Area case studies. When the mouse is hovered over a square, the client displays the words from the corresponding utterances that contributed to the sentiment value. Figure 5.12 shows a portion of a heatmap generated for legislator's sentiments toward Los Angeles, Anaheim, and San Diego. The use of summations instead of averages or another aggregation formula reduces the confidence of these results, just like in Section 5.3.2.



x: los angeles
y: D Connie Leyva
z: -4
congestion bad good
like like
worst Sadly proud rail Unfortunately poverty cancer

Figure 5.12: Screenshot of a heatmap presented by the client for sentiments towards Southern California cities

If the user did not specify any entities, the client instead queries for `Sentiment` edges, as shown in Figure 5.13. The client processes only the edges whose destination nodes were found in the first query for retrieving bills. The sentiment values aggregated and grouped by speaker, using averages instead of sums. The results are displayed in a bar chart, with the left-most x values representing Democrats and the right-most x values representing Republicans. Hovering the mouse over a bar displays the name of the speaker corresponding to that sentiment value. The client also logs the utterance words that contributed to the sentiment values in the web debugging console. An example of the client's bar chart functionality is shown in Section 5.3.5.

```
MATCH (n:Senator)-[r:sentiment]->(b:Bill)
RETURN n, r, b
```

Figure 5.13: Neo4j Cypher query used by first section of client to retrieve speaker sentiments when discussing bills

The second section, shown in Figure 5.14, takes one of the following inputs:

- The name of a speaker
- A named entity

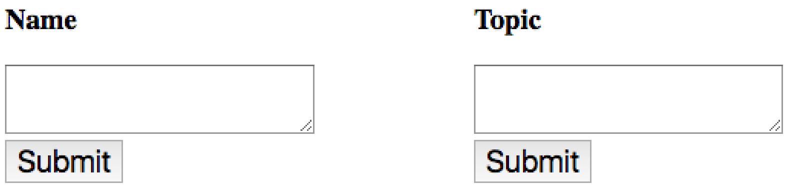


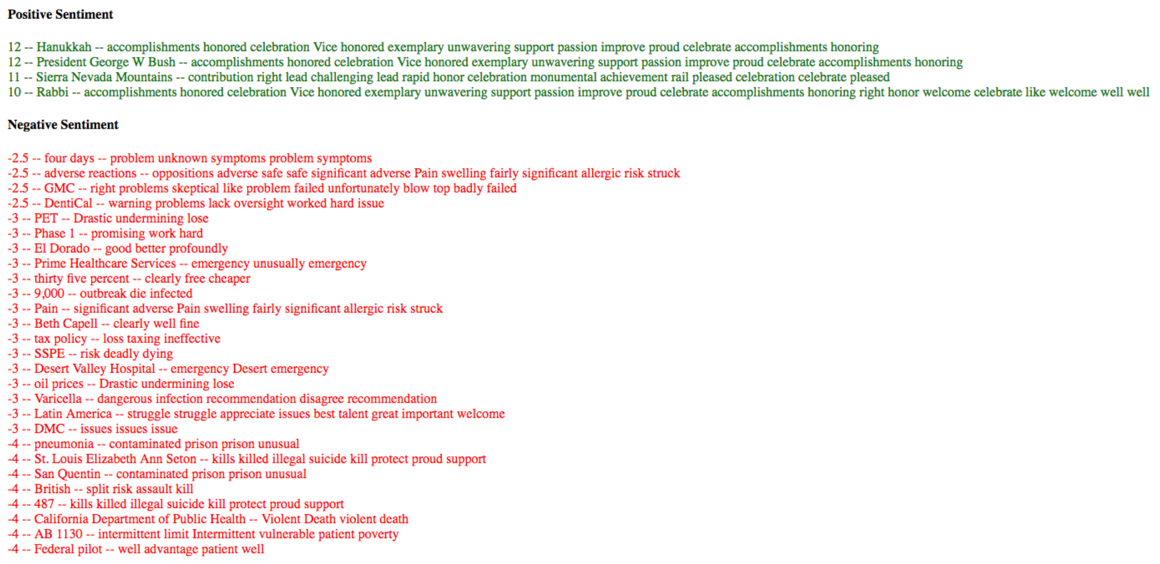
Figure 5.14: Screenshot of second section of prototype client

If the user inputs a speaker name, the client performs the query in Figure 5.15 to retrieve `Said` edges whose start nodes match the inputted speaker name. The sentiment values of those edges are aggregated by named entity using averages, and sorted from highest positive value to lowest negative value. The client then outputs the pairs with the three highest sentiment values, and the pairs with the three lowest values, along with the words from the corresponding utterances that contributed to those sentiment values.

```
MATCH (n:Person {first:|firstname|, last:|lastname|})-[r:said]->(e)
WHERE not e:Phrase and not e:Ordinal and not e:Time
RETURN r, e
```

Figure 5.15: Neo4j Cypher query used by second section of client to retrieve the named entities spoken by a given person

These results are good candidates for integration into the Digital Democracy website. Each speaker page can include a section showing the topics that the speaker likes the most, and topics that the speaker dislikes the most. Figure 5.16 shows the client's output for Senator Richard Pan. Notice the entities with the highest sentiments all have the same sentiment words, suggesting that the entities all came from a single utterance. In this case, Senator Pan is honoring a rabbi. The entities with the lowest sentiments mostly relate to health, suggesting that he has strong opinions on health-related issues.



```
Positive Sentiment

12 -- Hanukkah -- accomplishments honored celebration Vice honored exemplary unwavering support passion improve proud celebrate accomplishments honoring
12 -- President George W Bush -- accomplishments honored celebration Vice honored exemplary unwavering support passion improve proud celebrate accomplishments honoring
11 -- Sierra Nevada Mountains -- contribution right lead challenging lead rapid honor celebration monumental achievement rail pleased celebration celebrate pleased
10 -- Rabbi -- accomplishments honored celebration Vice honored exemplary unwavering support passion improve proud celebrate accomplishments honoring right honor welcome celebrate like welcome well well

Negative Sentiment

-2.5 -- four days -- problem unknown symptoms problem symptoms
-2.5 -- adverse reactions -- oppositions adverse safe safe significant adverse Pain swelling fairly significant allergic risk struck
-2.5 -- GMC -- right problems skeptical like problem failed unfortunately blow top badly failed
-2.5 -- DentiCal -- warning problems lack oversight worked hard issue
-3 -- PET -- Drastic undermining lose
-3 -- Phase 1 -- promising work hard
-3 -- El Dorado -- good better profoundly
-3 -- Prime Healthcare Services -- emergency unusually emergency
-3 -- thirty five percent -- clearly free cheaper
-3 -- 9,000 -- outbreak die infected
-3 -- Pain -- significant adverse Pain swelling fairly significant allergic risk struck
-3 -- Beth Capell -- clearly well fine
-3 -- tax policy -- loss taxing ineffective
-3 -- SSPE -- risk deadly dying
-3 -- Desert Valley Hospital -- emergency Desert emergency
-3 -- oil prices -- Drastic undermining lose
-3 -- Varicella -- dangerous infection recommendation disagree recommendation
-3 -- Latin America -- struggle struggle appreciate issues best talent great important welcome
-3 -- DMC -- issues issues issue
-4 -- pneumonia -- contaminated prison prison unusual
-4 -- St. Louis Elizabeth Ann Seton -- kills killed illegal suicide kill protect proud support
-4 -- San Quentin -- contaminated prison prison unusual
-4 -- British -- split risk assault kill
-4 -- 487 -- kills killed illegal suicide kill protect proud support
-4 -- California Department of Public Health -- Violent Death violent death
-4 -- AB 1130 -- intermittent limit Intermittent vulnerable patient poverty
-4 -- Federal pilot -- well advantage patient well
```

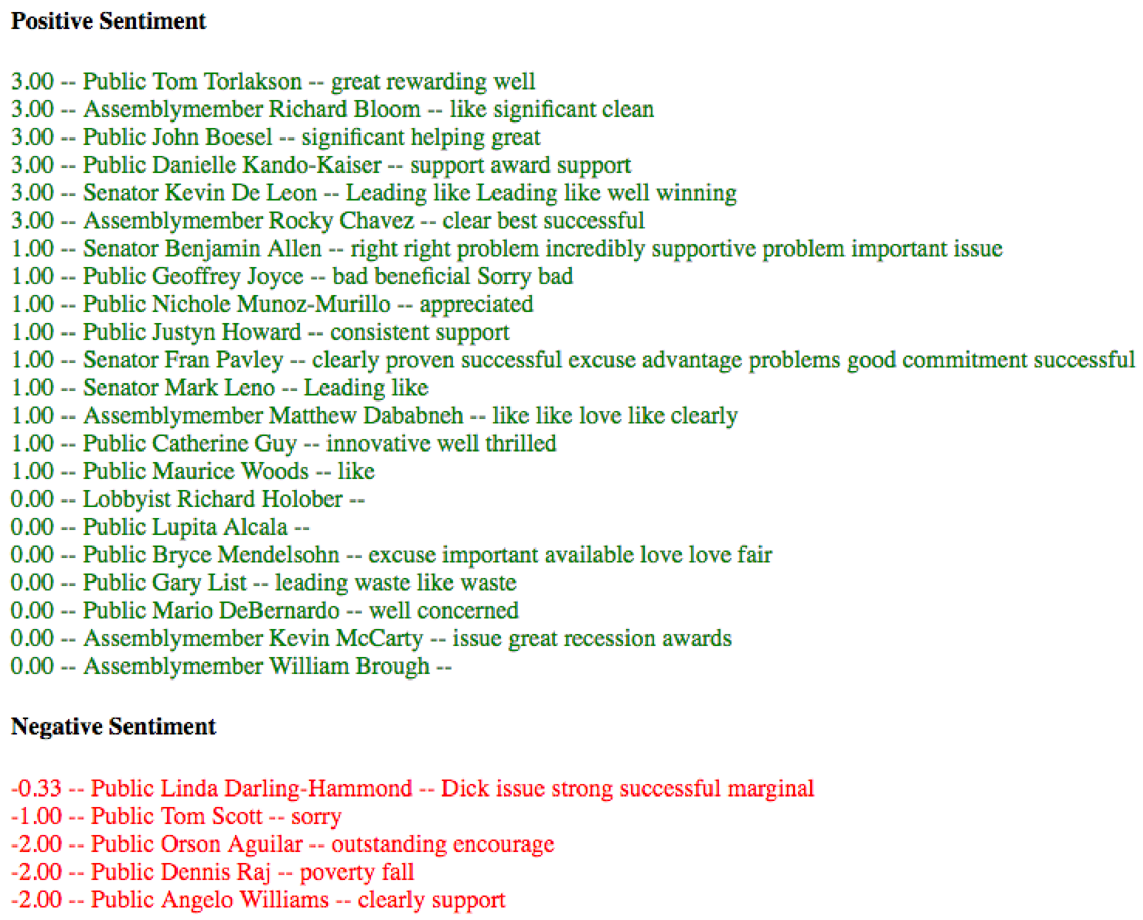
Figure 5.16: Screenshot from second section of the client showing the entities that Senator Richard Pan likes and dislikes the most

If the user inputs a named entity, the client performs the query in Figure 5.17 to retrieve `Said` edges whose destination nodes match the inputted entity. The sentiment values of those edges are aggregated by named entity using averages, and sorted from highest positive value to lowest negative value. The client then outputs the pairs with the three highest sentiment values, and the pairs with the three lowest values, along with the words from the corresponding utterances that contributed to those sentiment values.

```
MATCH (n:Person)-[r:said]->(e {name:|name|})
WHERE not e:Phrase
RETURN n, r
```

Figure 5.17: Neo4j Cypher query used by second section of client to retrieve the speakers who spoken a given entity

This output represents the speakers who like the entity the most, and the speakers who dislike it the most. Figure 5.18 shows the client's output for the entity Apple, suggesting a largely positive sentiment towards the company.



Positive Sentiment

3.00 -- Public Tom Torlakson -- great rewarding well
3.00 -- Assemblymember Richard Bloom -- like significant clean
3.00 -- Public John Boesel -- significant helping great
3.00 -- Public Danielle Kando-Kaiser -- support award support
3.00 -- Senator Kevin De Leon -- Leading like Leading like well winning
3.00 -- Assemblymember Rocky Chavez -- clear best successful
1.00 -- Senator Benjamin Allen -- right right problem incredibly supportive problem important issue
1.00 -- Public Geoffrey Joyce -- bad beneficial Sorry bad
1.00 -- Public Nichole Munoz-Murillo -- appreciated
1.00 -- Public Justyn Howard -- consistent support
1.00 -- Senator Fran Pavley -- clearly proven successful excuse advantage problems good commitment successful
1.00 -- Senator Mark Leno -- Leading like
1.00 -- Assemblymember Matthew Dababneh -- like like love like clearly
1.00 -- Public Catherine Guy -- innovative well thrilled
1.00 -- Public Maurice Woods -- like
0.00 -- Lobbyist Richard Holober --
0.00 -- Public Lupita Alcala --
0.00 -- Public Bryce Mendelsohn -- excuse important available love love fair
0.00 -- Public Gary List -- leading waste like waste
0.00 -- Public Mario DeBernardo -- well concerned
0.00 -- Assemblymember Kevin McCarty -- issue great recession awards
0.00 -- Assemblymember William Brough --

Negative Sentiment

-0.33 -- Public Linda Darling-Hammond -- Dick issue strong successful marginal
-1.00 -- Public Tom Scott -- sorry
-2.00 -- Public Orson Aguilar -- outstanding encourage
-2.00 -- Public Dennis Raj -- poverty fall
-2.00 -- Public Angelo Williams -- clearly support

Figure 5.18: Screenshot from second section of the client showing the speaker sentiments towards the company Apple

5.3.5 Sentiments of legislators towards line item budgets

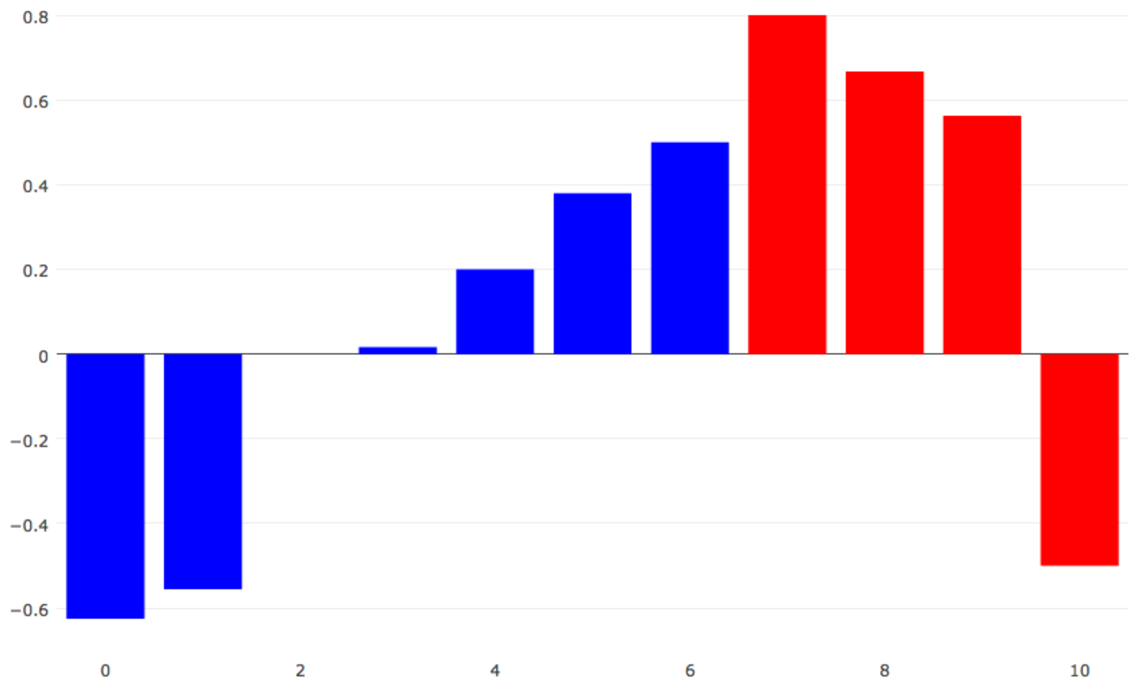


Figure 5.19: Bar chart representing the sentiments of legislators when discussing the line item budget for California State University

This case study uses the bar chart functionality of the client to chart legislator sentiments towards line item budgets by topic. Figure 5.19 shows the graph representing the average sentiment of each legislator when discussing the line item budgets for the California State University system. The Cypher query to retrieve the relevant bills took 69 ms to execute, and the query for retrieving the sentiment values and processing its results took 4458 ms to execute. Figure 5.20 shows the graph representing average sentiments for transportation and high-speed rail authorities. The Cypher query to retrieve the relevant bills took 94 ms to execute, and the query for retrieving the sentiment values and processing its results took 4956 ms to execute. These bar charts provide actionable data by surfacing the speakers that should be investigated further by looking at the Digital Democracy website's speaker pages and committee hearing transcripts, or by querying the knowledge graph for phrases relating to the line item budgets that were said by those speakers.

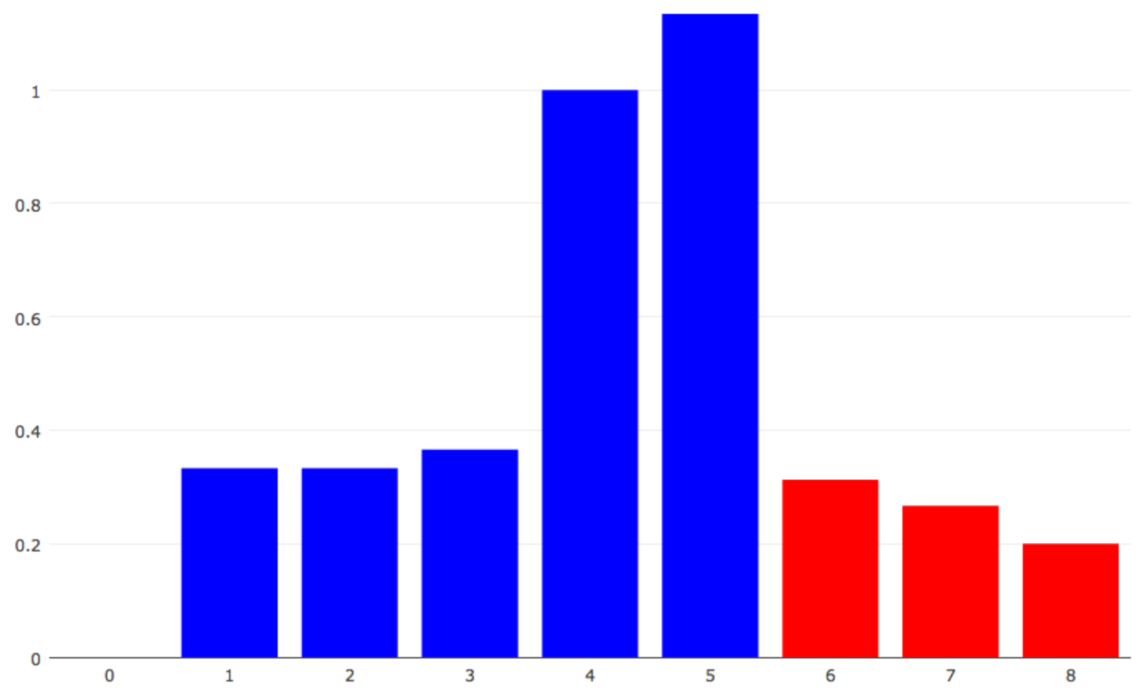


Figure 5.20: Bar chart representing the sentiments of legislators when discussing the line item budgets for transportation and rail authorities

5.3.6 General sentiments of legislators along party lines

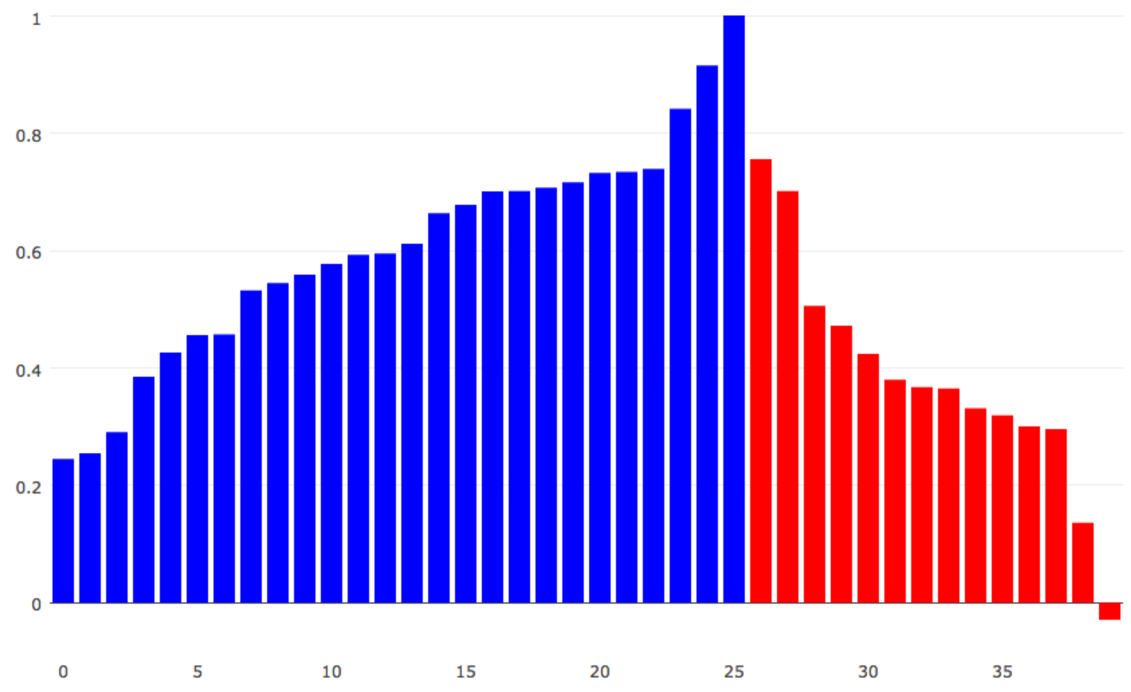


Figure 5.21: Bar chart representing the sentiments of Senators

This case study uses the bar chart functionality of the client to chart legislator sentiments across all Assembly bills, Senate bills, and budget line items. Figure 5.21 shows the graph representing the average sentiments of Senators, and Figure 5.22 shows the graph representing the average sentiments of Assemblymembers. The time to query the sentiment values from the knowledge graph and process the results took 64.2 seconds for Senators and 57.2 seconds for Assemblymembers. Interestingly, Republican Senator Bob Huff is the only legislator with a negative average sentiment at -0.02. This can serve as a starting point for users to look into more detail at Senator Huff's arguments in committee hearings.

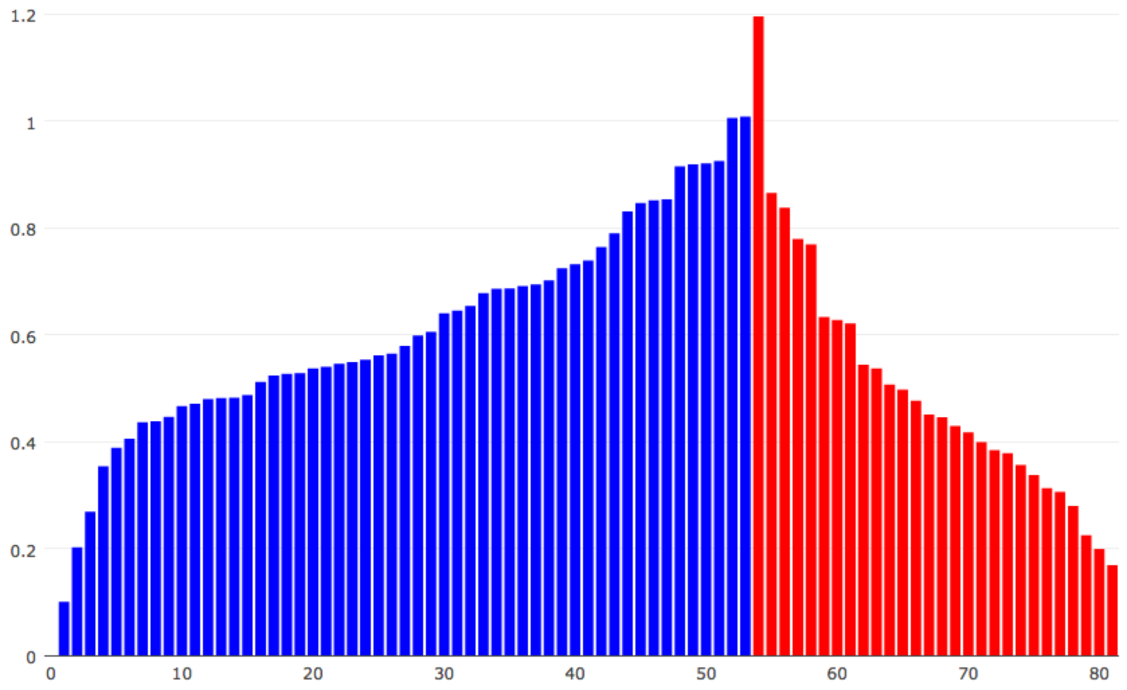


Figure 5.22: Bar chart representing the sentiments of Assemblymembers

5.4 Comparison With Other Knowledge Representation Methods

Table 5.6 outlines the advantages of representing the transcript annotation data in a Neo4j-backed knowledge graph as compared to more traditional knowledge representation methods. The use of a graph database to form a knowledge graph means that relations between speakers and entities have as much priority as the speakers and entities themselves, without needing additional steps such as foreign keys which add complexity to both the data schema and data retrieval. The graph data model intuitively fits the SKEWER problem domain of storing person-said-thing triples. By comparison, the rigid schema of a relational database would require multiple implicit connections between various tables to relate speakers with their spoken entities and sentiment values. Both relational models and NoSQL models such as key-value stores and column-oriented stores would also require full scans of their respective data

Table 5.6: Comparison of Neo4j-backed Knowledge Graph (KG) method with other knowledge representation methods for SKEWER data.

Feature	Neo4j KG	KG	NoSQL	Relational
Intuitive Data Model	Yes	Yes	No	No
Flexible Data Model	Yes	Yes	Yes	No
Ease of Data Addition	Yes	Yes	Yes	No
Fast Data Access	Yes	No	No	No
Simple Query Language	Yes	No	No	No
Expressive Query Language	Yes	No	No	No

structures to look for specific entities or bills.

Neo4j also brings advantages to SKEWER not offered by other graph databases. Index-free adjacency allows Neo4j graph traversal speed to be unaffected by the size of the graph itself. Therefore, the speed of analysis of speakers, sentiments, and entities is not affected by the addition of new committee hearings into the graph. The declarative Cypher query language combines the familiarity of SQL with the intuitive node-edge-node graph traversal pattern, allowing users of various backgrounds to quickly understand the data model of SKEWER and make simple yet powerful graph queries. The simplicity of the graph queries also means that the input parameters in a potential client user interface, such as bill names, speaker names, and entity names, directly match the parameters used in those graph queries with only minor additional query syntax required.

Chapter 6

CONCLUSION

This thesis presents a method for creating a knowledge graph from legislative committee hearing transcripts using a suite of NLP tools to annotate named entities and provide sentiment values, allowing Digital Democracy users to gain information regarding the opinions of various individuals towards specific bills, topics, and other people.

The contributions of this thesis are the following:

- **A method for annotating Digital Democracy's textual transcripts into XML documents using NLP tools:** The transcripts from Digital Democracy are currently unstructured data that are only represented as supplemental information for committee hearings. This thesis presented a pipeline for running a combination of NLP tools on the individual sentences of these transcripts to extract named entities, noun phrases, and emotions, and outputs these results as XML documents.

- **A method for populating a knowledge graph from annotated transcript XML documents:** This thesis presented the design and population of a knowledge graph from the newly created annotations of the Digital Democracy transcripts. This knowledge graph represents the testimonies spoken by legislators, lobbyists, and the general population.

- **Case studies using the knowledge graph to generate insights regarding committee hearing testimonies:** This thesis presented a number of case studies for which the knowledge graph is queried to provide analysis.

- **Prototype client for querying the knowledge graph by person name or topic:** Finally, this thesis presented a prototype client for users to analyze sentiment values for a particular person or topic.

6.1 Future Work

There are several areas of future work, including improvement of results from the NLP libraries and an interface for querying the knowledge graph from the Digital Democracy website.

6.1.1 Sentiment Analysis

As seen in Section 5.3, the sentiment values are not always accurate. This is not just a problem with TONGS, but with sentiment analysis in general, and much research is still be done on this topic.

This thesis did not define a concrete algorithm for aggregating multiple sentiment values for a given speaker and topic into a final value. Sums were used in some case studies, while averages were used in others. A tested formula will improve the confidence of every case study in this thesis as well as the overall accuracy of the knowledge graph.

6.1.2 Exploration of other NLP Libraries

The limitations of AlchemyAPI have been discussed in Section 4.3.6. As research into named-entity recognition and NLP in general progresses, other libraries may be better served for this thesis than AlchemyAPI and spaCy. Investigating these libraries and possibly replacing one or more of the libraries used in this thesis may improve the quality of the entities annotated, and therefore the accuracy of the knowledge graph.

6.1.3 Case Studies Involving Lobbyists

The case studies presented in this thesis revolved around legislators or the general public. The knowledge graph also contains data on the utterances spoken by lobbyists and also the organizations represented by those lobbyists. A case study analyzing the opinions of companies towards certain bills and how strongly those opinions are voiced through their lobbyists would be very interesting.

6.1.4 Client for Digital Democracy

The case studies and prototype client presented in this thesis assume that the user is an internal member of the Digital Democracy project with access to the existing Digital Democracy database. This setting does not apply to users of the Digital Democracy website. An external client for website users to query the knowledge graph would be a great benefit.

BIBLIOGRAPHY

[1] Oxford english dictionary definition: Noun phrase. `https://www.oxforddictionaries.com/us/definition/american_english/noun-phrase`.

[2] Aileen Agricola. Neo4j named most popular graph database forrester research. `http://neo4j.com/news/neo4j-named-most-popular-graph-database-forrester-research/`, 2015.

[3] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[4] AlchemyAPI. Alchemyapi: About us. `http://www.alchemyapi.com/about-us`.

[5] AlchemyAPI. Alchemyapi: Alchemylanguage features. `http://www.alchemyapi.com/products/alchemylanguage`.

[6] AlchemyAPI. Alchemyapi: Entity extraction api. `http://www.alchemyapi.com/api/entity-extraction`.

[7] AlchemyAPI. Alchemyapi: Entity extraction api. `http://www.alchemyapi.com/products/alchemylanguage/entity-extraction`.

[8] Bernd Amann and Michel Scholl. Gram: a graph data model and query language. In *BDA*, 1992.

[9] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853, 2005.

[10] Renzo Angles. A comparison of current graph database models. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 171–177. IEEE, 2012.

[11] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40, 2008.

[12] ARDA and Pennsylvania State University Deparment of Sociology. Association of religion data archives. `http://www.thearda.com/`.

[13] Governor Edward G. Brown. California 2015-2016 State Budget, 2015.

[14] Jason Brownlee. A Tour of Machine Learning Algorithms. `http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/`, 2013.

[15] First Amendment Center. CaliforniaâĂŹs sunshine law celebrates 50 years. `http://www.firstamendmentcenter.org/californias-sunshine-law-celebrates-50-years`, 2003.

[16] Fabio Gagliardi Cozman, Ira Cohen, Marcelo Cesar Cirelo, et al. Semi-supervised learning of mixture models.

[17] Sergei N Dorogovtsev and José FF Mendes. *Evolution of networks: From biological nets to the Internet and WWW*. OUP Oxford, 2013.

[18] Donald Feinberg, Merv Adrian, Nick Heudecker, Adam Ronthal, and Terilyn Palanca. Magic quadrant for operational database management systems. `https://www.gartner.com/doc/reprints?id=1-2PO8Z2O&ct=151013&st=sb`, 2015.

[19] Center for Continuing Study of the California Economy. California Remains the World's 8th Largest Economy, 2015.

[20] California State Government. California legislative information. `https://leginfo.legislature.ca.gov/`.

[21] Mark Graves, Ellen R Bergeman, and Charles B Lawrence. A graph-theoretic data model for genome mapping databases. In *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, volume 5, pages 32–41. IEEE, 1995.

[22] The Stanford Natural Language Processing Group. The stanford parser: A statiscal parser. `http://nlp.stanford.edu/software/lex-parser.shtml`.

[23] Ralf Hartmut Guting. Graphdb: Modeling and querying graphs in databases. In *VLDB*, 1994.

[24] Marc Gyssens, Jan Paredaens, and Dirk Van Gucht. A graph-oriented object database model. In *PODS*, 1990.

[25] Michael Hunger and William Lyon. Analyzing the panama papers with neo4j. `http://neo4j.com/blog/analyzing-panama-papers-neo4j/`, 2016.

[26] IATPP Digital Democracy Team. Digital Democracy. `http://digitaldemocracy.org/`.

[27] P James. Knowledge graphs. 1991.

[28] Foaad Khosmood, Alexander Dekhtyar, Hisham Assal, Franz Kurfess, and Joanna Snyder. Making california legislative process transparent. Technical report, California Polytechnic State University, San Luis Obispo, 2014.

[29] Rüdiger Klar and Otto Opitz. *Classification and Knowledge Organization: Proceedings of the 20th Annual Conference of the Gesellschaft für Klassifikation eV, University of Freiburg, March 6–8, 1996.* Springer Science & Business Media, 2013.

[30] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *NIPS*, 2002.

[31] Hans-Joachim Klein and Jochen Rasch. Functional dependencies for object databases. In *Grundlagen von Datenbanken*, pages 56–60. Citeseer, 1997.

[32] Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[33] Gabriel M Kuper and Moshe Y Vardi. The logical data model. *ACM Transactions on Database Systems (TODS)*, 18(3):379–413, 1993.

[34] Mark Levene, Alexandra Poulovassilis, Kerima Benkerimi, Sara Schwartz, and Eran Tuv. Implementation of a graph-based data model for complex objects. *SIGMOD Record*, 22:26–31, 1993.

[35] Bing Liu. Professor bing liu homepage. `https://www.cs.uic.edu/~liub/`.

[36] Steffen Lyngbaek. Spork: A summarization pipeline for online repositories of knowledge. 2013.

[37] MapLight. Maplight: Revealing money's influence on politics. `http://maplight.org/`.

[38] Elaine Marsh and Dennis Perzanowski. MUC-7 Evaluation of IE Technology: Overview of Results. `http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/marsh_slides.pdf`, 1998.

[39] Guy W Mineau and Robert Godin. Automatic structuring of knowledge bases by conceptual clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 7(5):824–829, 1995.

[40] Robert Munro and Christopher D Manning. Accurate unsupervised joint named-entity extraction from unaligned parallel text. In *Proceedings of the 4th Named Entity Workshop*, pages 21–29. Association for Computational Linguistics, 2012.

[41] David Nadeau. *Semi-supervised named entity recognition*. PhD thesis, Citeseer, 2007.

[42] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[43] U.S. Deparment of Commerce. California Census QuickFacts. `http://quickfacts.census.gov/qfd/states/06000.html`, 2014.

[44] University of Pennsylvania Computer and Information Science Department. The penn treebank project. `https://www.cis.upenn.edu/~treebank/`.

[45] Office of the California Secretary of State. Cal-access: Lobbying. `http://cal-access.sos.ca.gov/lobbying/`.

[46] GitHub UserId: oxymor0n. Nltk github issue: Multiprocessing and nltk don't play nicely together. `https://github.com/nltk/nltk/issues/947`.

[47] Jan Paredaens, Peter Peelman, and Letizia Tanca. G-log: A graph-based query language. *IEEE Trans. Knowl. Data Eng.*, 7:436–453, 1995.

[48] Roel Popping. *Computer-assisted text analysis*. Sage, 2000.

[49] Roel Popping. Knowledge graphs and network text analysis. *Social Science Information*, 42(1):91–106, 2003.

[50] Elaine Rich and Kevin Knight. Artificial intelligence. *McGraw-Hill, New*, 1991.

[51] Giuseppe Rizzo and Raphaël Troncy. Nerd: evaluating named entity recognition tools in the web of data. 2011.

[52] I. Robinson, J. Webber, J. Webber, and E. Eifrem. *Graph Databases*. O'Reilly Media, 2013.

[53] Kepa Joseba Rodriquez, Mike Bryant, Tobias Blanke, and Magdalena Luszczynska. Comparison of named entity recognition tools for raw ocr text. In *KONVENS*, pages 410–414, 2012.

[54] Laurel Rosenhall. California lawmakers want to regulate home-sharing businesses like airbnb. `http://www.sacbee.com/news/politics-government/capitol-alert/article15202547.html`, 2015.

[55] Amit Sheth, Boanerges Aleman-Meza, I Budak Arpinar, Clemens Bertram, et al. Semantic association identification and knowledge discovery for national security applications. *Journal of Database Management*, 16(1):33, 2005.

[56] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *ACL*, 2013.

[57] John F Sowa. Semantic networks. *Encyclopedia of Cognitive Science*.

[58] spaCy Development Team. spacy. `https://spacy.io/`.

[59] spaCy Development Team. spacy: Github page. `https://github.com/spacy-io/spaCy`.

[60] Frans N Stokman and Pieter H de Vries. *Structuring knowledge in a graph*. Springer, 1988.

[61] Jun Suzuki and Hideki Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. 2008.

[62] Digital Democracy Leadership Team. Digital Democracy. `http://www.iatpp.calpoly.edu/projects/digitaldemocracy.asp`.

[63] Neo Technologies. Neo4j: The world's leading graph database. `http://neo4j.com/`.

[64] Neo Technologies. Neo4j developer manual 3.0, 2016.

[65] Maksim Tkachenko and Andrey Simanovsky. Named entity recognition: Exploring features. In *KONVENS*, pages 118–127, 2012.

[66] Victoria Uren, Philipp Cimiano, Jose Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: science, services and agents on the World Wide Web*, 4(1):14–28, 2006.

[67] Andrew Wang. Tongs: Tldr; online narrative generatiing system. Master's thesis, California Polytechnic State University, San Luis Obispo, 2016.

[68] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *ACL*, 2013.

[69] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.