WHISK: WEB HOSTED INFORMATION INTO SUMMARIZED KNOWLEDGE

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

Jiewen Wu

July 2016

COMMITTEE MEMBERSHIP

TITLE:	WHISK: Web Hosted Information into Summarized Knowledge
AUTHOR:	Jiewen Wu
DATE SUBMITTED:	July 2016
COMMITTEE CHAIR:	Alexander Dekhtyar, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Foaad Khosmood, Ph.D. Assistant Professor of Computer Science
COMMITTEE MEMBER:	Chris Lupo, Ph.D. Associate Professor of Computer Science

ABSTRACT

WHISK: Web Hosted Information into Summarized Knowledge

Jiewen Wu

Today's online content increases at an alarmingly rate which exceeds users' ability to consume such content. Modern search techniques allow users to enter keyword queries to find content they wish to see. However, such techniques break down when users freely browse the internet without knowing exactly what they want. Users may have to invest an unnecessarily long time reading content to see if they are interested in it. Automatic text summarization helps relieve this problem by creating synopses that significantly reduce the text while preserving the key points. Steffen Lyngbaek created the SPORK [32] summarization pipeline to solve the content overload in Reddit comment threads. Lyngbaek adapted the Opinosis graph model for extractive summarization and combined it with agglomerative hierarchical clustering and the Smith-Waterman algorithm to perform multi-document summarization on Reddit comments.

This thesis presents WHISK as a pipeline for general multi-document text summarization based on SPORK. A generic data model in WHISK allows creating new drivers for different platforms to work with the pipeline. In addition to the existing Opinosis graph model adapted in SPORK, WHISK introduces two simplified graph models for the pipeline. The simplified models removes unnecessary restrictions inherited from Opinosis graph's abstractive summarization origins. Performance measurements and a study with Digital Democracy compare the two new graph models against the Opinosis graph model. Additionally, the study evaluates WHISK's ability to generate pull quotes from political discussions as summaries.

ACKNOWLEDGMENTS

Thanks to:

- My advisor, Alex Dekhtyar, for enlightening me on the value of data science
- My loved ones, for their warm support throughout the years
- My friend, Andrew Wang, for being a T.A./instructor forever and ever
- Kavita Ganesan, for publishing online the Opinosis dataset and useful scripts for working with ROUGE
- Andrew Guenther, for uploading this template to the Cal Poly Github [1]
- Corey Ford, for updating this template to the latest specifications
- Digital Democracy project members, for collaborating on a study with me

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1 Introduction 1

2 Background & Related Work 5

2.1 Automatic Text Summarization 5

2.2 Query-based Summarization 7

2.3 Personalized Summarization 8

2.4 Supervised Summarization 10

2.5 Graph-based Summarization 11

2.6 Multi-document Summarization 12

2.7 Opinosis 13

2.8 SPORK 17

2.9 Digital Democracy 21

3 Design 25

3.1 Generic Data Model 25

3.2 Data Collection 27

3.3 Preprocessing 28

3.4 Summarization 28

3.5 Service 30

3.6 Sentence-Id Graph 31

3.7 Numeric Graph 33

4 Implementation 36

4.1 Data Collection 36

4.2 Preprocessing 38

4.3 Summarization 43

4.4 Service 53

4.5 Example Drivers 53

5 Validation 55
5.1 Graph Performance Comparison 55
5.1.1 Summary Similarity 62
5.2 ROUGE 64
5.3 Digital Democracy 67
5.3.1 Pull Quote Selection 67
5.3.2 Evaluation Method 74
5.3.3 Participants 78
5.3.4 Results 78
6 Conclusion 83
6.1 Future Work 85
BIBLIOGRAPHY 87
APPENDICES
A Graph Performance Data 97
B Survey Example and Results 110

LIST OF TABLES

Table		Page
5.1	Dataset of bills for graph performance comparison	56
5.2	Operation settings of WHISK	57
5.3	Cluster statistics	58
5.4	Cluster graph statistics	58
5.5	Aggregate performance measures of new graphs	62
5.6	Aggregate jaccard similarities	64
5.7	ROUGE-1 and ROUGE-2 scores	66
5.8	Bills for pull quote generation	68
5.9	Bill discussions for pull quote generation	69
5.10	Cluster statistics for bill discussions	70
5.11	Cluster graph statistics for bill discussions for all clusters	71
5.12	Cluster graph statistics for bill discussions for usable clusters	72
5.13	Quote generation methods	73
5.14	Evaluation methods	77
5.15	Number of responses for the survey for each bill discussion	79
5.16	Number of best quotes	80
5.17	Aggregate evaluation results	82
A.1	Comparison of timings for graph creation and traversal	99
A.2	Jaccard similarity between Opinosis and sentence-id	104
A.3	Jaccard similarity between Opinosis and numeric	109
B.1	SB128 Hearing Id#123 pull quotes	131
B.2	SB128 Hearing Id#123 method sources and rating counts	132
B.3	SB277 Hearing Id#308 pull quotes	137
B.4	SB277 Hearing Id#308 method sources and rating counts	139
B.5	SCA14 Hearing Id#1266 pull quotes	142
B.6	SCA14 Hearing Id#1266 method sources and rating counts	143

LIST OF FIGURES

Figure		Page
2.1	Opinosis Graph	15
2.2	SPORK Summarization Pipeline	18
2.3	Digital Democracy Screenshot	23
3.1	WHISK Pipeline	26
3.2	WHISK Data Model	27
3.3	Sentence-Id Graph	32
3.4	Numeric Graph	34
4.1	WHISK Data Model Implementation	37
5.1	Graph creation timings	59
5.2	Graph traversal timings	60
5.3	Jaccard similarities of extracted sentences	63
5.4	Quote selection method comparison for SB277 Hearing Id#261	74
5.5	Graph model comparison for SB277 Hearing Id#261	75
5.6	Graph model comparison for best quotes	81
B.1	Screenshot of the survey for AB1405 Hearing Id#302	111
B.2	Quote selection method comparison for AB1135 Hearing Id#1123	112
B.3	Graph model comparison for AB1135 Hearing Id#1123	112
B.4	Quote selection method comparison for AB1405 Hearing Id#302	113
B.5	Graph model comparison for AB1405 Hearing Id#302	113
B.6	Quote selection method comparison for AB66 Hearing Id#192	114
B.7	Graph model comparison for AB66 Hearing Id#192	114
B.8	Quote selection method comparison for AB884 Hearing Id#1266	115
B.9	Graph model comparison for AB884 Hearing Id#1266	115
B.10	Quote selection method comparison for AB884 Hearing Id#1284	116
B.11	Graph model comparison for AB884 Hearing Id#1284	116
B.12	Quote selection method comparison for SB10 Hearing Id#1048	117

B.13 Graph model comparison for SB10 Hearing Id#1048 117
B.14 Quote selection method comparison for SB1235 Hearing Id#1074 . 118
B.15 Graph model comparison for SB1235 Hearing Id#1074 118
B.16 Quote selection method comparison for SB1235 Hearing Id#1173 . 119
B.17 Graph model comparison for SB1235 Hearing Id#1173 119
B.18 Quote selection method comparison for SB128 Hearing Id#123 . . . 120
B.19 Graph model comparison for SB128 Hearing Id#123 120
B.20 Quote selection method comparison for SB128 Hearing Id#139 . . . 121
B.21 Graph model comparison for SB128 Hearing Id#139 121
B.22 Quote selection method comparison for SB277 Hearing Id#308 . . . 122
B.23 Graph model comparison for SB277 Hearing Id#308 122
B.24 Quote selection method comparison for SB329 Hearing Id#161 . . . 123
B.25 Graph model comparison for SB329 Hearing Id#161 123
B.26 Quote selection method comparison for SB350 Hearing Id#113 . . . 124
B.27 Graph model comparison for SB350 Hearing Id#113 124
B.28 Quote selection method comparison for SB350 Hearing Id#413 . . . 125
B.29 Graph model comparison for SB350 Hearing Id#413 125
B.30 Quote selection method comparison for SCA14 Hearing Id#1266 . . 126
B.31 Graph model comparison for SCA14 Hearing Id#1266 126
B.32 Quote selection method comparison for SCA14 Hearing Id#1284 . . 127
B.33 Graph model comparison for SCA14 Hearing Id#1284 127

Chapter 1

INTRODUCTION

This thesis presents WHISK, an automatic text summarization pipeline for multi-document summarization based on Steffen Lyngbaek's Summarization Pipeline for Online Repositories of Knowledge (SPORK) [32].

Today's internet provides many outlets for user-generated content. Wikipedia, which allows users to create/edit articles in an encyclopedia form, has reached close to five million English articles to this day [61]. Social media platforms, such as Facebook, Twitter, Tumblr, and Reddit, provide an informal community space for almost any type of content. All of these social media platforms have a grouping mechanism, such as community groups or tags, to categorize and organize their content so users find what they want faster.

Keywords are commonly used to generate tags for new content. Topical phrases and distinctive words in a document may be eligible to be a keyword. Topic and keyword extraction utilize natural language processing techniques to automatically find the overall topic and the keywords within a given body of text. However, content creation speed has grown alarmingly and can overwhelm users even with the help of such tools. Within the same topic, there can still be a high volume of discussion and content. Therefore, tagging alone may not be enough to help users find the content they are interested in. Helping users deal with the increasingly large content flow motivated this thesis.

Information retrieval (IR) systems provide a specialized solution using natural language processing techniques. Every piece of text content, referred to as a *document*, goes through an indexing process. Many indexing processes rely on keywords for

servicing user queries. These IR systems help the users find articles most relevant to their needs by matching documents to the user-specified query text. For example, Google is a prominent search engine which makes use of IR techniques to quickly search for web pages relevant to the user's search query. However, these techniques truly shine only in situations where the user knows exactly what they want.

In situations where users explore text content freely, automatic text summarization serves as a great time-saving tool. Radev, Hovy, and McKeown define a summary as "a text that is produced from one or more text(s), that conveys important information in the original text(s), and that is no longer than half of the original text and usually significantly less," where text can be "speech, multimedia documents, hypertext, etc." [43]. Summaries help users quickly decide whether they want to continue reading or move onto another content by drastically reducing the amount of content to process.

Political discussions is one area where summaries can not only save time but also increase awareness. The Digital Democracy project by the Institute for Advanced Technology and Public Policy (IATPP) provides better accessibility to bill discussions in California state legislature through their website (`https://digitaldemocracy.org/`). The website serves as public resource for information on legislative bills as well as video recordings and transcripts for hearings of those bills. Hearings may become quite lengthy when a debate occurs over the issues of a bill. However, the general public may not have enough time to watch a hearing or read a transcript.

Excerpts from the bill discussions may be used to represent the key ideas that were mentioned. These excerpts can give the reader a general highlight of what is being discussed in the hearing. The highlights allow the reader to quickly decide whether they are interested in learning more about the hearing or not. Additionally, because excerpts are far easier to digest, citizens will be less put off from learning

about the current bills compared to watching a video recording or skimming a transcript. Summaries create a lower barrier of entry for the general public to consume information about the current political issues.

Summaries also save time for users on Reddit, an online messaging board for users to post text or links. Reddit users often create synopses, called "TL;DR" or "Too long; Didn't read", to save each other time on long posts. However, no summaries exist as of yet for the comment threads of posts. Summarization Pipeline for Online Repositories of Knowledge (SPORK) [32], a summarization pipeline built by Steffen Lyngbaek, aimed to solve this problem by summarizing Reddit comments using an adapted Opinosis graph model [19] and the Smith-Waterman algorithm [55].

We improve upon the SPORK pipeline through the following ways. First, because SPORK mainly worked with Reddit comment threads, we expand SPORK for general summarization usage. Second, we introduce two different simplifications of the Opinosis graph model to improve performance as a backend service while retaining similar summarization results. Last, we contribute to the Digital Democracy pipeline by creating a module that uses WHISK to select representative sentences from legislative bill discussions.

We perform three validation experiments to test the effectiveness of the new graph versions compared to the adapted Opinosis graph:

- A comparison of graph processing timings and similarities between the resulting summaries.
- Automatic summary evaluations with ROUGE [27] using gold-standard summaries from the Opinosis dataset [19].
- A study in collaboration with the Digital Democracy project on WHISK's ability to generate summaries for political discussions.

The contributions of this thesis are as follows:

- Generalization of SPORK to support general multi-document summarization (WHISK)
- Two simplified variants of the Opinosis graph model for WHISK
- A bill discussion summarization module for Digital Democracy using WHISK

Chapter 2

BACKGROUND & RELATED WORK

2.1 Automatic Text Summarization

Under the umbrella of natural language processing, active research continues to improve automatic text summarization. The summarization techniques can be divided into two approaches: abstractive and extractive.

The abstractive summarization approach aims to create novel sentences that capture the semantics from the corpus text. Humans create summaries in this manner, so the ideal abstractive summaries would be indistinguishable from summaries done by humans. Because capturing semantics is difficult, the existing work have been mostly shallow and does not reflect true abstractions. One of the popular approaches makes use of templates to create novel sentences. McKeown and Radev's SUMMONS (SUMMarizing Online NewS articles) fills in manually created templates with algorithmically selected words to create summaries [34]. However, Das and Martins find that SUMMONS is ineffective in large topic domains because it would require a large amount of templates [10]. Other works [17][19][28] create novel sentences by performing sentence compression techniques. Majority of the work in automatic text summarization utilizes extractive techniques rather than abstractive due to its difficulty.

The extractive summarization approach, on the other hand, involves creating a summary from parts extracted from the corpus text such as phrases or sentences. This approach relies on detecting representative phrases or sentences that are relevant and salient. The phrases or sentences are then arranged into a readable form. A good variety of extractive methods have been explored towards automatic text sum-

marization. Existing methods vary on information usage from the pure document text to the user's metadata.

One popular model used to measure the important of a word or to transform text into a vector space model is the Term Frequency–Inverse Document Frequency (TF-IDF) model [49][56]. Given a collection of texts, TF-IDF weighs terms (or words) based on the term's frequency in a document and inversely on the document frequency. The vocabulary is the set of all terms present in the collection of texts. The term frequency (tf_{d_w}) in a document is simply the number of times the term (w) appears in the document (d). The document frequency (df_w) equals to the number of documents in the collection where the term (w) appears. If there are N documents in the collection, the inverse document frequency (idf_w) equals to $\log \frac{N}{df_w}$. The TF-IDF weight of a term w in a document d can then be calculated as follows:

$$tfidf_w = tf_{d\_w} * idf_w \tag{2.1}$$

There are several variants of the TF-IDF model, but they all work very similarly.

Using the weighing calculation, a document may then be represented in space as a vector of TF-IDF weights. The vector contains TF-IDF weights for all terms in the vocabulary. Terms that are not present in a document d have a term frequency of zero in the document and simply has a TF-IDF weight of zero in document d. This vector space abstraction of documents allows application of more general algorithms than pure natural language processing techniques.

This following five sections cover different types of summarization techniques researched today: query-based summarization, personalized summarization, supervised summarization, graph-based summarization, and multi-document summarization. The two sections afterwards provide an extensive overview on the design of the Opinosis graph model and SPORK. The last section covers the Digital Democracy project and the input data for the study.

2.2 Query-based Summarization

Unlike generic summarization which provides a summary over all ideas in a corpus, query-based summarization specializes summaries towards specific queries or topics. Queries or topics may be generated automatically, entered dynamically by a user, or statically defined through a template. Applications of query-based summarization fall under general search with specific domains such as question-and-answer and scoped product reviews. Existing works [39][66] perform semantic comparisons between queries and sentences using statistical or clustering techniques. However, automatic query generation is one of the most important features towards performing unsupervised summarization.

Lu et al. applies query-based summarization for eBay transaction feedback using automatically discovered categories as queries [31]. Shipping, communication, and service are the three predefined areas in the eBay feedback form, but the user feedback text may contain important aspects of the transaction. Aspects are discovered via either clustering or probabilistic models. The feedback text are then grouped by the most related aspect. For each aspect, representative phrases can be extracted based on frequency. Lu et al. suggests that their approach can be generalized to any rated aspects [31]. One such possibility may be product features in product reviews. Each product may have different types of features than other products, so product features must be defined or discovered. Each product feature as a query would return a summary relating to that specific feature. Instead of a single summary about the product, specialized summaries can provide more details on the specific features. Combining query discovery with query-based summarization, automatic text summarization may be performed unsupervised for arbitrary text.

2.3 Personalized Summarization

Personalized summarization approaches summarization on a user-by-user basis. This approach aims to provide the summaries predicted to be most interesting or useful to the target user. Query-based summarization can be used as a base for personalized summarization. The following works makes use of information about or from the user in order to perform summarizations.

A simple approach identical to query-based summarization is to use keywords. Diaz et al. present a keyword approach by using a user-given vector of keyword weights to score sentences [13]. The sentences containing more keywords will be considered more relevant to the user. However, their approach requires users to manually input the vector of keyword weights.

Annotations, or marked words or phrases, is another similar method to keywords for personalizing summarization. Zhang et al. combine users' annotations in text along with the TF-IDF model [49][56] to find the most representative sentences [67]. The annotated keywords and the sentences that contain them are given weights according to frequency and predefined annotation weights. The annotated weights with TF-IDF weights of words together form word scores. Sentences with the highest sum of word scores are then composed into a summary. Móro et al. goes further by combining user annotations with domain-specific words to get better personalization in the area of learning [37]. Like the previous approach, annotation approaches also requires manual user input.

More generically, user models can be used to track characteristics or interests of users. When selecting candidate sentences for a summary, the candidates are scored by their similarity to the user model. Díaz and Gervás use a user model consisting of a long term model that represents stable information needs and a short term model

that represents temporary information needs [12]. The long term model would be built with general categories, such as sports or national newspaper sections, and user-inputted keywords. The short term model, on the other hand, contains representative keywords from the users' feedback on the documents they have read. Both models are represented as term weight vectors that can be compared with candidate sentences. However, because this work uses user feedback, manual user actions are still required specifically for the summarization.

Campana and Tombros introduce a method for automatically discovering user models based on the user's interactions [4]. For every page the user reads, sentences are scored based on term frequency, position in the page, and shared words with the title. The top scoring sentences up to a maximum number are saved into the user model represented as a complete graph with sentences as nodes and similarity between sentences as edges. Candidate sentences are scored by their maximum score when compared to the model nodes by sentence similarity and the node's degree. With the help of automatic user model discovery, systems can learn the user's interests and provide better personalized summaries the more the user uses the system.

In the Twitter application domain, Ren et al. select tweets as personalized time-aware summarizations using a user's history and social network. The authors proposed a Tweet Propagation Model (TPM) which predicts a user's interests using probability distributions. TPM considers a user's personal posts as well as the interests of the user's friends. Ren et al. implement time awareness by sampling for each fixed time interval and finding the tweets most aligned with the TPM for that interval [45]. By tracking a user's interests, TPM would be able to show the most relevant tweets relative to a user.

2.4 Supervised Summarization

Several studies have applied supervised machine learning towards summarizing text. Supervised learning approaches require training on labelled data which express the "ground truth" of data points. The labelled data is often specialized to a specific domain or application. Additionally, labelled data is usually manually obtained through human experts which can be expensive in both time and money. To alleviate strong reliance on labelled data, one may consider semi-supervised approaches such as Wong et al.'s solution which combines labelled data with unlabelled data to achieve comparable results as fully supervised approaches [62]. The following works apply supervised approaches such as classification, which builds classifiers for categorizing items, or regression, which builds a mathematical function for predicting numeric values, towards summarizing tweets.

Rudra et al. takes advantage of language patterns during disaster events to extract situational information, such as number of causalities or the current situation in a region [47]. The authors trained a classifier called Support Vector Machine (SVM) [60] with 1000 random labeled (or categorized) samples to distinguish tweets with situational information versus those that contain non-situational information, e.g. a person's sentiment for the victims. The trained classifier, evaluated at 80% accuracy, filters out non-situational tweets and allows extraction of situational information. Rudra et al. finds the tweets with the most important content through a scoring called TF-IDF [49][56]. Part of Speech (POS) taggers help identify numerals, nouns, and main verbs for reporting constantly changing information [47].

User contextual information leads Chang et al. towards performing regression on features derived from time, popularity, and text to summarize Twitter conversations [7]. Popularity features come from following the PageRank algorithm focused on *reply* and *retweet* relationships. After converting text to a vector-space model using TF-

IDF [49][56], distance measures from the center point of the conversation dictate a tweet's textual features. Chang et al. applies a regression algorithm called *Gradient Boosted Decision Tree* to decide which few tweets summarize the conversation [7].

While both Rudra et al. and Chang et al. apply machine learning techniques to enhance summarization, both works target special domains. Because of this specialization, their work is harder to applied to other areas of summarization, especially if the goal is to summarize arbitrary text which do not follow any assumptions about structure or topic. Since we are interested in generic summarization of text, the approaches we focus on will largely be unsupervised.

2.5 Graph-based Summarization

Many summarization techniques use a graph model at some point. Even in the works above, graphs make a difference in popularity measures. The following works center their algorithms around graph models.

Sharifi et al. proposes the *Phrase Reinforcement* algorithm which builds a graph of common word sequences for summarizing user query results. The root node contains the topic phrase and its number of occurrences. All other nodes contain a word and are weighted proportional to their count lessened by their distance from the root node. The joined path with the maximum total weight on both the left and right side of the topic phrase identifies the best summary sentence [52].

Instead of complete sentences, Kim et al. proposes a method to find the most important keywords for user queries. After filtering by the query, words within a 1-25% frequency range are considered as keywords. Kim et al. constructs a graph based on co-occurrences of those keywords. Maximal k-cliques are then identified in the graph and used to find clusters, or groups, of tweets. Such clusters are then merged if there is enough tweets shared by two clusters. The words in those cliques

then become the most important keywords for that cluster [25].

Sumblr, created by Shou et al., clusters similar tweets together using k-means clustering and selects representative tweets from each cluster with LexRank to generate a summary. Sumblr takes into account the tweet's timestamp and the posting user's rank in the Twitter social network based on user relationships during clustering. New clusters are made when the nearest cluster for a new tweet is too high above a given parameter [53].

Khan et al. performs graph-based querying after topical clustering [24]. Tweets are first clustered by topic with a topic model called *Latent Dirichlet Allocation (LDA)* [3]. Khan et al. constructs a co-occurrence graph and find the most popular terms and co-occurrences using the weighted, undirected version of the PageRank algorithm. Tweets containing the most popular terms are elected as representative tweets for the summary [24].

2.6 Multi-document Summarization

Within automatic text summarization, multi-document summarization focuses on summarizing core ideas spanning across multiple corpus texts instead of a single coherent document. Examples of multi-document corpora are Reddit comment threads, Twitter tweets, and forum threads. Each comment, tweet, or post can be considered as a single document. However, each document can contain its own distinctive ideas. Modifications to existing single document techniques will be necessary to avoid redundant summary sentences while still capturing the unique ideas about various topics. The ability to group documents of similar ideas or topics together play a significant role in generating quality summaries.

One approach by Celikyilmaz and Hakkani-Tür identifies key concepts and relationships between the concepts in a hierarchical fashion. Two types of topics, low-level

and high-level, are identified and associated with one another. High-level topics are more general and abstract while low-level topics are more specific subtopics. Sentences are scored based on related words to the topics. The best scoring sentences form the non-redundant summary [6].

Fung et al. and Hu et al., on the other hand, utilize clustering techniques to create topically coherent document clusters. For each cluster, the sentence closest to the cluster centroid would be selected as the representative sentence. Assuming the clusters are non-redundant relative to each other, the representative sentences together would then form a non-redundant summary [18][21][22].

WHISK offers another novel approach by applying the Opinosis graph [19] towards extractive summarization on multiple documents or texts. The Opinosis graph relies on highly redundant text to find the most important word sequences. With a high number of texts available, there will be a good amount of redundancy for the Opinosis graph to take advantage of. A query-based traversal from SPORK helps alleviate situations where there are still a low number of topically aligned texts.

To better understand the foundation WHISK is built on, the next sections cover the inner workings of the Opinosis graph model and the original SPORK pipeline.

2.7 Opinosis

Ganesan et al. present the Opinosis graph model as an abstractive summarization technique [19]. In abstractive summarization, the system constructs novel sentences in order to summarize the given text. Opinosis targets bodies of text that contain highly redundant sentences, or sentences which consist of shared sequences of words. A word sequence graph naturally captures the redundancy in the text. On microblogs like Twitter, a good portion of posts on the same topic are expected to be fairly redundant as people often share similar opinions in groups.

Before generating the graph, preprocessing includes tokenization and part-of-speech tagging. Tokenization breaks up bodies of text into parts called tokens. In English, sentences may be tokenized by splitting at recognized punctuations such as periods, question marks, semicolons, ellipses, and exclamation marks. At the next level, sentences may be tokenized into words by naively splitting the text at space characters. Different tokenization techniques may be used, but the overall goal of breaking the text down into sentences and words is the same.

Part-of-speech tagging involves recognizing what part of speech a word is within the given sentence. This is important when a word may be used in multiple ways with different meanings. Abstractive summarization requires this in order to make meaningful sentences because the summary should following proper grammatical structures and rules. Many part-of-speech tagging solutions exist [8][9][11][20][35][44][50][51][58], but none are perfect. After running a selected tagger on the tokenized text, the Opinosis graph can be built.

The Opinosis graph contains nodes, representing a word and its part-of-speech, and directed edges, which indicate one word unit following another in some sentence (also known as a bigram). A node contains metadata about its word including its part-of-speech and its occurrences. The word and its part-of-speech are considered the uniquely identifying factors of a node. Each node's occurrence is recorded as a pair: the unique id of the sentence the word occurs in and the position in which the word occurs in the sentence. Figure 2.1 shows how an Opinosis graph instance looks like for an example set of sentences.

Special nodes include start nodes and end nodes. Start nodes signify words that are possible starts of a sentence based on the average position index a word occurs in sentences. The maximum average position to be considered a start node is empirically determined. End nodes are punctuation or coordinating conjunctions which can end

Id	Sentence
1	This tests for great usability with mice.
2	We use tests to measure usability with mice and keyboard.
3	Why should we test usability?

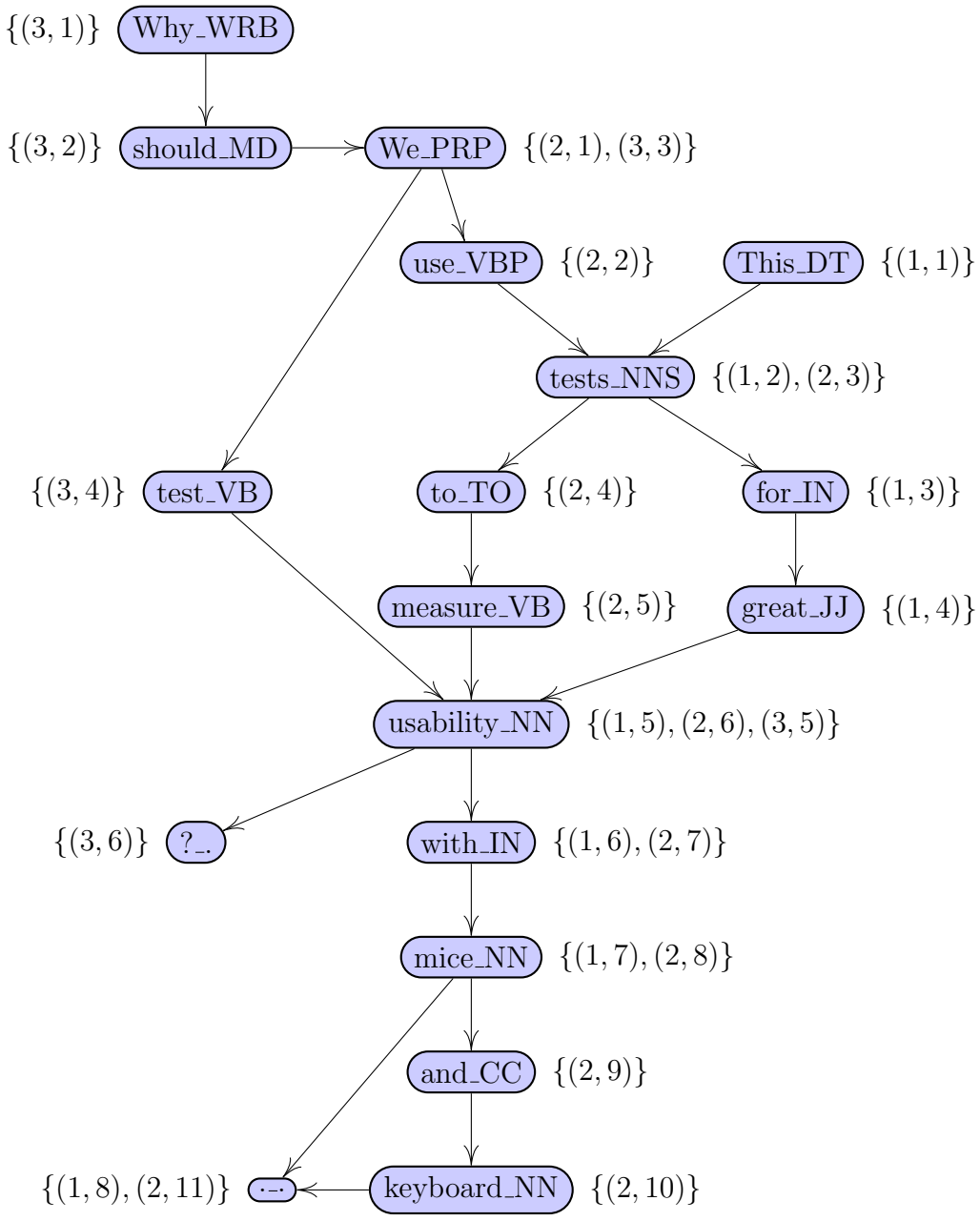

Figure 2.1: ***Opinosis Graph.*** **An example Opinosis graph for a given set of sentences.**

a sentence. These special nodes are captured from the tokenized text while building the graph.

Sentences are derived by scoring possible paths in the graph with several criteria. Eligible paths in the original Opinosis technique must start at a valid start node and end at a valid end node. Paths must also follow grammar rules enforced via part of speech ordering. These two requirements are loosened in the experimentation discussed in later sections. Stitching techniques are not discussed here since they only relate to abstractive summarization and are not used in SPORK.

Consider an eligible path $W = w_1, ..., w_k$ where w_1 is the starting node and w_k is the ending node. The path score can be computed using the following formula:

$$s(W) = \frac{1}{|W|} * \left[r(w_1, w_2) + \sum_{i=3}^{k} \Big(\log_2 |w_1, ..., w_i| * r(w_1, ..., w_i) \Big) \right] \tag{2.2}$$

where $|W|$ is the length of the path, $|w_1, ..., w_i|$ is the length of the subpath between w_1 and w_i, and $r(w_1, ..., w_i)$ is a function used to determine the redundancy score of the subpath. The function $r(a, ..., b)$ is defined as $r(a, ..., b) = |p_a \cap ... \cap p_b|$ where p_i is the sentence occurrences of the node by sentence id. Intersection is restricted by a gap criteria where shared sentence ids are only considered common if the word position difference between the given node and the following node are less than or equal to the maximum gap [19].

The top valid paths are reordered by sentence order and composed to form the abstractive summary. The number of top paths is empirically set. SPORK uses the Opinosis graph model to discover important word sequences in the text.

2.8 SPORK

Steffen Lyngbaek built the SPORK pipeline in Python and applied it towards summarizing Reddit comment threads [32]. SPORK first makes use of a clustering technique to group similar comments together before performing summarization. SPORK combines abstractive and extractive techniques to form a hybrid summarization solution. Unlike abstractive techniques which create novel sentences, extractive techniques take parts of the original text to create the resulting summary. SPORK adapts the Opinosis graph model [19] as an extractive approach to derive key features of the summary. The key features are then used with the Smith-Waterman algorithm [55] to perform sentence extraction from the original text. The SPORK pipeline is split into three separate stages: data collection, preprocessing, and summarization. A visual overview of the SPORK pipeline is shown in Figure 2.2 [32].

Data Collection. SPORK collected reddit posts and comments from subreddit forums to be summarized. The two subreddits Lyngbaek focused on were *r/technology* and *r/politics*. To do this, a Python wrapper called Python Reddit API Wrapper (PRAW) was used to interface with Reddit's API. The data is then persisted in a MySQL database for local consumption. Only new posts or comments are queried using the PRAW interface [32].

Preprocessing. In the preprocessing stage, SPORK performs tokenization, part-of-speech tagging, keyword ranking, and clustering.

Because SPORK needs to work on the word level in the later stages of the pipeline with the Opinosis graph model, tokenization and part-of-speech tagging are required. SPORK utilizes tokenizers available through the Natural Language Toolkit (NLTK) [29] package for Python. The *PunkSentenceTokenizer* is used for sentences while the

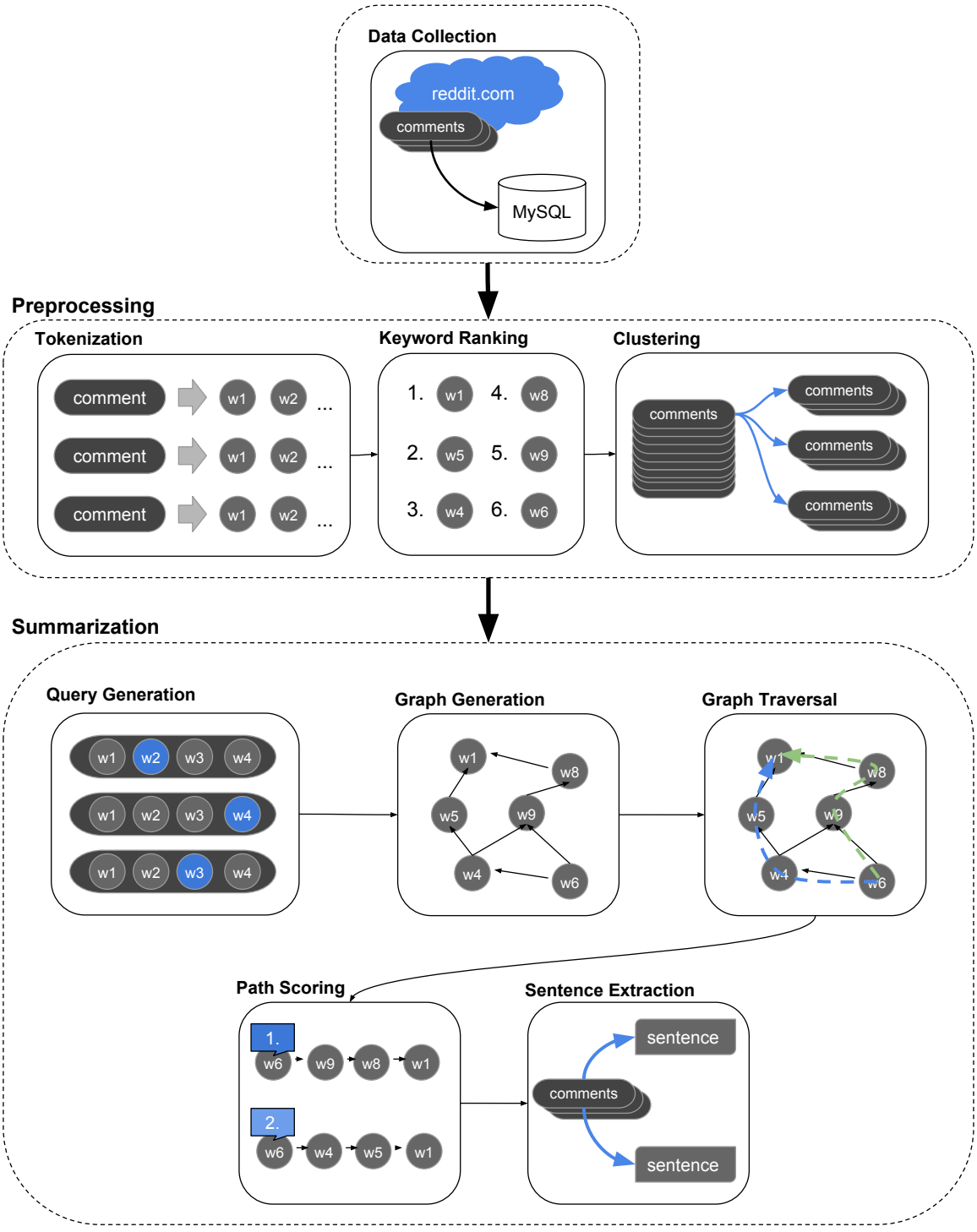

Figure 2.2: *SPORK Summarization Pipeline.* (adapted from [32])

TreebankWordTokenizer is used for words. Part-of-speech tagging is handled by the Stanford POS Tagger [58] using the left3words model: *wsj-0-18-left3words.tagger* [32].

Keyword rankings are generated from a set of threads in a subreddit forum to detect stop words. Besides the common articles such as "the" and "a", subreddits can contain commonly used words since they are specialized in a certain topic. For example, Lyngbaek mentioned that the *r/technology* subreddit may commonly use the words "technology" or "computer," but they do not necessarily help give distinctive meaning since many posts may contain them. SPORK uses the TF-IDF model [49][56] on a set of comments in a subreddit to determine stop words for each subreddit. These keyword rankings are used later in the pipeline to score paths in the Opinosis graph [32].

Comment threads are discussions, so various disjoint topics appear. The different topics must be detected, grouped, and summarized properly. SPORK topically groups comments in a post by using agglomerative hierarchical clustering using the single-link method. Comparisons between comments are made using the cosine similarity metric. SPORK assumes that child comments are topically related to its parent comment, so the clustering only targets top-level comments. The output groups (or clusters) of comments are then fed into the summarization stage where they are processed independently [32].

Summarization. In the summarization stage, SPORK works with the Opinosis graph [19] and the Smith-Waterman algorithm [55]. The steps are as follows: query generation, graph generation, graph traversal, path scoring, and sentence extraction.

Query generation involves discovering representative keywords for a cluster of comments. Such keywords are found by using the χ^2 measure [33] on co-occurrence of the top 30% frequent words in the cluster. The main idea is that frequent words that

co-occur with a small subset of other frequent words are considered *biased* towards the subset. The higher the bias, the more important the word is considered to be. SPORK uses these representative keywords for each cluster to optimize traversing the Opinosis graph [32].

The Opinosis graph is generated following the techniques described in Section 2.7 with a few modifications. Each node contains some metadata information such as the generic id of the comment the word is found in, the id of the comment within the cluster, and the (upvote rating) score received by the comment. Recall that the Opinosis graph relies on heavy redundancy in the text to be effective. Instead of creating a graph for the entire body of text consisting of every comment in a post, SPORK generates a graph for every cluster of comments. This method helps to create more cohesive graphs with less noise. Each Opinosis graph is then traversed for meaningful paths [32].

The query words from query generation guide the traversal of the Opinosis graphs. In addition to the the start/end node restriction described in Section 2.7, paths must also contain at least one of the query words. Instead of finding paths and then filtering based on the queries, SPORK starts at the node representing a query word and finds paths via forward and backwards propagation. Each possible forward path is then combined each possible backward path to form the set of paths containing a given query word. These sets are then combined to form the total set of paths. Instead of following the more restrictive sentence structure requirement, paths are tested for validity using sentence length and a requirement to contain a *verb*. The valid paths of a graph are then scored to find the most useful and meaningful path [32].

SPORK scores paths using an augmented version of the path scoring formula using TF-IDF scores and metadata. The TF-IDF keyword rankings from the preprocessing stage are used to boost paths containing an important keyword. The sum of upvote

scores of comments containing the word also contribute towards raising the score. The customized formula compared to the one described in Section 2.7 is as follows:

$$s(W) = \frac{1}{|W|} * \left[r(w_1, w_2) + \sum_{i=3}^{k} \Big(\log_2 |w_1, ..., w_i| * r(w_1, ..., w_i) * tfidf(w_i) * upvote(w_i) \Big) \right] \tag{2.3}$$

[32]

Using top paths discovered with the graph models, SPORK extracts the most important sentences of each topic cluster to form a summary. The top two paths for each query word in graph traversal are used to discover summary sentences. The paths are then compared to sentences in the original comments using a similarity metric to find the top matching sentence for each path. Lyngbaek experimented with a number of metrics, including Jaccard Similarity, Dice Similarity, Cosine Similarity, Minimum Edit Distance, and Local Alignment, to find that Local Alignment excelled in discovering up to 80% of the important topics. SPORK utilizes the Smith-Waterman algorithm [55] to implement Local Alignment. The top matching sentences from each cluster together form a concise summary spanning the various topics in the comments [32].

WHISK takes the foundational ideas of SPORK to build a more generic summarization pipeline. The next chapter discusses the conceptual design behind WHISK and how it differs from SPORK.

2.9 Digital Democracy

One area automatic text summarization can apply is helping the general public have better access and understanding of their state legislatures. We applied WHISK to the

problem of finding parts of political discussions that serve as good summaries. This work was done in collaboration with the Digital Democracy project by the Institute for Advanced Technology and Public Policy (IATPP). This section provides a brief overview of the project.

The Digital Democracy project provides a website (ontaining resources on the California state legislature's committee hearings. The website serves as a searchable database that allows users to view the video recordings, transcripts, and additional data on committee hearings. The Digital Democracy project retrieves video recordings of committee hearings and information on legislative bills, legislators and lobbyists from official public databases and websites. The transcriptions and additional data such as the speaker's position on the issue come from Digital Democracy's set of internal tools. Users may search for committee hearings by keyword, topic category, or by date. Upon viewing a committee hearing, users can review the video recording along with the synchronized transcript as shown in Figure 2.3. Committee hearings consist of discussions of individuals bills with a committee vote on a bill taking place usually at the end.

The amount of information from the video recording and transcript may be too much for users to handle. The committee hearing pages, as shown in Figure 2.3, lack synopses or overviews that easily convey to the user the main ideas within the discussions. In journalism, phrases, quotations, or excerpts called pull quotes highlight the key points from the article. We apply WHISK by extracting a select number of representative sentences as pull quotes for committee hearings.

The input data from Digital Democracy comes in units called utterances. Each utterance represents a single, uninterrupted thought by a speaker during a hearing. Utterances are represented on the committee hearing pages as the separate containers for each speaker in the transcript shown in Figure 2.3. Each utterance comes with

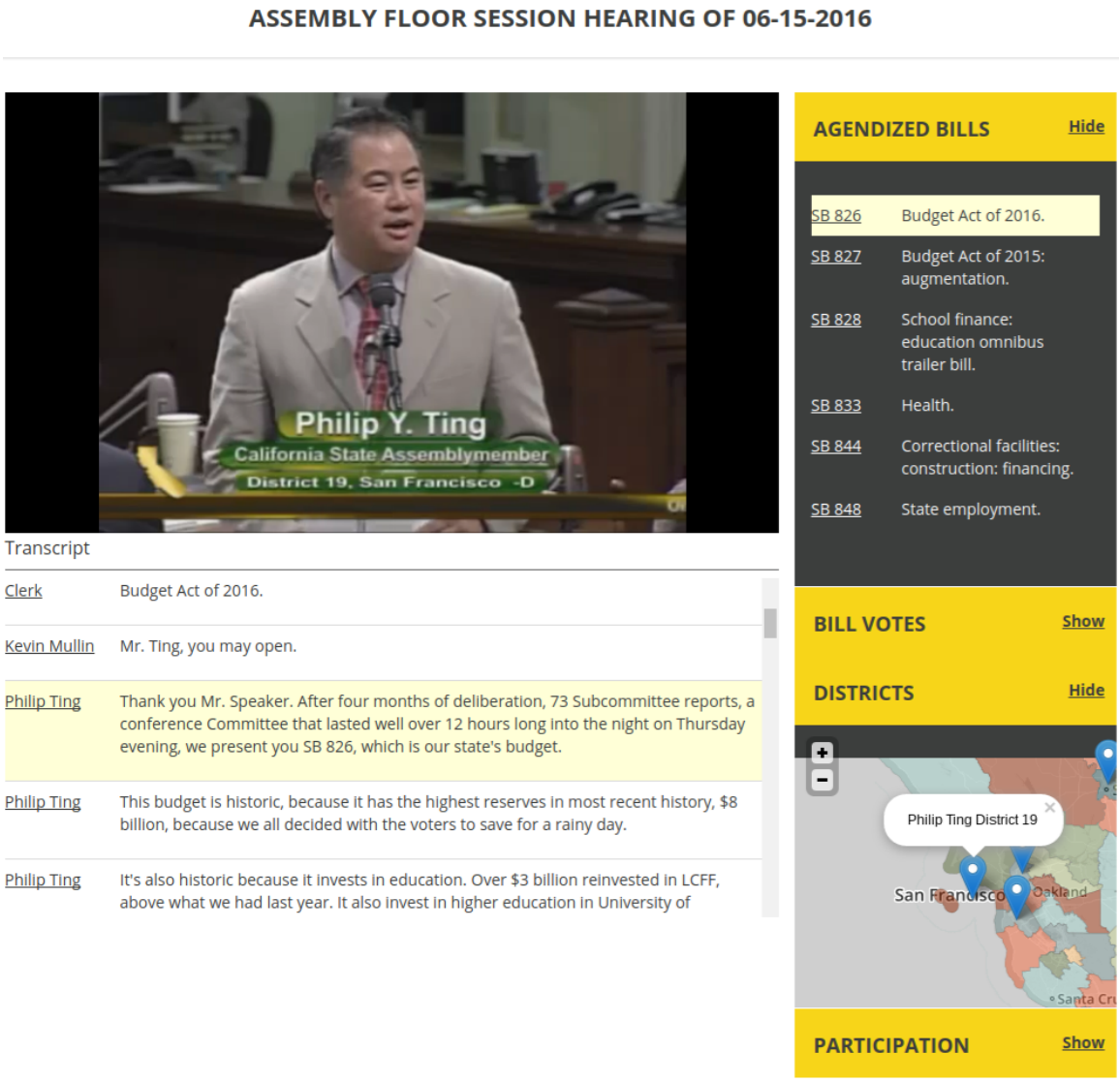

Figure 2.3: ***Digital Democracy Screenshot.*** **A committee hearing page on the Digital Democracy website.**

the following metadata established during the transcription process: the name and identification of the related bill and hearing, the position in the video recording, the speaker's identity (including their name and whether they are a legislator, a lobbyist, or part of the general public) and position on the issue (for or against).

Utterance datasets for WHISK come in the form of comma-separated values (CSV) files. Each dataset contains all utterances from one bill discussion. The speaker's name is used to set the context for selected sentences. We primarily work with the textual transcription using summarization techniques to select representative sentences. Please see the work by Rovin [46] and Wu [63] on Digital Democracy for details on the transcription process that creates the utterances used in our study.

Chapter 3

DESIGN

WHISK extends SPORK by generalizing the pipeline towards summarization for many platforms and improving the performance of pipeline components to better serve as a back-end service. The top level layers defined in WHISK include the original SPORK layers: *data collection*, *preprocessing*, and *summarization*, as well as a new *service* layer. A visual overview of the new WHISK pipeline is shown in Figure 3.1.

3.1 Generic Data Model

To enable support for many platforms, a generic data model is introduced for working with WHISK. The generic data model allows drivers for different services to manage the corpus to be summarized and their resulting summaries. An additional goal of the generic data model is to allow extension modules to be easily added to pipeline. Features such as sentiment analysis may be of interest as a service and can share similar work done for summarization. Metadata may be introduced into the data objects for use in extension modules that can make use of such data.

The overall structure of the data model is displayed in Figure 3.2. A driver program initializes the corpus object with the plain text and set applicable pipeline options in the options field. The rest of the fields are populated by stages in the pipeline with appropriate results. The metadata fields are used to hold domain-specific metadata as well as additional data introduced by extension modules.

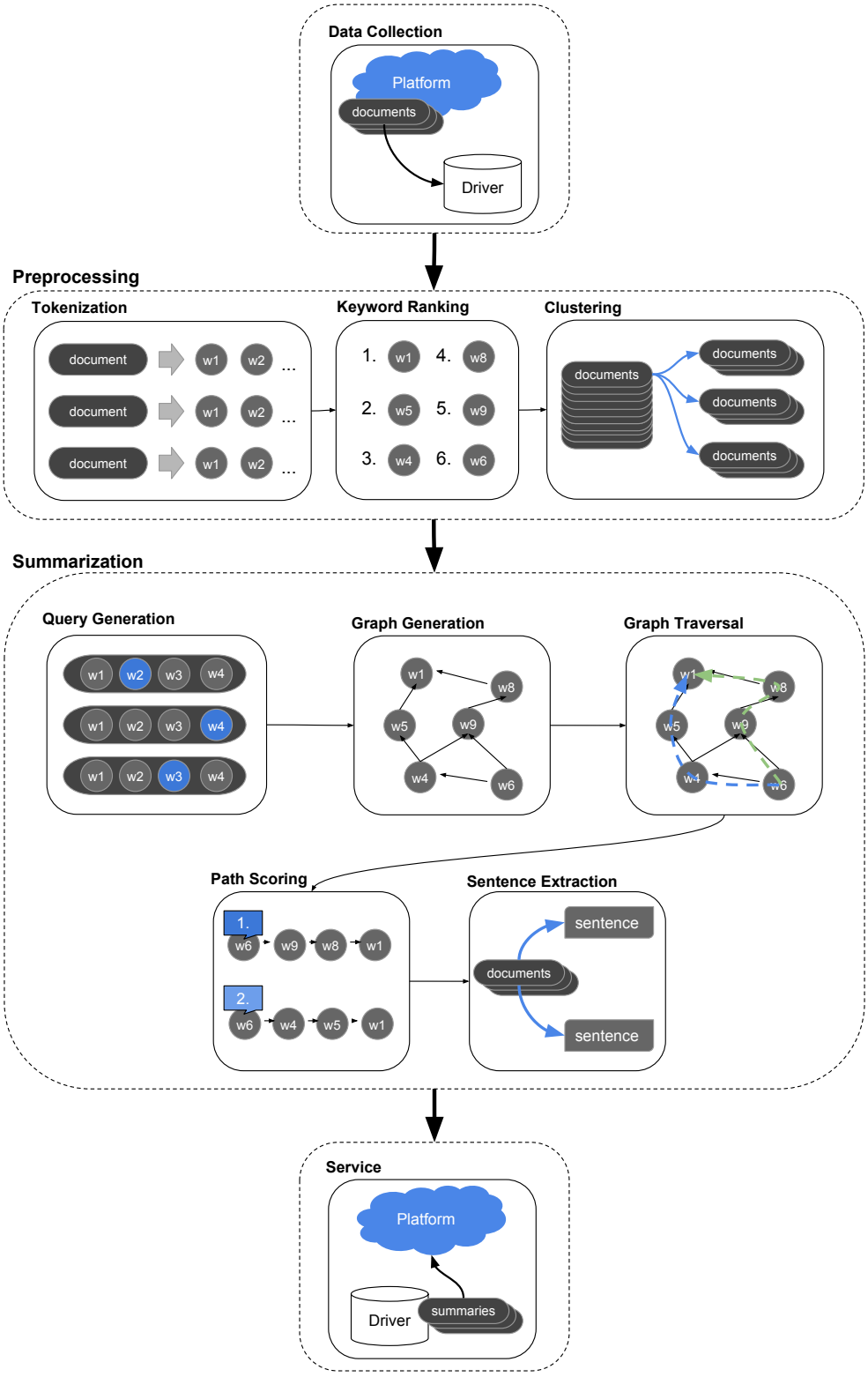

Figure 3.1: ***WHISK Pipeline***

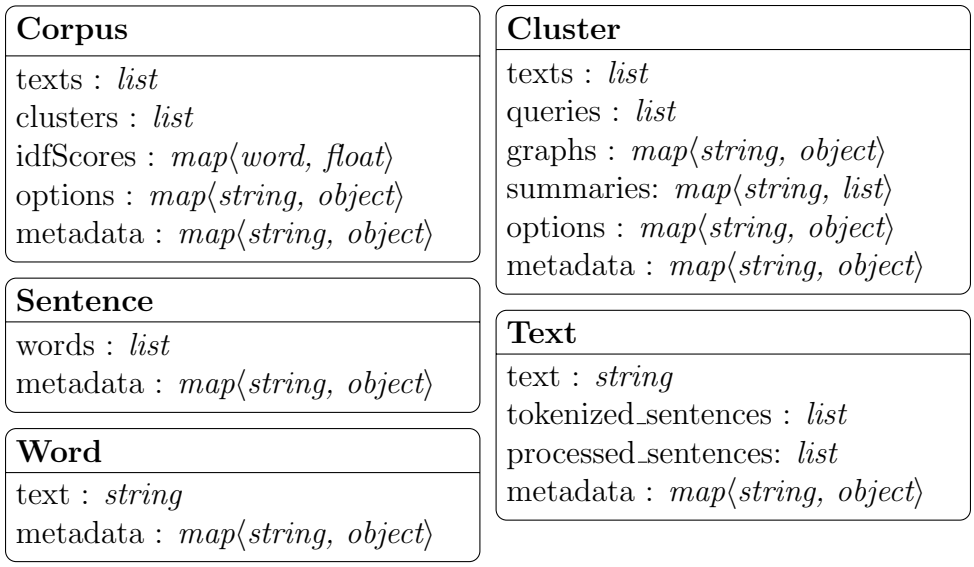

Figure 3.2: *WHISK Data Model.* An overview of the generic data model in WHISK.

3.2 Data Collection

Drivers for a specific platform handle collection of the corpus to be summarized. These drivers are to be created and handled by developers. For example, a Reddit comment driver collects comment threads to be summarized as well as any other threads required for intermediate stages such as keyword ranking. Since the driver performs the collection, the driver must define what it considers to be a unit of text and how multiple units of text compose the corpus. After collection, the driver converts the corpus into the generic data model for working with WHISK. A driver program is responsible for any other features useful for its operation such as caching and persistence of corpus text and summaries since WHISK does not implement these features. Once the corpus data is ready, the driver passes it into the preprocessing stage.

3.3 Preprocessing

WHISK's preprocessing steps follow the same steps in SPORK with added features. The corpus text is tokenized by sentences and then by words. Additionally, simple stopword filtering and word stemming [30] may be optionally performed. The tokenized text is then analyzed using keyword ranking in order to detect special stopwords in the corpus. Finally, the text in the corpus is clustered into groups to be processed by the summarization stage.

3.4 Summarization

WHISK introduces several changes to query generation, graph generation, graph traversal, and path scoring. Sentence extraction as the last step remains the same design as in SPORK using the Smith-Waterman algorithm.

Query Generation. Query generation becomes a more optional step to be used by the developer's discretion. Certain corpora benefit from querying based on keywords, while others may be redundant enough that such work is unnecessary. By moving queries as an option, developers with domain knowledge will have more control over what WHISK produces.

Graph Generation. Three different versions of the Opinosis graph model are included in WHISK for summarization: the original Opinosis graph model described in Section 2.7, a sentence-id graph model, and a numeric graph model. The original Opinosis graph model based on the paper by Ganesan et al. is the basic model used in SPORK [19][32].

The sentence-id graph model is a simplified version of Opinosis which ignores word

positions. Because word positions are ignored, the gap threshold present in Opinosis is not used. The formalization of the sentence-id graph is covered in Section 3.6. We make this simplification because we are not generating novel sentences as done by Ganesan et al. [19] which requires grammatical correctness. Ganesan et al. found that increasing the maximum gap threshold between words leads to better performance in generating summary sentences but also carries the possibly of generating grammatically incorrect sentences [19]. Since we use the sequences instead to match and extract sentences from the original text, grammatical correctness of the sequences does not matter. However, the order in which the words appear in the sentences is still preserved via the directed edges. Therefore, the paths in the sentence-id graph still capture the number of sentences which contain the same word sequences.

The numeric graph model is a further simplification based on the sentence-id graph model. Instead of keeping any sentence information, we keep track of word nodes and how many times they co-occur with an edge and a counter. The graph becomes a classic directed graph where edges represent co-occurrence. The formalization of the numeric graph is covered in Section 3.7. An interesting possibility is to perform LexRank [14] on this co-occurrence graph for keyword extraction. The effectiveness of this graph is to be tested in the experimentation section of this thesis.

Additional metadata can be added onto nodes in the graph for the purpose of utilizing extension modules that can work the graph models. Methods for doing so is discussed later in Chapter 4. After generating the selected graph model, WHISK traverses the graph for paths.

Graph Traversal. WHISK loosens the restrictions on traversing the Opinosis graph compared to SPORK. Instead of following the original requirement of valid start and end nodes, WHISK allows traversal to begin at any node in the graph. Additionally, paths are not required to follow the sentence structure restriction presented by Gane-

san et al. or the simplified sentence restriction presented by Lyngbaek [19][32]. Path traversal also takes into account queries if they are used as SPORK does. WHISK scores discovered paths with a numeric value to determine the top paths.

Path Scoring. Because different graph models are used, the scoring method changes depending on the graph. The redundancy measure on the sentence-id graph model no longer checks for gaps and simplifies to a set intersection. On the even simpler numeric graph model, scores are merely done with numbers without any set operations. The design of the new scoring methods for sentence-id and numeric graphs are described in the Section 3.6 and Section 3.7 respectively.

The path scoring from SPORK for the Opinosis graph is updated in WHISK to reflect TF-IDF scores on paths containing only two nodes. The updated formula based on Equation 2.3 in SPORK is as follows:

$$s(W) = \frac{1}{|W|} * \Bigg[tfidf(w_1) + r(w_1, w_2) * tfidf(w_2) + \sum_{i=3}^{k} \Big(\log_2 |w_1, ..., w_i| * r(w_1, ..., w_i) * tfidf(w_i) \Big) \Bigg] \tag{3.1}$$

We have $tfidf(w_1) + r(w_1, w_2) * tfidf(w_2)$ instead of the original $r(w_1, w_2)$ to balance the TF-IDF scoring bonus. Without this addition, paths containing two nodes have a severe scoring disadvantage compared to longer paths.

3.5 Service

The service layer is an optional layer to be implemented in a driver program which handles summarization as a service to users. This layer ties in with the data collection layer to form the input and output of the WHISK pipeline. The service layer handles

interpreting output from the summarization stage and extension stages and packaging it into a useful result.

For example, if the summarization service is exposed as a REST API, a driver may take the relevant result values and package it into a JSON object. The presentation of the resulting JSON object can then be handled by javascript programs on the user's browser.

The following sections formalize the sentence-id and numeric graph models introduced earlier in the overview of WHISK's graph generation.

3.6 Sentence-Id Graph

The sentence-id graph model inherits the following from the Opinosis graph model: nodes, representing a word and possibly its part of speech, and directed edges indicating one word following another in some sentence. Like Opinosis, a node contains metadata about its word including its part-of-speech and its occurrences. The word and its part-of-speech are considered the unique identifiers of a node if the part-of-speech feature is enabled. Otherwise, the word alone becomes the unique identifier. Each node records a set of unique identifiers of the sentences that the word occurs in. Figure 3.3 shows how a sentence-id graph instance looks like for an example set of sentences.

The special start and end nodes from the Opinosis graph are still maintained in the sentence-id graph in a simplified manner. The first word in a sentence is considered a start node, while the last word in a sentence is considered an end node. Users may choose to restrict traversal to paths starting at start nodes and ending at end nodes. However, this is not recommended since the idea of the sentence-id graph is to not worry about word positions.

Id	Sentence
1	This tests for great usability with mice.
2	We use tests to measure usability with mice and keyboard.
3	Why should we test usability?

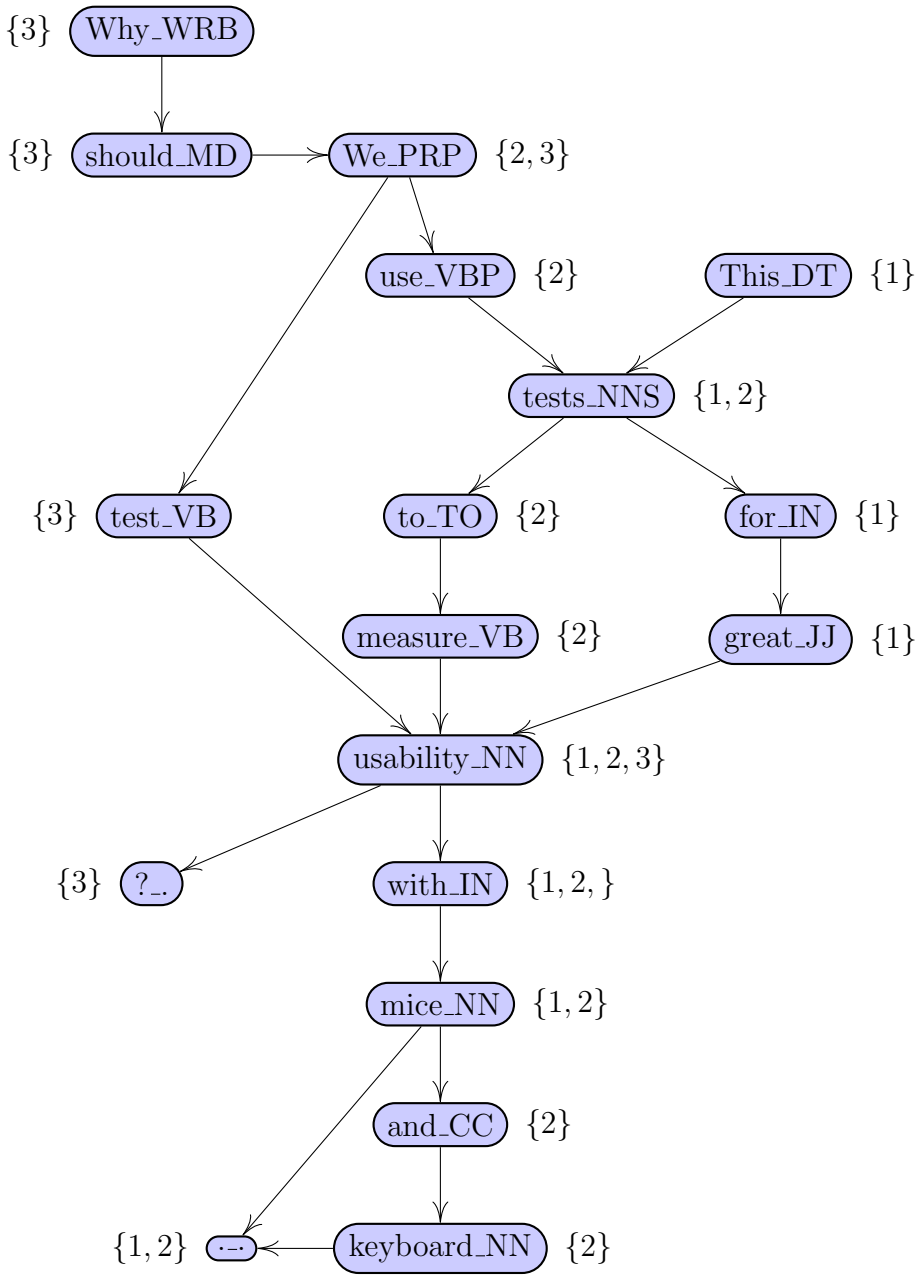

Figure 3.3: ***Sentence-Id Graph.*** **An example sentence-id graph for a given set of sentences.**

The scoring of paths is simplified compared to SPORK by using simple set intersections. Consider an path in the sentence-id graph $W = w_1, ..., w_k$ where w_1 is the starting node and w_k is the ending node. The path score can be computed using the following formula based on Equation 2.3 in SPORK:

$$s(W) = \frac{1}{|W|} * \Bigg[tfidf(w_1) + r(w_1, w_2) * tfidf(w_2) + \sum_{i=3}^{k} \Big(\log_2 |w_1, ..., w_i| * r(w_1, ..., w_i) * tfidf(w_i) \Big) \Bigg] \tag{3.2}$$

where $|W|$ is the length of the path, $|w_1, ..., w_i|$ is the length of the subpath between w_1 and w_i, and $r(w_1, ..., w_i)$ is a function used to determine the redundancy score of the subpath. The function $r(a, ..., b)$ is defined as $r(a, ..., b) = |p_a \cap ... \cap p_b|$ where p_i is the set of sentence occurrences of node i by sentence id. The function $tfidf(i)$ returns the TF-IDF score of the word the node i represents. The TF-IDF scores are identified solely by the word text.

3.7 Numeric Graph

The numeric graph model inherits the following from the Opinosis graph model: nodes, representing a word and possibly its part of speech, and directed edges indicating one word following another in some sentence. A node contains metadata about its word such as its part-of-speech. The word and its part-of-speech are unique identifiers of a node if the part-of-speech feature is enabled. Otherwise, the word alone serves as the unique identifier. Each edge records the number of co-occurrences between two nodes as its edge weight. Figure 3.4 shows how a numeric graph instance looks like for an example set of sentences.

The special start and end nodes are maintained in the same way as the sentence-id graph described in Section 3.6. Again, users may choose to restrict traversal to paths

Id	Sentence
1	This tests for great usability with mice.
2	We use tests to measure usability with mice and keyboard.
3	Why should we test usability?

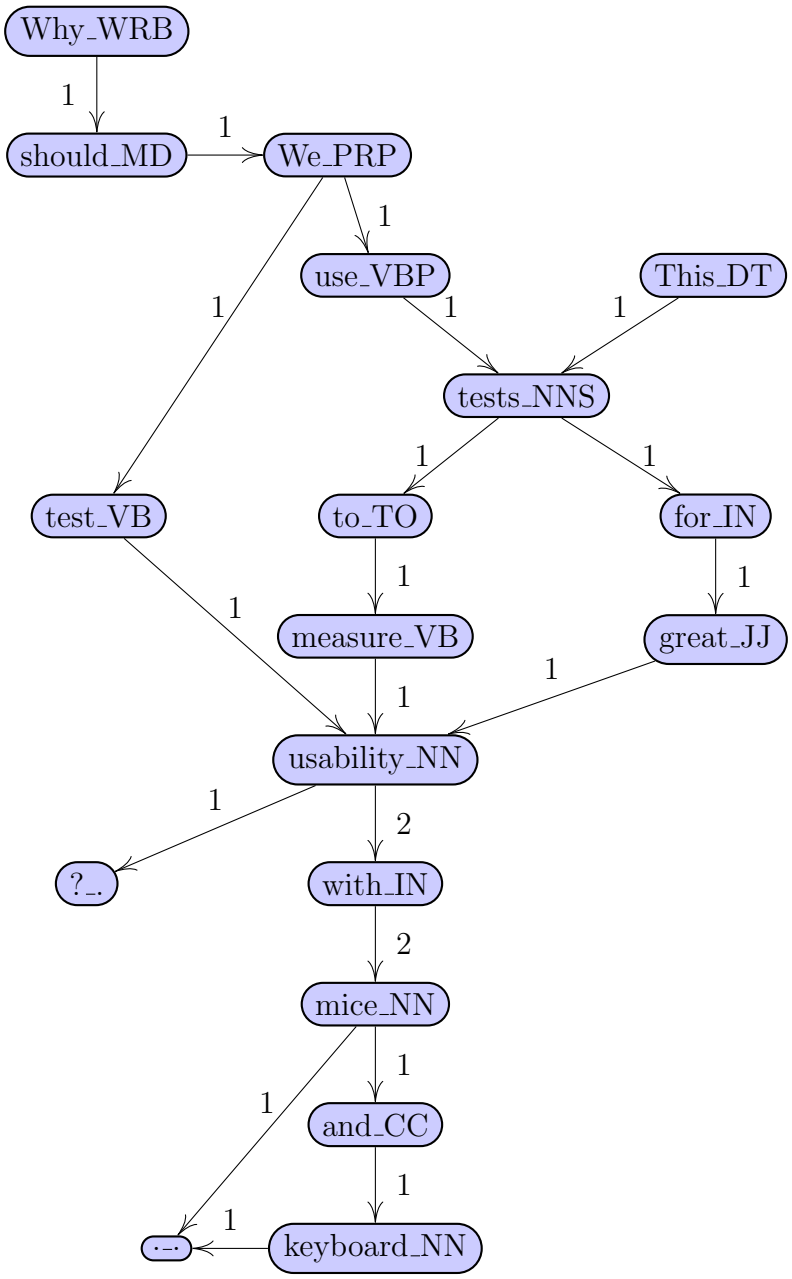

Figure 3.4: ***Numeric Graph.*** **An example numeric graph for a given set of sentences.**

starting at start nodes and ending at end nodes, but it is not recommended since the graph does not capture word positions as a primary goal.

The path scoring is further simplified from the sentence-id graph since we only keep numeric counts. Consider an path in the numeric graph $W = w_1, ..., w_k$ where w_1 is the starting node and w_k is the ending node. The path score can be computed using the following formula based on Equation 2.3 in SPORK:

$$s(W) = \frac{1}{|W|} * \Bigg[tfidf(w_1) + e(w_1, w_2) * tfidf(w_2) + \sum_{i=3}^{k} \Big(\log_2 |w_1, ..., w_i| * e(w_{i-1}, w_i) * tfidf(w_i) \Big) \Bigg] \tag{3.3}$$

where $|W|$ is the length of the path, $|w_1, ..., w_i|$ is the length of the subpath between w_1 and w_i. The function $e(a, b)$ returns the edge weight between nodes a and b. The function $tfidf(i)$ returns the TF-IDF score of the word represented by node i.

Moving from the conceptual design, the next chapter discusses the core implementation details of each layer in WHISK.

Chapter 4

IMPLEMENTATION

This chapter covers the implementation details behind WHISK. All modules written for WHISK are implemented using *Python 3* and its available packages.

4.1 Data Collection

As the beginning of the pipeline, data collection is the responsibility of the driver program. The driver must convert its input corpora into WHISK's generic data model covered in Section 3.1. The generic data model is implemented as *namedtuple*s in Python as shown in Figure 4.1. Since Python's *list* and *dict* (dictionary) objects are loose on type restrictions, the stated type restrictions in Figure 4.1 are to be respected by the developer. The primary objects of interest for the driver during initialization are the *Corpus* and *Text* objects.

WHISK provides a list of model creation methods to help ease the creation of model objects. Most list and dictionary parameters are optional by making use of default parameters while the required parameters are expected to be appropriate non-None values. Parameters that have the *None* default value are initialized appropriately to their type. Lists and dictionaries are initialized with empty lists or dictionaries respectively.

The *Corpus* object allows for all optional parameters because the driver may not have the information it needs yet. The *Text* object requires the plaintext string during creation since the object should only be used to represent a piece of text. *Cluster*, *Sentence*, and *Word* objects, on the other hand, are more internal since they are created and handled by the pipeline, so fields such as `texts`, `words`, and `text` should

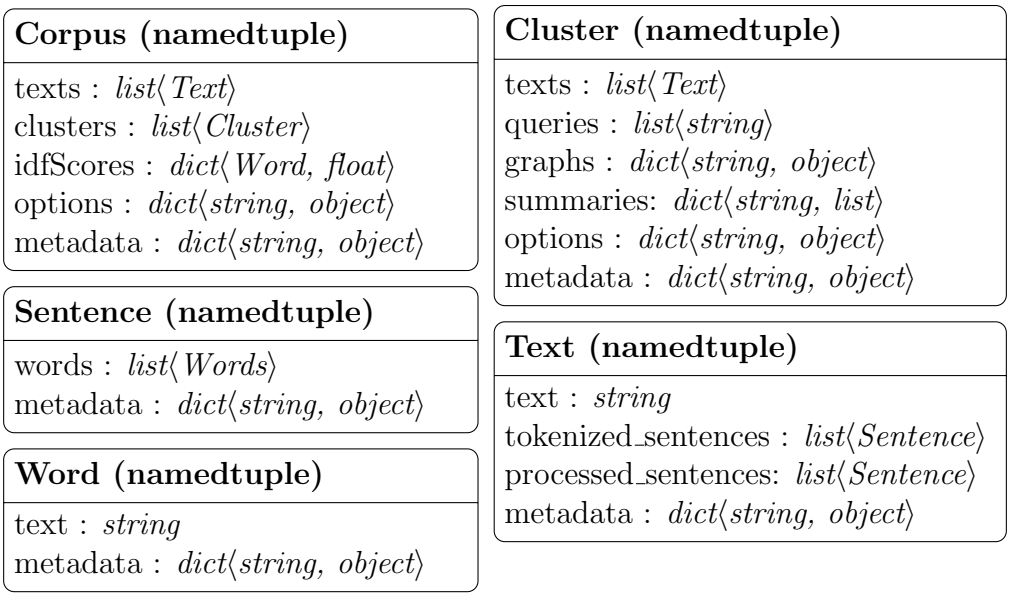

Figure 4.1: *WHISK Data Model Implementation.* An overview of the generic data model implementation in WHISK.

already be known during creation.

WHISK model creation methods are as follows:

- `new_Corpus(texts=None, clusters=None, idfScores=None, options=None, metadata=None)`

  Creates a new Corpus object with the provided fields

- `new_Cluster(texts, tfidfScores=None, queries=None, graphs=None, summaries=None, options=None, metadata=None)`

  Creates a new Cluster object with the provided fields

- `new_Text(text, tokenized_sentences=None, processed_sentences=None, metadata=None)`

  Creates a new Text object with the provided fields

- `new_Sentence(words, metadata=None)`

  Creates a new Sentence object with the provided fields

- `new_Word(text, metadata=None)`

  Creates a new Word object with the provided fields

4.2 Preprocessing

The preprocessing stage of WHISK performs tokenization, part-of-speech tagging, stopword filtering, stemming, keyword ranking, and clustering.

Tokenization. Tokenization is used to allow WHISK work with small units that compose the text. The original text from a *Text* object is first transformed to lowercase letters and then tokenized at the sentence and word level using *NLTK*'s [29] `sent_tokenize` for sentence tokenization and `word_tokenize` for word tokenization. Words that are of one character are automatically ignored.

The tokenized sentences and words may go through optional part-of-speech tagging before being stored inside *Sentence* and *Word* model objects. The original text string is also saved with the key `'text'` in metadata dictionary of *Sentence* objects. These saved objects are use during sentence extraction on the original text. Separate *Sentence* and *Word* model objects are created for the other processing steps during summarization. Stopword filtering and stemming may be performed for these processed sentences.

Part-of-Speech Tagging. Part-of-speech (POS) tagging allows more detailed NLP work by extensions of WHISK. The developer may enable or disable POS tagging through a boolean value with the key `'tag_pos'` in the *Corpus* object's *options* dictionary. The tagging is done by the Java-based Stanford POS Tagger [58] using the left3words model: *wsj-0-18-left3words-nodistsim.tagger*. Since the Stanford POS Tagger is Java-based, a JVM process would have to be started every time we call the

tool via command-line. Since we may be tagging thousands of sentences, the start-up overhead would slow down the pipeline since the pipeline is linear. To minimize the overhead, we extend off *NLTK*'s [29] own interface class `StanfordPOSTagger` as the `POSTaggerServer` to manage the Stanford POS tagger instance.

WHISK's `POSTTaggerServer` allows for POS-tagging by keeping only one instance of the Stanford POS Tagger instance alive throughout its lifetime. The Stanford POS Tagger continuously reads one line and outputs the tagged version. Since the Stanford POS Tagger would wait on reading from an empty pipe, `POSTTaggerServer` takes advantage of this by passing sentences to the instance and returning the tagged version by reading from the output without ending the process. The `POSTTaggerServer` makes use of a separate writer thread to the instance to avoid write blocking on full pipes.

Methods in a separate POS tagger module allow the developer to interface with the `POSTTaggerServer`. Appropriate initialization and cleanup methods provide for creation and destruction of the class along with the Stanford POS Tagger instance. `tag_sentence` and `tag_sentences` allow the developer to tag either by single sentences or by batch.

Stopword Filtering. Stopword filtering allows developers with domain expertise to filter out any words that would not be meaningful in their domain. The default list makes use of *NLTK*'s [29] `stopwords` list for English and a custom list of standard punctuations. Tokens that match any item in the stopword list are ignored. The developer can specify their own stopword list by placing the list into the *Corpus* object's *options* dictionary with the key `'removable_tokens'`.

Stemming. The stemming technique maps variants such as verb or adjective forms of the same word to a common root [30]. For example, "having" would be stemmed to

"have." WHISK makes use of *NLTK*'s [29] `SnowballStemmer`. Stemming's mapping of word variants can lead to better redundancy for the Opinosis graph since more commonality is introduced. Stemming may be turned on or off through a boolean value with the key `'stem'` in the *options* dictionary of the *Corpus* object.

Keyword Ranking. WHISK follows SPORK [32] in using the TF-IDF model [49][56] for ranking keywords. Since the TF-IDF vectors are also be used during clustering, WHISK uses the `TfidfVectorizer` from *scikit-learn* [41] which also provides for the clustering algorithm.

Before clustering, WHISK only obtains the inverse document frequencies (IDF) of each token in the corpus. The *Corpus* object's *idfScores* dictionary contains the IDF token scores. After clustering, the term frequencies in each cluster multiplies with the respective inverse document frequencies of the overall corpus to give the full TF-IDF measure for a token within a cluster. The cluster TF-IDF scores are saved into the respective *Cluster* object's *tfidfScores* dictionary.

Clustering. The agglomerative hierarchical clustering in SPORK [32] clusters a low number of documents well by restricting to only the top-level comments, but with generic summarization, WHISK will need to be able to handle hundreds or thousands of documents. Each run of the agglomerative hierarchical clustering compares all documents to each other and merges the two closest clusters if the similarity satisfies a threshold [32]. SPORK continues run the agglomerative hierarchical clustering until there are no more clusters to merge [32]. This procedure gets very slow to around 20 minutes during a sample run of over one hundred documents.

To make clustering quicker, WHISK adopts Density Based Spatial Clustering of Applications with Noise (DBSCAN) [15] as its clustering algorithm. DBSCAN employs the algorithm described in Algorithm 1 and Algorithm 2. DBSCAN takes

two parameters: ϵ and *MinPts*. ϵ denotes the maximum distance between any two points to be considered neighbors. Cosine distance, equivalent to 1 - cosine similarity, is used as the distance metric between TF-IDF vectors. *MinPts* denotes the minimum number of neighboring points for a point to be considered a *core* point.

Core points are crucial in combining clusters together. Each core point may start out as its own cluster with its neighboring points. If any two core points are neighbors, they merge into the same cluster. This merging propagates to neighboring non-core points as well. As long as there are neighboring core points, a cluster grouping may continue to propagate. The full details on DBSCAN can be found through its original paper [15].

WHISK makes use of the *scikit-learn* [41] package for its DBSCAN clustering implementation. The ϵ and *MinPts* parameters can be passed via the *options* dictionary in the *Corpus* object using the keys `'epsilon'` and `'minPts'`. The default values of these parameters are 0.5 and 5 respectively. By default, WHISK allows *scikit-learn*'s DBSCAN to compute a full pair-wise NxN distance matrix. However, this may not be feasible when the number of points (or texts in this case) increase the size of the distance matrix so much that it does not fit into memory.

WHISK allows the calculation of a sparse distance matrix in Python with the same distance function from *scikit-learn*. This option can be enabled through a boolean with the key `'sparse_dist'` in the *Corpus* object's *options* dictionary. The custom sparse distance matrix only keeps distances below or equal to ϵ and passes it to *scikit-learn* which ignores missing distances. This method saves memory by keeping only the relevant distances, but it also loses performance since the distance calculations are done linearly in Python compared to optimizations and vectorizations in *NumPy* [59].

Using the previously generated TF-IDF vectors for the corpus texts, *scikit-learn*'s

Algorithm 1: DBSCAN Clustering

Function $DBSCAN(D, \epsilon, MinPts)$

```
// D is the collection of points to cluster.
// ϵ and MinPts are floating point numbers
Core := ∅
Compute distance between all pairs of points in D
foreach point P in D do
    let N be the P's neighbors in D within ϵ distance
    if |N| ≥ MinPts then
        Core := Core ∪ {P}
ClusterId := 0
foreach point C in Core do
    ClusterId := ClusterId + 1
    if cluster(C) = ∅ then
        ExpandCluster(D, C, Core, ClusterId)
Clusters := ∅
for i := 1 to ClusterId do
    Cluster[i] := {p ∈ D | cluster(d) = i}
    Clusters := Clusters ∪ {Cluster[i]}
Noise := {p ∈ D | cluster(p) = ∅}
Border := D - (Core ∪ Noise)
return clusters, Core, Border, Noise
```

Algorithm 2: Auxiliary Recursive Function for DBSCAN

```
Function ExpandCluster(D, P, Core, Cid)
    let N be the P's neighbors in D within ε distance
    foreach point M in N do
        cluster(M) := Cid
        if M ∈ Core then
            ExpandCluster(D, M, Core, Cid)
```

`DBSCAN` method clusters the texts and outputs the list of cluster labels. WHISK groups the texts by the cluster labels into new *Cluster* objects and saves them into the *Corpus* object's *clusters* list. The corpus texts finish the preprocessing stage here and may now move onto the summarization stage for further processing.

4.3 Summarization

The summarization stage works with the preprocessed corpus texts to select representative sentences as the corpus summary. Each cluster discovered during preprocessing undergoes query generation, graph generation, graph traversal, path scoring, and sentence extraction.

Query Generation. Query generation finds the top N keywords in a text cluster using the χ^2 measure [33]. WHISK makes use of SPORK's [32] χ^2 implementation for discovering the keywords. Small changes to the syntax were made for compatibility with *Python 3*, but overall algorithm remains the same. An interface method `generate_queries` is available under the `QueryGeneration` module to perform the query generation using the model objects. The discovered keywords are saved into the *Cluster* object's *queries* list.

To make use of query generation, the developer must set a `true` boolean value with the key `'use_queries'` and the number of query words to be generated for each cluster with the key `'num_queries_per_cluster'` in the *Corpus* object's *options* dictionary. The default number of query words is 10.

Sample runs of the query generation showed that the process takes over 100 seconds for a cluster with over 45,000 words and over 50 seconds for a cluster with over 35,000 words. Smaller clusters with under 10,000 words finish around or under five seconds. Query generation may be better used for offline summarization, rather than for online summarization.

Graph Generation. The Opinosis graph captures the redundancy present in the corpus texts using word sequences. WHISK generates a graph for each cluster using either the `PRIGraph`, `SentIdGraph`, or the `NumericGraph` class. The three classes map to the graph models described in Chapter 3. The `PRIGraph` implements the original Opinosis graph model, the `SentIdGraph` implements the sentence-id graph model, and the `NumericGraph` implements the numeric graph model. It is up to the developer to create the appropriate graph object and provide the graph the appropriate *Cluster* object to work on through either the graph's constructor or the `create_graph` class method. The developer may save the graph objects into the *graphs* dictionary in the *Cluster* object.

For each sentence in a cluster, the graphs create new nodes as necessary for each word and update the edge properties following Algorithm 3. Nodes within a graph are uniquely identified using a string identifier. If POS tagging is used, the string is formatted as `word_pos`. Otherwise, the word's plaintext string is used. The `OpinosisGraph` and `OpinosisNode` superclasses contain common data and methods between the graph variants.

Algorithm 3: General Graph Generation

```
Function createGraph(cluster)
    startNodes := ∅
    endNodes := ∅
    numSentences := |getSentences(cluster)|
    for sentId := 0 to numSentences do
        sent := getSentence(cluster, sentId)  // Get sentence by index
        numWords := |getWords(sent)|
        prevNode := ∅
        for wordNdx := 0 to numWords do
            word := getWord(sent, wordNdx)  // Get word by index
            node_id := getId(word)  // Generate id for word
            node := createNode(word, sentId, wordNdx)
            nodes[node_id] := node
            if wordNdx := 0 then
                startNodes := startNodes ∪ {node}
            if wordNdx := (numWords - 1) then
                endNodes := endNodes ∪ {node}
            Update positions using sentId and wordNdx OR update edge
            weights using prevNode and node
            if prevNode ≠ ∅ then
                prevNode.next := node
                node.prev := prevNode
            prevNode := node
    return nodes, startNodes, endNodes
```

The `OpinosisGraph` class exists as an abstract graph class for the three graph implementations. It has common functions between the graph implementations such as the skeletons for graph creation, score calculations, and selection of the top redundant paths in the graph. Common data elements include a reference to the working *Cluster* object, the dictionary for mapping node identifiers to nodes in the graph, the sets of start and end nodes, the boolean representing whether or not queries are to be used, and bookkeeping variables for tracking the number of sentences and words in the graph as well as for enabling verbose information printing.

The `OpinosisNode` class represents the basic node for the three graph variants. This class stores its string identifier, the plaintext string of a word, metadata about the node, and adjacent nodes. The POS tag, if available, is saved with the key `'pos'` in the node's *metadata* dictionary. Adjacent nodes are saved in sets separated by the edge direction: `in_nodes` for incoming edges and `out_nodes` for outgoing edges. Specifically for the `PRIGraph`, the `PRINode` extends off the `OpinosisNode` by adding a *positions* list that stores the sentence identifier and word positional information in tuples. `SentIdGraph`, on the other hand, uses the `SentIdNode` class which adds a *positions* set to the `OpinosisNode` to store sentence identifiers. Each graph variant provides their own node creation method for dealing with custom node classes.

Both `PRIGraph` and `SentIdGraph` maintain positional information for path scoring. Since neither graphs maintain edge weights, these subclasses do not add additional data elements to the `OpinosisGraph` superclass.

`NumericGraph`, however, only uses the base `OpinosisNode` since it does not store any positional information. Instead, `NumericGraph` adds an *edges* dictionary to store integer edge weights. Each edge is identified by a tuple of the starting `OpinosisNode` and the ending `OpinosisNode`.

Graph Traversal. WHISK discovers redundant word sequences by traversing paths in the generated Opinosis graphs. The general traversal algorithm is described in Algorithm 4 and Algorithm 5. The graph is traversed recursively in a depth-first search (DFS) fashion until the candidate path fails to satisfy the minimum redundancy, distinct node, or maximum path length requirements.

Algorithm 4: Redundant Sequence Discovery

```
Function getRedundantSequences(graph, tfidfScores, minRedundancy,
startEndRestrict, maxLength, distinctNodes)
    seqs = ∅
    if startEndRestrict then
        startingSet := graph[startNodes]
    else
        startingSet := graph[nodes]
    foreach node n in startingSet do
        newPath := [node] // Start a new list to represent a path
        foreach neighbor m of n do
            seqs := seqs ∪ traverse(graph, node, newPath, tfidfScores,
            minRedundancy, startEndRestrict, maxLength, distinctNodes)
    return seqs
```

The minimum redundancy requirement keeps traversal to potentially interesting paths and prunes the search tree. WHISK uses 3 as the default minimum redundancy value. A good minimum redundancy value can help speed up searching but discovering the value may be difficult. Domain knowledge on the input text allows one to infer the expected background redundancies. Without domain knowledge, one may possibly consider using a logarithmic function on the number of input texts or sentences. Developers can set their own requirement value using the key `'min_redundancy'` in

Algorithm 5: Recursive Graph Traversal

```
Function traverse(graph, n, curPath, tfidfScores, minRedundancy,
startEndRestrict, maxLength, distinctNodes)
    seqs := ∅
    if (maxLength > 0) ∧ (¬distinctNodes ∨ ({n} ∩ curPath = ∅)) then
        newPath := curPath + [n]  // Append current node to path
        redundancy := getRedundancy(newPath)
        if redundancy ≥ minRedundancy then
            score := getScore(newPath, tfidfScores)
            if (¬startEndRestrict) ∨ ({n} ∩ graph[endNodes] ≠ ∅) then
                seqs := seqs ∪ {newPath}
            foreach neighbor m of n do
                seqs := seqs ∪ traverse(graph, m, newPath, tfidfScores,
                minRedundancy, startEndRestrict, maxLength - 1,
                distinctNodes)
    return seqs
```

the *Corpus* object's *options* dictionary.

To prevent sequences from containing duplicate words, WHISK has an option for a distinct node requirement. By default, this requirement is enabled to avoid word sequences that repeat the same phrase. Additionally, the `SentIdGraph` and the `NumericGraph` may get into a cycle in the graph since they does not maintain any word positional information. The distinct node requirement prevents any cycles from developing. To change this setting, developers may set the boolean value with the key `'distinct_nodes'` in the *Corpus* object's *options* dictionary.

Because depth-first search observes one path at a time, WHISK limits the search depth to prevent getting stuck while searching a very long branch or a cycle. WHISK searches up to a maximum path length of 20 by default. This option may be changed with the key `'max_path_length'` in the *Corpus* object's *options* dictionary.

`PRIGraph` has an extra integer value `'max_gap'` available in the *Corpus* object's *options* dictionary. This option maps to the maximum gap in original Opinosis path restrictions. WHISK has the maximum gap set to 3 by default. The other two models ignore this value since they lack the word positional information to track gaps between words.

If queries are enabled, candidate paths are further filtered by the query keywords. SPORK [32] uses forward and backward propagation from the query word's node in the graph, if it exists, to optimize path searching. This method would work as long as the nodes are not identified with POS tags in addition to the word's plaintext. WHISK takes a safer approach by exploring all candidate paths and performing a simple filtering check. Only paths which contain at least one of the query keywords are saved into the resulting sequence pool. There are potential time savings using SPORK's method if one is using queries in a fairly dense graph where a large number of paths satisfy the minimum redundancy requirement. However, in the interest of

development time, WHISK only implements the simple filter.

Overall, the graph traversal in `PRIGraph`, `SentIdGraph`, and `NumericGraph` are identical and differ in only in the path redundancy and score calculation.

Path Scoring. The redundancy and path scoring methods for Opinosis, sentence-id, and numeric graphs described in Chapter 3 allow WHISK to rank word sequences. For each score calculation, WHISK uses *NumPy*'s [59] implementation for log_2. To easily get the top sequences, WHISK stores candidate sequences based on the path score and the length of the sequence in a max heap. Once all candidates have been added to the heap, the top N sequences can be selected based on the score and length of the path as well as the similarity of a candidate sequence to sequences already selected. The number of top sequences can be set using the key `'num_top_seqs'` and the maximum similarity value (between 0 to 1 following cosine similarity or `None` to disable) can be set using the key `'max_path_similarity'` in the *Corpus* object's *options* dictionary. By default, WHISK tries to retrieve the top 10 sequences with the maximum similarity comparison disabled.

WHISK iterates through the max heap to provide the final top sequences as shown in Algorithm 6. The iteration order follows the ordering of the max heap: primarily by path score and secondarily by path length.

The comparison of the candidate sequence with selected top sequences prevent similar sequences from crowding and overtaking the top spots. For example, "cold day" would be very similar to "cold winter day." Having both sequences as top sequences would not provide any more insight than had we picked "cold winter day." By filtering out sequences that are too similar, WHISK has a better chance of capturing a more diverse set of redundant word sequences.

Algorithm 6: Selection of Top Sequences

Function *getTopSequences(heap, num, maxPathSimilarity)*

```
topSeqs := ∅
while ¬isEmpty(heap) ∧ (num > 0) do
    seq := pop(heap)
    if (maxPathSimilarity = ∅) ∨ (topSeqs = ∅) ∨
    (max({cosine_similarity(seq, x)|x ∈ topSeqs}) ≤ maxPathSimilarity)
    then
        topSeqs := topSeqs ∪ {seq}
        num := num - 1
return topSeqs
```

Sentence Extraction. Using the scored top sequences, WHISK can now extract representative sentences from the original text for each cluster following Algorithm 7. Each word sequence is compared to every tokenized sentence in the cluster using SPORK's [32] Local Alignment similarity implementation. The implementation has been tweaked in WHISK to have the similarity value be between 0 and 1 like cosine similarity. The sentences with the highest non-zero similarity to the word sequences are selected as the representative sentences. Ties are broken by choosing the longer length sentence. The selection can also be constrained with a minimum sentence length. By default, the minimum sentence length is 7. Developers can change this option using the key `'min_sent_len'` in the *Corpus* object's *options* dictionary. The developer may choose to save the cluster summary into the *summaries* dictionary in the *Cluster* object. The list of extracted sentences are outputted in descending order of the graph path score multiplied with the sentence similarity score.

The Local Alignment similarity is implemented using the Smith-Waterman algorithm's [55] matrix values. The Smith-Waterman algorithm generates a NxM matrix,

Algorithm 7: Sentence Extraction

```
Function extractSentences(sentences, seqs, minSentLength)
    repSentences := ∅
    foreach sequence S in seqs do
        Calculate similarity between S and every sentence in sentences
        let T be the the sentence of the highest similarity
        if length(T) ≥ minSentLength then
            repSentences := repSentences ∪ {T}
    return repSentences
```

where N is the length of the first string and M is the length of the second, as part of its dynamic programming approach. Each matrix cell holds a matching score for its position in matching the two input strings. The recurrence relation is as follows (adapted [55]):

$$H(i,j) = max\{0, H(i-1,j-1) + (MATCH \text{ if } a_i = b_j \text{ else } MISMATCH), \\ H(i-1,j) + DELETION, \\ H(i,j-1) + INSERTION)\} \tag{4.1}$$

where a is the first string, b is the second string, and a_i and b_j are the i^{th} character of string a and the j^{th} character of the string b. Each situation's value in the recurrence relation is set to the following: $MATCH = 2, MISMATCH = -1, DELETION = -1,$ and $INSERTION = -1$.

As the matrix is filled with computed values, the maximum value is tracked as the similarity value. Once the maximum has been determined, the value is normalized by the maximum possible score which is equal to $(min(length(a), length(b)) * MATCH)$.

4.4 Service

The implementation of the service layer is entirely up to the developer. One can have the driver be a command-line utility, perform HTML generation, or enable a web API. The presentation of the summary sentences is completely unrestricted since WHISK simply provides the sentences in plaintext.

4.5 Example Drivers

Digital Democracy. A case study with pull quotes from state legislative hearings is used to test the practicality of applying WHISK to a complex domain. Each hearing has been already transcribed into text segments. For the case study, we provide a simple command-line driver that takes transcription datasets in csv format and outputs the representative sentences as pull quotes. Each transcribed segment of a speaker is considered an utterance. In this domain, an utterance is a text unit, therefore, we create *Text* objects on each segment.

With a little domain knowledge, we provide a custom stopword list (`dd.stop`) based on Lewis's smart stopword list in *RCV1* [26]. From reading the transcription, the hearings often contain many greetings message and administrative utterances that do not add meaning to the discussion. Words such as "sir," "ma'am," or "secretary" tends to carry little weight. The custom stopword list removes unnecessary phrases that often arise. NLTK's [29] English stopword list is added as well during runtime.

The driver also provides functionality to generate HTML pages. The generated pages make use of basic carousels from the web framework Bootstrap. This allows for displaying and animating representative sentences one by one in a scrolling fashion. These pages merely serve as a prototype to see how pulled quotes may be presented to serve and aid the user.

Since pull quotes should be highlights, a requirement for pulling at most 10 quotes naturally came about. To select the 10 quotes, we created two basic selection methods: *TopScores* and *RoundRobin*. The *TopScores* scheme selects the sentences based on the highest value from multiplying the graph sequence's path score with the matching Local alignment similarity value. The *RoundRobin* scheme, on the other hand, goes to each cluster in a round robin fashion and picks the highest value like *TopScores* does. Clusters are ordered by their number of words.

To enable convenient evaluation, the driver supports saving the results to a CSV file. The CSV file contains the union of all selected sentences and uses column fields as flags to indicate which graph method and which selection method resulting in outputting each quote.

Plaintext. For more generic plaintext files, we provide a driver that performs similar functionality to the Digital Democracy pull quote driver. The input plaintext files are expected to present each text unit in a separate line. Instead of saving the output as a CSV file, the plaintext driver outputs each representative sentence into a line in a plaintext output file. Lewis's smart stopword list in *RCV1* [26] (saved as english.stop) combined with NLTK's [29] English stopword list compose the list of removable tokens. This driver is used to create summaries for evaluating with ROUGE [27].

Chapter 5

VALIDATION

Three experiments have been run in order to evaluate different parts of WHISK. The first experiment involves a processing time comparison to evaluate the performance improvements of the new graph models: sentence-id and numeric. The second experiment uses an automatic summary evaluation tool created by Lin called ROUGE [27] with the dataset provided by Ganesan et al. [19] to evaluate the summaries generated by the different graph models. The last experiment is a study of WHISK's ability to generate pull quotes from a number of bill discussions that took place in the California state legislature.

5.1 Graph Performance Comparison

The proposed sentence-id and numeric graph models simplify the original Opinosis graph model to save processing time. To grasp the performance improvements, we isolate time recordings for graph creation and traversal of the graph for redundant sequences. On every run of the pipeline, timings are recorded and record for all graph variants used. Since a graph is created and processed for every text cluster, a single WHISK run may have multiple timings outputted.

The datasets from the Digital Democracy project shown in Table 5.1 are used to run performance timings. Each dataset contains all hearings from a single bill. The bills covered by the datasets include: SB145, SB277, SB34, SB530, and SB9.[1] Performance timings are uniquely identified by the dataset name, the cluster number, and the procedure (either graph creation or graph traversal). Abbreviated graph

[1]For 2015-2016 legislative session.

Dataset	# Utterances
SB145	598
SB277	5510
SB34	976
SB530	293
SB9	3018
Average	2079

Table 5.1: *Dataset of bills for graph performance comparison*

variant names are used for column names in Table A.1: Opinosis graph (PRIGraph) to *PRI*, sentence-id graph (SentIdGraph) to *SentId*, and numeric graph (NumericGraph) to *Numeric*.

The operational settings of WHISK during this experiment are shown in Table 5.2. All values are static except for the `'min_redundancy'` for `numeric` graphs. Since the `numeric` graphs only use edge weights without any restriction to share common sentences, paths can cross between sentences. This crossing can easily increase the potential number of paths which increases the processing workload. To better prune the search tree, we need a different minimum redundancy value for numeric graphs than the ones appropriate for Opinosis and sentence-id graphs. A higher minimum redundancy requirement is determined using the following function which scales the static minimum redundancy value by the natural log on the number of texts in the cluster:

$$minRedundancy' = min\{3, max\{1, ln(numClusterTexts) - 5\}\} * minRedundancy \tag{5.1}$$

The shift of 5 makes the scaling occur after the number of cluster texts surpasses $e^6 \approx 400$. We limit the scale to a factor of 3 to prevent losing too many candidates while keeping the minimum possible value at the static value.

Setting	Value
removable_tokens	*custom stopword list: dd.stop*
stem	true
tag_pos	false
epsilon	0.5
minPts	3
sparse_dist	false
use_queries	false
num_queries_per_cluster	10
num_top_seqs	20
max_path_similarity	0.3
min_redundancy	5
max_gap	3
start_end_restrict	false
max_path_length	20
distinct_nodes	true
min_sent_len	15

Table 5.2: *Operation settings of WHISK*

Dataset	# Clusters	# Usable Clus.	Max # Utt.	Min # Utt.	Avg # Utt.
SB145	21	5	458	3	27.1904
SB277	206	40	3764	3	23.3349
SB34	28	8	604	3	31.3214
SB530	12	5	223	3	22.5
SB9	42	16	2126	3	66.7142
Average	61.8	14.8	1435	3	34.2122

Table 5.3: *Cluster statistics*

Dataset	Max # graph nodes	Min # graph nodes	Avg # graph nodes
SB145	1736	1	102.9047
SB277	6289	1	50.1941
SB34	2119	2	101.4285
SB530	881	3	93.3333
SB9	4834	2	150.6904
Average	3171.8	1.8	99.7102

Table 5.4: *Cluster graph statistics*

Results. For each dataset, we recorded cluster statistics and timings for graph creation and traversal. Clusters that generated a summary from at least one graph model are considered "usable" clusters. Statistics on number of clusters, number of usable clusters, number of utterances per cluster, and number of graph nodes per cluster are shown in Tables 5.3 and 5.4. For text clusters with less than 200 words, the timings are similarly low and are affected by other processes being executed. For this validation, we will only look at data for clusters with 200 words or more.

Table A.1 shows the graph creation and traversal timings for the three graph methods by dataset and by cluster id (*CId*). Figures 5.1 and 5.2 visualize the comparison

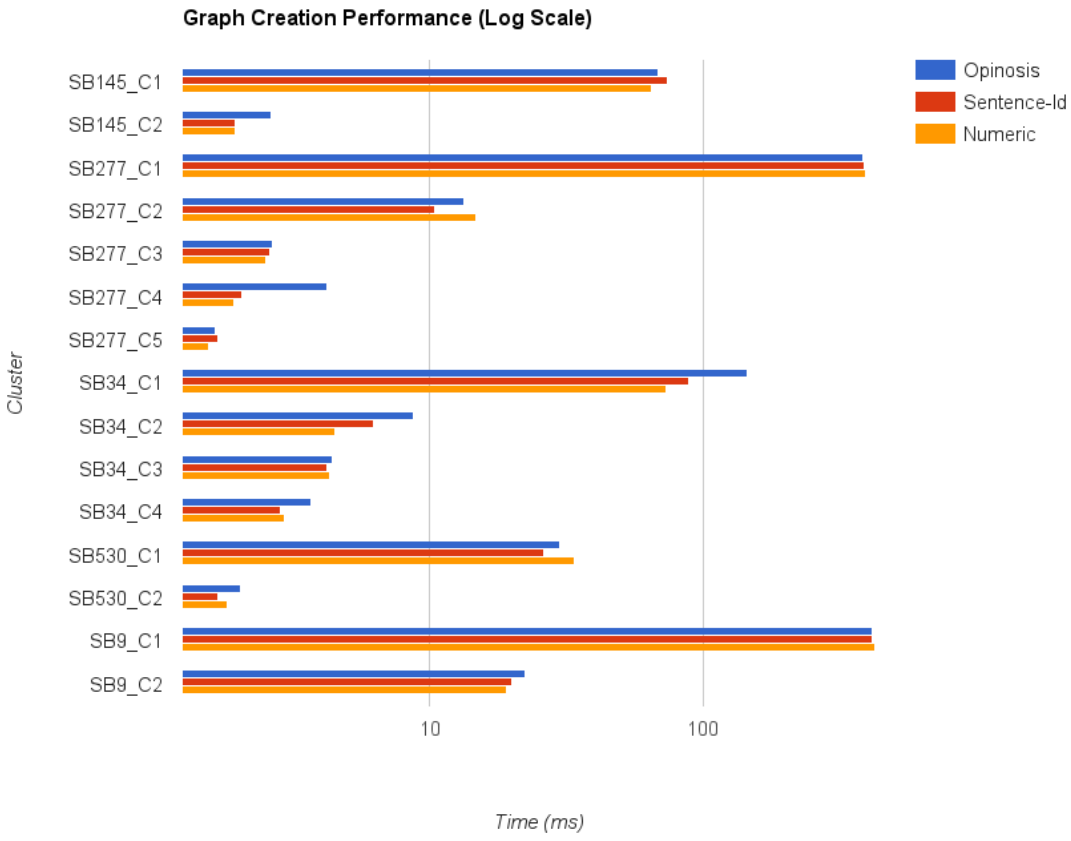

Figure 5.1: ***Graph creation timings.*** **Comparison of graph creation speed for the three graph models in logarithmic scale**

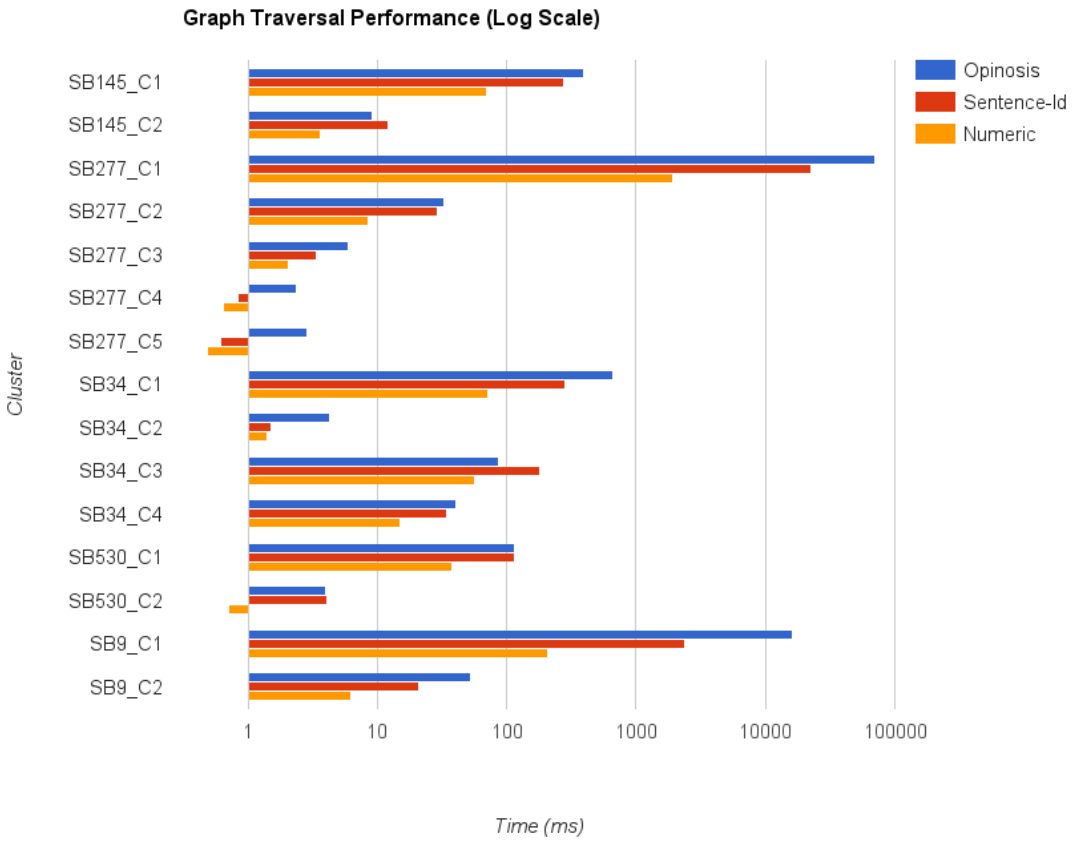

Figure 5.2: ***Graph traversal timings.*** **Comparison of graph traversal speed for the three graph models in logarithmic scale**

between three graph models' performance. Each cluster is denoted in the following format: $\langle Dataset \rangle$_C$\langle ClusterId \rangle$. Since the recorded timings vary from short durations to fairly long durations, Figures 5.1 and 5.2 utilize a logarithmic scale to make the differences easier to see in the smaller cases. The logarithmic scale portrays values below one as "negative" bars that go further left of the 1-value mark the smaller they are.

The graph creation timings between the new graphs, as shown in Figure 5.1, are fairly similar. In the clusters *SB145_C1*, *SB277_C1*, *SB277_C2*, *SB277_C5*, *SB530_C1*, and *SB9_C1*, sentence-id or numeric graph creation take longer than Opinosis graph creation, but the difference is not too large. Graph creation times between the graph models are comparable overall. The bigger area for time savings is during graph traversal.

The sentence-id graph pulls ahead of Opinosis in most graph traversal cases while numeric graph beats Opinosis in all graph traversal cases shown in Figure 5.2. For the *SB277_C1* cluster, Opinosis graph took 70064.53 ms while sentence-id graph took 22378.12 ms and numeric graph took 1940.99 ms. For the *SB9_C1* cluster, Opinosis graph finished in 16211.79 ms while sentence-id graph finished in 2386.54 ms and numeric graph finished in 206.85 ms.

However, sentence-id graph traversal performs slower than Opinosis graph traversal in the *SB145_C2* and *SB34_C4* clusters. This slowdown is due to many similar sequences with high scores being discovered while traversing the sentence-id graph. WHISK filters out similar sequences using a heap data structure as described by Algorithm 6 in Chapter 4. With a high number of similar sequences, more time is spent going through the top sequences in the heap to filter out sequences that have cosine similarities above the maximum threshold.

We can calculate the percentage of time improvement from a time S to a time T by

	Sentence-Id	Numeric
Average % time improvement (graph creation)	14.8228	14.3367
Average % time improvement (graph traversal)	29.1596	75.0436
Overall average % time improvement	21.9912	44.6901

Table 5.5: *Aggregate performance measures of new graphs*

using the following equation: $improvement = \frac{S-T}{S} * 100$. We then averaged the time improvements to get an idea of the overall time improvements shown in Table 5.5. Using the sentence-id and numeric graph models, we gain a modest improvement of around 14% time improvement in graph creation on average over the Opinosis graph. However, we achieve around 29% or 75% graph time improvement in graph traversal on average with sentence-id or numeric graphs respectively compared to the Opinosis graph. Overall, we can save around 22% and 44% of graph operation time by using sentence-id or numeric graphs respectively.

5.1.1 Summary Similarity

In addition to timings, we wanted to see how similar the output summaries are from using the different graph models. This observation does not act as a performance metric, but it provides some insight on how differently the graph models operate.

The generated summaries by each graph model are compared using Jaccard similarity [23]. Specifically, sentences extracted using sentence-id and numeric graph are compared to sentences extracted using the Opinosis graph. Two sentences are considered equal only if they are identical. We compare the sets of sentences in a pairwise fashion with the Jaccard similarity coefficient [23]:

$$sim = \frac{|A \cap B|}{|A \cup B|} \tag{5.2}$$

We calculate the similarity for the first i sentences where i comes from the range

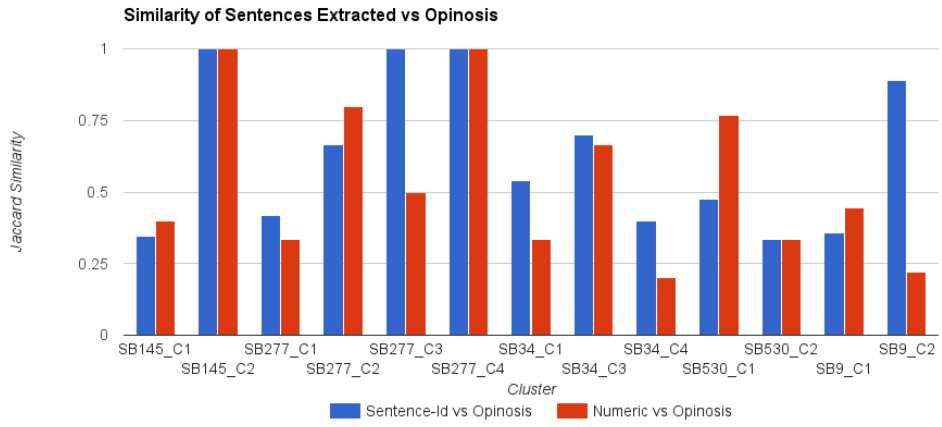

Figure 5.3: *Jaccard similarities of extracted sentences.* Extracted sentences using Sentence-Id and Numeric graphs are compared to those using the Opinosis graph.

$[1...L]$. L equals the smaller total number of sentences between the two sets. For example, when $i = 1$, only the first sentence is compared, but when $i = 5$, the first five sentences of both sets are compared. By recording using a range on the number of sentences, we can observe the similarity of extracted sentences by the order of retrieval. These comparisons allow us to see how different the output from the new graph models are from the original.

Tables A.2 and A.3 show the results from the described Jaccard similarity procedure. Only clusters that generate a non-zero number of extracted sentences and has over 200 words were considered for the comparison. We expected that the similarity grows towards 1 as the number of sentences increase, but the results show the similarity decreasing from 1 or a value near 1. Like the time improvements, we can average the Jaccard similarities to get an idea of the overall similarity in Table 5.6. However, instead of averaging using all of the similarities, we use only the similarities of the highest number of sentences for each cluster because we are not observing the order of retrieval. Figure 5.3 visualizes the Jaccard similarity values by cluster used for the average.

	Sentence-Id	Numeric
Average Jaccard Similarity	0.6249	0.5386

Table 5.6: *Aggregate jaccard similarities.* Average jaccard similarities of summaries generated by the new graph models versus summaries generated by the Opinosis graph

As shown in Table 5.6, the overall similarities of the sentence-id and numeric graphs to the Opinosis graph average close to 0.5. This means that around half the sentences retrieved with the new graphs were also retrieved with the Opinosis graph. Using a strict requirement of identical sentences may have contributed to the low similarity. The sentences retrieved with the new graphs may be similarly relevant but not identical to the sentences retrieved with Opinosis. The next experiment uses ROUGE [27], an automatic summary evaluation tool, to compare the n-gram recall of the summaries generated by the different graph models. A study with human evaluators in the later Section 5.3 further explores the performance of the new graphs compared to Opinosis with respect to their precision in extracting relevant sentences.

5.2 ROUGE

Besides comparing the performance measures, we also compared the effective summaries produced using the different graph models. Recall Oriented Understudy for Gisting Evaluation (ROUGE) is a tool created by Lin to perform automatic evaluations of summaries [27]. ROUGE has been validated through comparing its results to the Document Understanding Conference's (DUC) three years of manually labelled summaries. Numerous works [2, 6, 7, 19, 22, 28, 36, 38, 39, 47, 52, 53, 54, 62, 64, 65] use some version of ROUGE for validation.

ROUGE takes gold-standard summaries and system-generated summaries as input and can perform a variety of scoring metrics. In particular, we are interested in

ROUGE-N which performs a n-gram recall. ROUGE-N scores the system-generated summaries by counting n-grams that co-occur in the system-generated summaries and the gold-standard summaries. We specifically use ROUGE-1, which uses unigrams, and ROUGE-2, which uses bigrams.

We used the Opinosis dataset [19] by Ganesan et al. for working with ROUGE. We picked the Opinosis dataset over data from Digital Democracy because we do not have gold-standard summaries for the bill discussions. Since Ganesan et al. published their dataset for public consumption, we make use of the available resources that has been prepared for ROUGE evaluation. However, the Opinosis dataset was created in mind for evaluating abstractive summarization which is a bad fit for WHISK which performs extractive summarization. We do not expect high performing scores from ROUGE because WHISK is not tuned for this type of dataset. Our main goal here is to use ROUGE as an objective measurement to compare the three graph models.

The Opinosis dataset [19] contains 51 corpora of topically aligned sentences from user reviews. Each corpus follows a certain topic in the reviews. Some example topics are "video quality on an ipod nano," "price of Amazon kindle," and "performance of a Honda Accord." The user reviews were retrieved from sources such as *Tripadvisor*, *Edmunds.com*, and *Amazon.com*. Each corpus is stored as a plaintext file with each line containing a sentence from user reviews. Additionally, each corpus comes with four or five gold-standard human-generated summaries that was used to evaluate the abstractive approach presented in the original Opinosis paper [19].

The plaintext driver described in Section 4.5 is used to generate the summaries using the 51 corpora provided by Ganesan et al.'s dataset [19]. Since each sentence in a corpus of this dataset may have come from a different user review, we consider each sentence in a corpus as a text document. This models a situation where multiple people are contributing to a discussion around the corpus topic. We use the

	ROUGE-1			ROUGE-2		
Graph Method	Recall	Precision	F-score	Recall	Precision	F-score
Opinosis	0.3954	0.1212	**0.1584**	0.0824	**0.0221**	**0.0295**
Sentence-Id	**0.4240**	0.1144	0.1520	**0.0948**	0.0174	0.0259
Numeric	0.3791	**0.1221**	0.1558	0.0768	0.0183	0.0259

Table 5.7: *ROUGE-1 and ROUGE-2 scores*

settings for WHISK described in Table 5.2 to generate summaries for each corpus. The summaries are saved as plaintext files and compiled into the appropriate formats for ROUGE using scripts publicly published by Ganesan et al.

Jackknifing [42] is used to reduce bias in the sampling. For each corpus that have N gold-standard summaries, we perform N ROUGE evaluations using $N-1$ gold-standard summaries, which leaves one summary out each time. Every gold-standard summary is left out of the evaluation exactly once. All system generated summaries are used in every evaluation. We enabled the stopword removal and Porter stemmer options in ROUGE for all executions. The resulting scores from the N evaluations are averaged and aggregated into precision, recall, and f-scores presented in Table 5.7.

Results. All summaries generated by the three graph models performed equally poorly as shown in Table 5.7. The two new graph methods perform comparably to the Opinosis graph. The Opinosis graph wins in F-scores in both ROUGE-1 and ROUGE-2, however sentence-id graph has the best recall overall. Both sentence-id and numeric graphs achieved ratings close if not better than the Opinosis graph. To further verify the competence of the two new graphs, the next section describes a study with human evaluations on portions of political discussions extracted by WHISK as summary pull quotes.

5.3 Digital Democracy

To measure the real-world applicability of WHISK, we performed a study with the Digital Democracy project to generate pull quotes from political discussions that would serve as good summaries. Given a dataset of committee hearings on several bills, we extract the top sentences from each bill discussion as the representative pull quotes for the respective bill discussion. We use the settings described in Table 5.2. The resulting pull quotes are then reviewed by human evaluators with domain knowledge on the bill's subject matter. We consider these human evaluators to be experts in the subject matter. The reviews are then used to determine the precision in which WHISK generates relevant pull quotes.

With the help of the Digital Democracy project management, we selected 17 bill discussions (listed in Table 5.9) for 11 bills (listed in Table 5.8) from the 2015-16 legislative session. Statistics on number of clusters, number of usable clusters, number of utterances per cluster, and number of graph nodes per cluster are shown in Tables 5.10 to 5.12. A cluster is considered a "usable" cluster if at least one sentence is extracted while using any of the three graph models. The counts are also averaged in the last row of the tables. As shown in Tables 5.10 and 5.11, the largest clusters contained a large portion of the utterances while the smallest cluster has only a few utterances. A similar pattern follows for the number of graph nodes.

5.3.1 Pull Quote Selection

The Digital Democracy project posed a requirement for a maximum of ten quotes for each bill discussion. For the simple case where WHISK produces ten or less sentences, all of the sentences are selected as pull quotes. However, WHISK may produce more sentences than appropriate. We propose two methods based on scores and cluster

Bill Id	Bill Title
AB1135	Firearms: assault weapons
AB1405	Developmental centers: closure
AB66	Peace officers: body-worn cameras
AB884	Legislature: legislative proceeding: audiovisual recordings
SB10	Health care coverage: immigration status
SB1235	Ammunition
SB128	End of life
SB277	Public health: vaccinations
SB329	Charter schools: petition denials: competitive bidding
SB350	Clean Energy and Pollution Reduction Act of 2015
SCA14	Legislative procedure

Table 5.8: *Bills for pull quote generation*

groupings to select a subset of ten sentences as pull quotes. Both methods ignore clusters that do not result in any extracted sentences.

The first method *TopScores* selects based on the path score from the graph method and the similarity score from sentence extraction. Sentences chosen by WHISK come from discovering redundant sequences using a graph method and finding a matching sentence from the original text using sentence extraction. We describe the graph methods' scoring in Chapter 3 and the sentence extraction's similarity score in Chapter 4. The core metrics from the graph method and the sentence extraction combine to represent the final score of a sentence in the following equation: $sentencescore = pathscore * pathsimilarity$. All sentences across clusters are sorted by the sentence score metric. The sentences with the top ten scores are chosen as the pull quotes.

The second method *RoundRobin* selects quotes from each cluster in a round robin

Id	Bill Id	Committee	Date	# Utt.
1	AB1135	Public Safety (Senate)	05-10-2016	58
2	AB1405	Human Services (Assembly)	06-09-2015	325
3	AB66	Privacy and Consumer Protection (Assembly)	04-30-2015	308
4	AB884	Elections and Constitutional Amendments (Senate)	06-08-2016	145
5	AB884	Appropriations (Senate)	06-13-2016	48
6	SB10	Health (Assembly)	04-26-2016	157
7	SB1235	Public Safety (Senate)	04-19-2016	66
8	SB1235	Senate Floor	05-19-2016	117
9	SB128	Judiciary (Senate)	04-07-2015	515
10	SB128	Health (Senate)	03-25-2015	478
11	SB277	Education (Senate)	04-15-2015	1276
12	SB277	Health (Assembly)	06-09-2015	1354
13	SB329	Education (Senate)	04-22-2015	162
14	SB350	Energy, Utilities and Communications (Senate)	04-07-2015	355
15	SB350	Natural Resources (Assembly)	07-13-2015	418
16	SCA14	Elections and Constitutional Amendments (Senate)	06-08-2016	213
17	SCA14	Appropriations (Senate)	06-13-2016	195

Table 5.9: ***Bill discussions for pull quote generation***

Id	# Clusters	# Usable Clusters	Max # Utt.	Min # Utt.	Avg # Utt.	Avg # Utt. in Usable Clusters
1	2	1	50	3	26.5	50
2	8	1	277	3	38.125	277
3	5	2	289	3	60.8	147
4	2	1	137	3	70	137
5	2	1	45	3	24	45
6	4	1	108	3	30.25	108
7	1	1	62	62	62	62
8	5	1	102	3	22.8	102
9	1	1	515	515	515	515
10	4	1	425	3	109.25	425
11	24	3	1082	3	48.7083	365.6667
12	15	4	1226	3	87.1333	310.25
13	5	1	115	3	27	115
14	6	3	311	4	56.1666	108
15	3	1	380	9	133	380
16	4	1	197	3	51.75	197
17	6	2	167	3	31.3333	85.5
Average	5.7058	1.5294	322.8235	37	81.9892	201.7303

Table 5.10: ***Cluster statistics for bill discussions***

Id	Max # graph nodes	Min # graph nodes	Avg # graph nodes
1	369	3	186
2	1008	2	136
3	1203	1	253.8
4	771	3	387
5	359	4	181.5
6	388	1	99.75
7	359	359	359
8	705	2	149.2
9	1819	1819	1819
10	2033	1	510
11	2856	1	128.8333
12	3294	3	239.4
13	636	11	141.4
14	1217	3	212.8333
15	1793	16	608.3333
16	1020	2	259.5
17	708	4	125.8333
Average	1208.1176	131.4705	341.0225

Table 5.11: ***Cluster graph statistics for bill discussions for all clusters***

Id	Max # graph nodes	Min # graph nodes	Avg # graph nodes
1	369	369	369
2	1008	1008	1008
3	1203	11	607
4	771	771	771
5	359	359	359
6	388	388	388
7	359	359	359
8	705	705	705
9	1819	1819	1819
10	2033	2033	2033
11	2856	9	966.3333
12	3294	40	860.5
13	636	636	636
14	1217	13	416.3333
15	1793	1793	1793
16	1020	1020	1020
17	708	19	363.5
Average	1208.1176	667.7647	851.3922

Table 5.12: ***Cluster graph statistics for bill discussion for usable clusters***

Method Id	Graph Method	Selection Method
A	Opinosis	RoundRobin
B	Opinosis	TopScores
C	sentence-id	RoundRobin
D	sentence-id	TopScores
E	numeric	RoundRobin
F	numeric	TopScores

Table 5.13: *Quote generation methods*

fashion. Clusters are ordered by number of words in their texts. We iterate over the clusters by picking one representative sentence from a cluster each time. The selection of a sentence in a cluster uses the same method as *TopScores*. We propose this method to prevent bigger cluster which may have higher redundancy from entirely overshadowing smaller clusters.

By combining the three graph methods with the two proposed pull quote selection methods, we have six different quote generation methods listed in Table 5.13.

Some quotes are chosen by more than one of the six quote generation methods. Figures 5.4 and 5.5 use Venn diagrams to portray the number of shared quotes from the different quote generation methods for bill discussion *SB277 Hearing Id#261*. Figure 5.4 shows a 2-set Venn diagram for each of the graph models. Each of the three 2-set Venn diagrams depicts the shared quotes between using RoundRobin versus TopScores selection methods while using the same graph model. Figure 5.5 uses a 3-set Venn diagram to show shared quotes between the three graph models. The 3-set Venn diagram groups quote generation methods by the graph method to ignore the quote selection method. This reduces the six quote generation into the three sets in the Venn diagram: "AB," "CD," and "EF." The combined sets are unions of the quotes generated by the original methods. For example, "AB" contains quotes

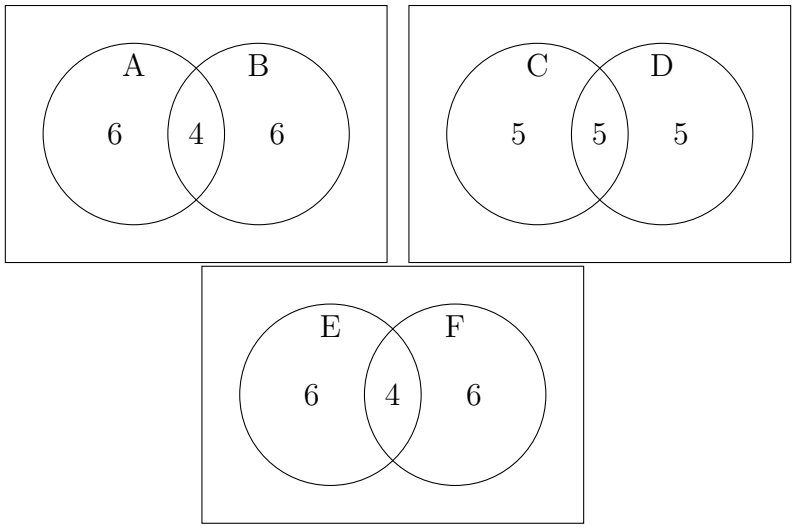

Figure 5.4: ***Quote selection method comparison for SB277 Hearing Id#261.*** **Venn diagram for showing the number of shared pull quotes selected from SB277 Hearing Id#261 by different quote selection methods grouped by the graph model used**

generated with method "A" or method "B." The Venn diagrams for the other bill discussions can be found in Figures B.2 to B.33 in Appendix B.

5.3.2 Evaluation Method

To evaluate selected pull quotes, we create surveys for each hearing that ask human evaluators to rate how appropriate each pull quote is for representing the ideas in the hearing. Each pull quote is presented as a multiple choice question. The speaker's name is shown with the sentence to establish context. The evaluators may choose from the following choices: "Don't know," "Bad," "Questionable," "Maybe," and "Good." The evaluators also have a checkbox option to indicate whether the quote is the best quote they saw for this bill discussion. Figure B.1 in Appendix B shows what a survey looks like. Additional feedback may be provided textually by the evaluator at the end of the survey. All responses are kept anonymous and no personal information is recorded.

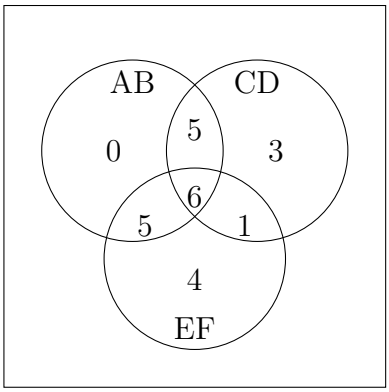

Figure 5.5: ***Graph model comparison for SB277 Hearing Id#261.*** **Venn diagram for showing the number of shared pull quotes generated from SB277 Hearing Id#261 using different graph models and ignoring the quote selection method**

Certain evaluators may prefer specific processing methods over others. We have six total methods of generating pull quotes by combining the three different graph methods and the two proposed pull quote selection methods. To prevent any bias about processing methods, the surveys show all pull quotes on a single page and do not reveal which method(s) generated which quotes.

Response Analysis. We analyze the precision of WHISK on producing good pull quotes using a number of measures with the survey responses. The survey responses provide insight on how relevant the pull quote is to the committee hearing it originated from. Ignoring "Don't Know" responses, the individual multiple choice responses are interpreted as relevant or irrelevant ratings based on a soft and a strict definition. The soft definition of relevant allows the response to be "Maybe" or "Good" while the strict definition requires the response to be "Good."

We then determine the overall rating of a quote based on the ratings. Three different quote rating methods provide different viewpoints on how relevant the pull quote is:

- Plurality takes the plurality rating (or majority vote) from the responses.

  In the case of a tie, we take preference towards relevant since losing quotes marked as relevant by some experts would be more detrimental than providing quotes some experts do not value.

- Consensus requires a consensus among the evaluators where a quote is only considered relevant if all responses rate the quote as relevant.

- Union takes the opposite side of Consensus where we consider a quote to be relevant if at least one response rates the quote as relevant.

The two definitions of relevance combined with the three quote rating methods create six different quote evaluation methods shown as methods #1 to #6 in Table 5.14.

After determining the overall rating for quotes in a bill discussion, we aggregate the rating as a precision metric for each quote generation method. We calculate the percentage of quotes that were considered relevant in the bill discussion. Additionally, we consider a bill discussion to be "consensus covered" if at least one quote achieves a consensus rating that it is relevant. Combining consensus coverage with the two definitions of relevance, we add two more methods (#7 and #8) in Table 5.14.

To calculate the overall score for each quote generation method, the percentage of relevant quotes is averaged across bill discussions. To aggregate consensus coverage, we calculate the percentage of bill discussions that are consensus covered for each quote generation method.

We also record the number of quotes that receive at least one "Best Quote" vote. Each quote generation method's count of best quotes is then summed across bill discussions to compare the number of best quotes achieved. Since the same quote can be selected by more than one method, we also create a 3-set Venn diagram similar to Figure 5.5 to display the number of shared best quotes generated with the three

Id	Relevance	Overall Rating	Explanation
1	Soft	Plurality	Majority of experts rated the quote as "Maybe" or "Good"
2	Strict	Plurality	Majority of experts rated the quote as "Good"
3	Soft	Consensus	All experts rated the quote as "Maybe" or "Good"
4	Strict	Consensus	All experts rated the quote as "Good"
5	Soft	Union	At least one expert rated the quote as "Maybe" or "Good"
6	Strict	Union	At least one expert rated the quote as "Good"
7	Soft	Consensus Coverage	At least one quote in the bill discussion was rated relevant under method #3
8	Strict	Consensus Coverage	At least one quote in the bill discussion was rated relevant under method #4

Table 5.14: ***Evaluation methods***

graph methods.

5.3.3 Participants

Seven experts familiar with committee hearings were asked to evaluate pull quotes through a list of surveys. Some experts were familiar with only a subset of the bills and filled out the appropriate surveys for those bills. As such, surveys may have a different number of responses as shown in Table 5.15.

5.3.4 Results

Results for 3 out of the 17 bill discussions (SB128 Hearing Id#123, SB277 Hearing Id#308, SCA14 Hearing Id#1266) are shown as data tables in Appendix B. Tables for the remaining 14 bill discussions are not shown for the sake of brevity. Tables B.1, B.3 and B.5 list the pull quotes generated for each bill discussion. Tables B.2, B.4 and B.6 show the quote generation method source and the rating counts from the survey for each generated pull quote in the bill discussions. We apply our evaluation methods to this data to compare our quote generation methods.

The number of best quotes for each quote generation method are shown in Table 5.16. The numeric graph achieved the highest number of best quotes as quote generation methods E and F followed by the Opinosis graph (methods A and B) and the sentence-id graph (methods C and D). The number of shared best quotes are shown using a 3-set Venn diagram in Figure 5.6.

Applying the evaluation methods in Table 5.14, we compare the aggregate results in Table 5.17. For all evaluation methods, higher values are better. The best scores of each evaluation method in Table 5.17 are highlighted in blue boxes. Quote generation method D (sentence-id graph using *TopScores*) scored the most number (4) of best scores followed by methods B (Opinosis graph using *TopScores*), C (sentence-id

Bill Discussion Id	# Responses
1	7
2	6
3	4
4	6
5	6
6	5
7	5
8	5
9	5
10	4
11	7
12	5
13	4
14	4
15	6
16	5
17	5

Table 5.15: ***Number of responses for the survey for each bill discussion***

Id	Method Source						Total # Best Quotes
	A	B	C	D	E	F	
1	1	1	1	1	1	1	1
2	1	0	1	0	2	2	3
3	1	1	1	1	2	2	2
4	2	2	3	3	2	2	3
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1
7	2	2	2	2	1	1	2
8	1	1	0	0	2	2	3
9	2	2	2	1	3	2	4
10	3	1	3	1	2	2	3
11	1	2	2	2	1	2	3
12	1	0	0	1	1	1	2
13	1	1	1	1	1	1	1
14	1	1	1	1	1	1	3
15	0	0	0	1	0	1	1
16	1	1	0	0	0	0	1
17	2	2	2	1	2	2	2
Sum	22	19	21	18	23	24	36

Table 5.16: *Number of best quotes.* The number of best quotes in a bill discussion achieved by each quote generation method along with the total number of best quotes in the bill discussion

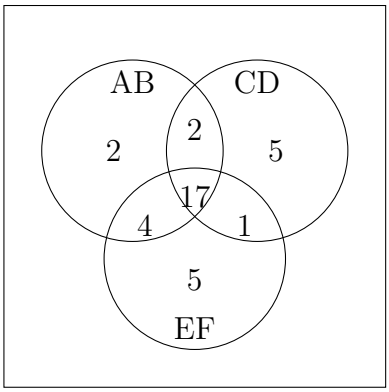

Figure 5.6: *Graph model comparison for best quotes.* Venn diagram for showing the number of shared best quotes using different graph models and ignoring the quote selection method

graph using *RoundRobin*) and F (numeric graph using *TopScores*) with 2 best scores. Method C and D tied for evaluation methods #7 and #8. In all of evaluation methods, quote generation methods (B,D,F) using the *TopScores* selection method performed comparably to, if not better than, their counterparts (A,C,E) using the *RoundRobin* selection method. Overall, the sentence-id graph (methods C and D) and the numeric graph (methods E and F) follow closely and sometimes beat the Opinosis graph (methods A and B). Interestingly, Table 5.17 shows a consistent pattern where the Opinosis graph is best for plurality rating, the sentence-id graph is best for consensus rating, and the numeric graph is best for union rating.

The aggregates in Table 5.17 also show the overall performance of WHISK in selecting portions of political discussions as summaries. Under evaluation method #1 using soft Plurality scoring, 55-64% of pull quotes generated by WHISK would be rated as relevant by a majority. On the other hand, evaluation method #5 using soft Union scoring shows that 83-93% of pull quotes generated by WHISK would be rated as relevant by at least one person. The result analyses support that WHISK has potential as module for the Digital Democracy project that generates pull quotes for political bill discussion.

Evaluation method	Quote generation method					
	A	B	C	D	E	F
1	59.8039	64.5098	55.3922	61.8627	60.2941	63.2353
2	28.6275	29.8039	26.7157	29.0686	27.0588	28.8235
3	10.3922	10.3922	9.1667	12.6961	10.2941	11.4706
4	3.5294	3.5294	3.1373	3.7255	3.5294	3.5294
5	85.0000	89.1176	83.0392	87.7451	88.8235	93.5294
6	64.3137	67.2549	63.1863	65.5392	66.9118	68.0882
7	47.0588	47.0588	52.9412	52.9412	41.1765	41.1765
8	11.7647	11.7647	17.6471	17.6471	11.7647	11.7647

Table 5.17: ***Aggregate evaluation results***

Chapter 6

CONCLUSION

The amount of online content grows at such a high rate that users are overwhelmed and have difficulties finding interesting and relevant material without using search queries. Summaries or synopses can help alleviate this problem by significantly reducing the amount of text that users have to digest. Automatic text summarization serves as a potential solution towards generating these time-saving summaries. This thesis presented WHISK as an automatic text summarization pipeline based on SPORK [32] that exploits redundancy in texts to perform multi-document summarization.

WHISK generalizes from SPORK by using a generic data model to allow working with multiple platforms or plaintext in general. Client drivers decide what is considered to be a corpus or text unit. All special domain knowledge are handled and provided by client drivers rather than by the generic pipeline. The pipeline itself can be extended to provide additional processing modules between or replacing stages in the pipeline.

Within this thesis, we introduced two alternate graph models, sentence-id and numeric, over the Opinosis graph [19] adapted in SPORK [32]. The new graphs simplify from the Opinosis by removing word positional information for sentence-id graphs and further removing sentence occurrence information for numeric graphs. Since we only need to extract redundant sequences from the graphs, unnecessary grammar and positional restrictions are removed to provide faster performance.

We performed three different experiments comparing the two new graphs against the Opinosis graph. The first performance comparison showed that the sentence-id graph performs around 14% faster on graph creation and 29% faster on graph traver-

sal compared to Opinosis. The numeric graph performs graph creation around 14% faster than Opinosis and graph traversal around 75% faster. Both graphs produced summaries that had Jaccard similarity values of around 0.5 on average to the summaries generated with the Opinosis graph.

ROUGE [27], an automatic summary evaluation tool, provides the second experiment which compares the graph models using gold-standard summaries from Ganesan et al.'s dataset [19]. We used ROUGE-1 and ROUGE-2 which scores based on unigrams and bigrams respectively. Although Opinosis achieves the higher F-scores in both ROUGE-1 and ROUGE-2 (0.1584 and 0.0295 respectively), sentence-id has the higher recall scores. Neither of the two new graphs fall too far behind Opinosis.

In the third experiment, in collaboration with the Digital Democracy project, WHISK generated pull quotes for a number of bill discussions in the California state legislature. We created a client driver for WHISK to work with the transcription data from the Digital Democracy project. The driver generates representative pull quotes that summarize the ideas presented by various speakers in the bill discussions. Seven domain experts rated the generated pull quotes for each bill discussion. Out of the 8 evaluation methods, sentence-id graph achieved 4 of the best scores while numeric graph and Opinosis graph achieved 2 of the best scores. Both sentence-id and numeric graphs performed comparably to the Opinosis graph. Across all quote generation methods, 83-93% of pull quotes generated by WHISK on average were considered relevant under a soft definition by at least one expert.

Overall, the sentence-id and numeric graphs proposed in this thesis produces similar or better summaries while improving the graph processing speed. Additionally, WHISK has demonstrated its potential in generating relevant pull quotes from political discussions.

6.1 Future Work

WHISK has a strong dependence on redundancy in the corpus text due to its underlying graph models. This works fine for multi-document summarization of discussions where we can reasonably expect people to say similar things. Techniques from single document summarization can enable WHISK to handle discussions where people do not use the same words or phrases.

The clustering of corpus texts can be further refined from DBSCAN [15]. Although DBSCAN offers a fast clustering, sample runs show that many texts may be labelled as noise and put into a noise cluster. The noise cluster is not grouped by similar texts, so it can suffer from too many contesting topics. If the texts can be better grouped without too much more processing time, WHISK has a lower chance of missing a smaller topic overshadowed by other topics.

The handling of query during graph traversal described in Chapter 4 can be further optimized. We currently use the simple filtering method which would still explore all candidate paths. However, graph traversal starting directly at the query word's node can potentially save more processing time. This can be done using forward and backward propagation as done by SPORK [32].

Graph traversal can also be optimized with a parallel implementation for large graphs. WHISK currently performs graph traversal in a serial manner. However, there are not any data dependencies between path discovery runs. The overall top paths can be later determined when all discoverd paths are merged back into a single pool.

Parallelization can also be done on each layer to make WHISK into a streaming pipeline. The pipeline would then be working on multiple corpora at the same time. WHISK currently processes corpora one at a time, so there is definitely room for

better scalability and throughput.

Additional changes to the driver for the Digital Democracy project can be tested to improve the quality of generated pull quotes. Rather than extracting sentences as the pull quote, it may be possible to extract entire utterances since each utterance should represent an uninterrupted thought. Extraction of multiple utterances by the same speaker may also be useful in the case of a short interruption. A restriction on the number of words may also lead to better extraction since long pull quotes can lose the reader's attention.

BIBLIOGRAPHY

[1] Cal Poly Github. `http://www.github.com/CalPoly`.

[2] S. Banerjee, P. Mitra, and K. Sugiyama. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1208–1214. AAAI Press, 2015.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] M. Campana and A. Tombros. Incremental personalised summarisation with novelty detection. In *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, FQAS '09, pages 641–652, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA, 1998. ACM.

[6] A. Celikyilmaz and D. Hakkani-Tür. Discovery of topically coherent sentences for extractive summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 491–499, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[7] Y. Chang, X. Wang, Q. Mei, and Y. Liu. Towards twitter context

summarization with user influence models. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 527–536, New York, NY, USA, 2013. ACM.

[8] K. W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, ANLC '88, pages 136–143, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics.

[9] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. Mbt: A memory-based part of speech tagger-generator. *arXiv preprint cmp-lg/9607012*, 1996.

[10] D. Das and A. F. Martins. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195, 2007.

[11] S. J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, 1988.

[12] A. Díaz and P. Gervás. User-model based personalized summarization. *Information Processing & Management*, 43(6):1715–1734, 2007.

[13] A. Díaz, P. Gervás, and A. García. Evaluation of a system for personalized summarization of web contents. In *Proceedings of the 10th International Conference on User Modeling*, UM'05, pages 453–462, Berlin, Heidelberg, 2005. Springer-Verlag.

[14] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479, Dec. 2004.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[16] R. Ferreira, F. Freitas, L. d. S. Cabral, R. D. Lins, R. Lima, G. França, S. J. Simskez, and L. Favaro. A four dimension graph model for automatic text summarization. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01*, WI-IAT '13, pages 389–396, Washington, DC, USA, 2013. IEEE Computer Society.

[17] K. Filippova. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 322–330, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[18] P. Fung, G. Ngai, and C.-S. Cheung. Combining optimal clustering and hidden markov models for extractive summarization. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering - Volume 12*, MultiSumQA '03, pages 21–28, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[19] K. Ganesan, C. Zhai, and J. Han. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[20] R. Garside. The CLAWS word-tagging system. In R. Garside, G. Leech, and

G. Sampson, editors, *The Computational Analysis of English: a corpus-based approach*, pages 30–41. Longman, London, 1987.

[21] M. Hu, A. Sun, and E.-P. Lim. Comments-oriented blog summarization by sentence extraction. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 901–904, New York, NY, USA, 2007. ACM.

[22] M. Hu, A. Sun, and E.-P. Lim. Comments-oriented document summarization: Understanding documents with readers' feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 291–298, New York, NY, USA, 2008. ACM.

[23] P. Jaccard. The distribution of the flora in the alpine zone. *New phytologist*, 11(2):37–50, 1912.

[24] M. Khan, D. Bollegala, G. Liu, and K. Sezaki. Multi-tweet summarization of real-time events. In *Social Computing (SocialCom), 2013 International Conference on*, pages 128–133, Sept 2013.

[25] T.-Y. Kim, J. Kim, J. Lee, and J.-H. Lee. A tweet summarization method based on a keyword graph. In *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '14, pages 96:1–96:8, New York, NY, USA, 2014. ACM.

[26] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[27] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In S. S. Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings*

of the ACL-04 Workshop, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[28] F. Liu and Y. Liu. From extractive to abstractive meeting summaries: Can it be done by sentence compression? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 261–264, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[29] E. Loper and S. Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[30] J. B. Lovins. *Development of a stemming algorithm.* MIT Information Processing Group, Electronic Systems Laboratory Cambridge, 1968.

[31] Y. Lu, C. Zhai, and N. Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 131–140, New York, NY, USA, 2009. ACM.

[32] S. Lyngbaek. Spork: A summarization pipeline for online repositories of knowledge. Master's thesis, California Polytechnic State University San Luis Obispo, June 2013.

[33] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. In Russell and Haller [48], pages 392–396.

[34] K. McKeown and D. R. Radev. Generating summaries of multiple news articles. In *Proceedings of the 18th Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval, SIGIR '95, pages 74–82, New York, NY, USA, 1995. ACM.

[35] B. Merialdo. Tagging english text with a probabilistic model. *Comput. Linguist.*, 20(2):155–171, June 1994.

[36] Z. Y. Ming, J. Ye, and T. S. Chua. A dynamic reconstruction approach to topic summarization of user-generated-content. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 311–320, New York, NY, USA, 2014. ACM.

[37] R. Móro and M. Bieliková. Personalized text summarization based on important terms identification. In *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, pages 131–135. IEEE, 2012.

[38] G. K. R. Naveen and P. Nedungadi. Query-based multi-document summarization by clustering of documents. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, ICONIAAC '14, pages 58:1–58:8, New York, NY, USA, 2014. ACM.

[39] S. Park and B. Cha. Query-based multi-document summarization using non-negative semantic feature and nmf clustering. In *Fourth International Conference on Networked Computing and Advanced Information Management, 2008. NCM '08.*, volume 2, pages 609–614, Sept. 2008.

[40] K. Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn:

Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] M. H. Quenouille. Notes on bias in estimation. *Biometrika*, 43(3/4):353–360, 1956.

[43] D. R. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408, Dec. 2002.

[44] A. Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, USA, 1996.

[45] Z. Ren, S. Liang, E. Meij, and M. de Rijke. Personalized time-aware tweets summarization. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 513–522, New York, NY, USA, 2013. ACM.

[46] J. Rovin. Reducing costs in human assisted speech transcription. Master's thesis, California Polytechnic State University San Luis Obispo, Mar. 2016.

[47] K. Rudra, S. Ghosh, N. Ganguly, P. Goyal, and S. Ghosh. Extracting situational information from microblogs during disaster events: A classification-summarization approach. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 583–592, New York, NY, USA, 2015. ACM.

[48] I. Russell and S. M. Haller, editors. *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference, May 12-14, 2003, St. Augustine, Florida, USA*. AAAI Press, 2003.

[49] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[50] H. Schmid. Part-of-speech tagging with neural networks. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, pages 172–176, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[51] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the international conference on new methods in language processing*, volume 12, pages 44–49. Citeseer, 1994.

[52] B. Sharifi, M.-A. Hutton, and J. Kalita. Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 685–688, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[53] L. Shou, Z. Wang, K. Chen, and G. Chen. Sumblr: Continuous summarization of evolving tweet streams. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 533–542, New York, NY, USA, 2013. ACM.

[54] G. Silva, R. Ferreira, R. D. Lins, L. Cabral, H. Oliveira, S. J. Simske, and M. Riss. Automatic text document summarization based on machine learning. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, DocEng '15, pages 191–194, New York, NY, USA, 2015. ACM.

[55] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[56] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[57] G. Thomas, M. Zahm, and D. Furcy. Using a sentence compression pipeline for the summarization of email threads in an archive. *J. Comput. Sci. Coll.*, 31(2):72–78, Dec. 2015.

[58] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[59] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[60] V. Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

[61] Wikimedia Foundation, Inc. Wikipedia:statistics. http://en.wikipedia.org/wiki/Wikipedia:Statistics, Oct. 2015.

[62] K.-F. Wong, M. Wu, and W. Li. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 985–992, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[63] C. Wu. Skewer: Sentiment knowledge extraction with entity recognition. Master's thesis, California Polytechnic State University San Luis Obispo, June 2016.

[64] R. Yan, J.-Y. Nie, and X. Li. Summarize what you are interested in: An optimization framework for interactive personalized summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1342–1351. Association for Computational Linguistics, 2011.

[65] S. Yan and X. Wan. Deep dependency substructure-based learning for multidocument summarization. *ACM Trans. Inf. Syst.*, 34(1):3:1–3:24, July 2015.

[66] X. Ye and H. Wei. Query-based summarization for search lists. In *First International Workshop on Knowledge Discovery and Data Mining, 2008. WKDD 2008.*, pages 330–333, Jan. 2008.

[67] H. Zhang, Z. C. W.-y. Ma, and Q. Cai. A study for documents summarization based on personal annotation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, HLT-NAACL-DUC '03, pages 41–48, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[68] L. Zhu, S. Gao, S. J. Pan, H. Li, D. Deng, and C. Shahabi. Graph-based informative-sentence selection for opinion summarization. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 408–412, New York, NY, USA, 2013. ACM.

APPENDICES

Appendix A

GRAPH PERFORMANCE DATA

This appendix contains datasets of performance timings and Jaccard similarity values for comparing the new graph variants to the Opinosis graph.

Dataset	CId	# Words	Procedure	PRI (ms)	SentId (ms)	Numeric (ms)
SB145	1	6552	Creation	68.1	74.07	64.36
			Traversal	389.02	275.81	70.53
	2	299	Creation	2.64	1.95	1.95
			Traversal	9.16	11.98	3.59
SB277	1	45504	Creation	381.83	389.5	390.72
			Traversal	70064.53	22378.12	1940.99
	2	1106	Creation	13.41	10.48	14.79
			Traversal	32.92	28.85	8.54
	3	273	Creation	2.67	2.62	2.51
			Traversal	6.01	3.41	2.05
	4	236	Creation	4.23	2.07	1.92
			Traversal	2.38	0.83	0.64
	5	226	Creation	1.64	1.69	1.55
			Traversal	2.85	0.61	0.48
SB34	1	8433	Creation	144.61	88.35	73.26
			Traversal	654.37	284.28	70.78
	2	675	Creation	8.69	6.26	4.53
			Traversal	4.24	1.52	1.42
	3	650	Creation	4.42	4.23	4.31
			Traversal	86.73	178.03	56.57
	4	354	Creation	3.7	2.85	2.95
			Traversal	40.89	34.61	15.13
SB530	1	2912	Creation	29.87	25.99	33.71
			Traversal	114.89	115.55	37.81
	2	232	Creation	2.03	1.69	1.82

			Traversal	3.95	4.11	0.71
SB9	1	36568	Creation	416.53	413.03	424.08
			Traversal	16211.79	2386.54	206.85
	2	2294	Creation	22.23	19.89	19.05
			Traversal	52.49	20.88	6.18

Table A.1: *Comparison of timings for graph creation and traversal*

Dataset	Cluster Id	Num Sentences	Jaccard Similarity
SB145	1	1	0
		2	0
		3	0.2
		4	0.1428
		5	0.1111
		6	0.2
		7	0.1666
		8	0.2307
		9	0.2857
		10	0.3333
		11	0.375
		12	0.3333
		13	0.3684
		14	0.3333
		15	0.3636
		16	0.3333
		17	0.3076
		18	0.3461
	2	1	1
		2	1
		3	1
SB277	1	1	0
		2	0.3333
		3	0.5
		4	0.6

		5	0.4285
		6	0.3333
		7	0.4
		8	0.3333
		9	0.3846
		10	0.3333
		11	0.375
		12	0.4117
		13	0.3684
		14	0.4
		15	0.3636
		16	0.3913
		17	0.4347
		18	0.4166
	2	1	1
		2	0.3333
		3	0.5
		4	0.6
		5	0.5
		6	0.6666
	3	1	1
		2	1
	4	1	1
SB34	1	1	0
		2	0.3333
		3	0.5

		4	0.6
		5	0.6666
		6	0.5
		7	0.4
		8	0.4545
		9	0.3846
		10	0.4285
		11	0.4666
		12	0.4117
		13	0.4444
		14	0.4736
		15	0.5
		16	0.5238
		17	0.4782
		18	0.5
		19	0.5416
	3	1	0
		2	0.3333
		3	0.5
		4	0.6
		5	0.6666
		6	0.7142
		7	0.5555
		8	0.6
		9	0.7
	4	1	1

		2	0.3333
		3	0.5
		4	0.4
SB530	1	1	1
		2	0.3333
		3	0.5
		4	0.6
		5	0.6666
		6	0.7142
		7	0.5555
		8	0.4545
		9	0.5
		10	0.4285
		11	0.4666
		12	0.4117
		13	0.4444
		14	0.421
		15	0.4736
	2	1	0
		2	0
		3	0.3333
SB9	1	1	1
		2	1
		3	0.5
		4	0.3333
		5	0.25

		6	0.2
		7	0.1666
		8	0.1428
		9	0.2
		10	0.1764
		11	0.2222
		12	0.2
		13	0.238
		14	0.2727
		15	0.3043
		16	0.28
		17	0.3076
		18	0.3333
		19	0.3571
	2	1	1
		2	1
		3	1
		4	1
		5	1
		6	1
		7	1
		8	1
		9	0.8888

Table A.2: *Jaccard similarity between Opinosis and* sentence-id

Dataset	Cluster Id	Num Sentences	Jaccard Similarity
SB145	1	1	0
		2	0.3333
		3	0.2
		4	0.1428
		5	0.1111
		6	0.2
		7	0.1666
		8	0.2307
		9	0.2
		10	0.25
		11	0.3125
		12	0.375
		13	0.4375
		14	0.4117
		15	0.4705
		16	0.4444
		17	0.421
		18	0.4
	2	1	1
		2	1
		3	1
SB277	1	1	1
		2	0.3333
		3	0.5
		4	0.6

		5	0.4285
		6	0.3333
		7	0.2727
		8	0.2307
		9	0.2
		10	0.1764
		11	0.1578
		12	0.2
		13	0.238
		14	0.2727
		15	0.25
		16	0.28
		17	0.3076
		18	0.3333
	2	1	0
		2	0.3333
		3	0.5
		4	0.6
		5	0.8
	3	1	1
		2	0.5
	4	1	1
SB34	1	1	0
		2	0.3333
		3	0.2
		4	0.3333

		5	0.4285
		6	0.3333
		7	0.2727
		8	0.2307
		9	0.2
		10	0.1764
		11	0.2222
		12	0.2631
		13	0.3
		14	0.2727
		15	0.3181
		16	0.3043
		17	0.2916
		18	0.3333
	3	1	1
		2	0.3333
		3	0.5
		4	0.6
		5	0.4285
		6	0.5
		7	0.625
		8	0.5555
		9	0.6666
	4	1	0
		2	0
		3	0.2

SB530	1	1	1
		2	0.3333
		3	0.5
		4	0.6
		5	0.6666
		6	0.7142
		7	0.75
		8	0.6
		9	0.6363
		10	0.5384
		11	0.6153
		12	0.6923
		13	0.7692
	2	1	0
		2	0
		3	0.3333
SB9	1	1	1
		2	0.3333
		3	0.5
		4	0.3333
		5	0.25
		6	0.3333
		7	0.2727
		8	0.2307
		9	0.2857
		10	0.25

		11	0.2222
		12	0.2631
		13	0.3
		14	0.2727
		15	0.3043
		16	0.3333
		17	0.36
		18	0.3846
		19	0.4074
		20	0.4444
	2	1	0
		2	0
		3	0
		4	0
		5	0
		6	0
		7	0.1111
		8	0.2222

Table A.3: *Jaccard similarity between Opinosis and* numeric

Appendix B

SURVEY EXAMPLE AND RESULTS

This appendix contains a subset of data from the validation study done in collaboration with the Digital Democracy project. Below is a example screenshot of the survey for bill AB1405 hearing id#302 which shows the format of the surveys used in validation. There are a total of 20 quotes to be evaluated in that survey, so the screenshot is cut off to remain brief.

Following the screenshot are venn diagrams that show how many shared quotes were generated by looking at the quote selection method and then at the graph model used for each bill discussion. The venn diagrams showing the number of shared quotes between the different quote selection methods are separated into three diagrams, one for each graph model. The venn diagrams showing the number of shared quotes between the different graph models ignore the quote selection method.

Lastly, tables of results are shown for three bill discussions:

- SB128 Hearing Id#123
- SB277 Hearing Id#308
- SCA14 Hearing Id#1266

The tables show the generated pull quotes, the source method(s) for each quote, the rating counts from the survey, and the average rating scores based on the numerical scale described in Section 5.3.

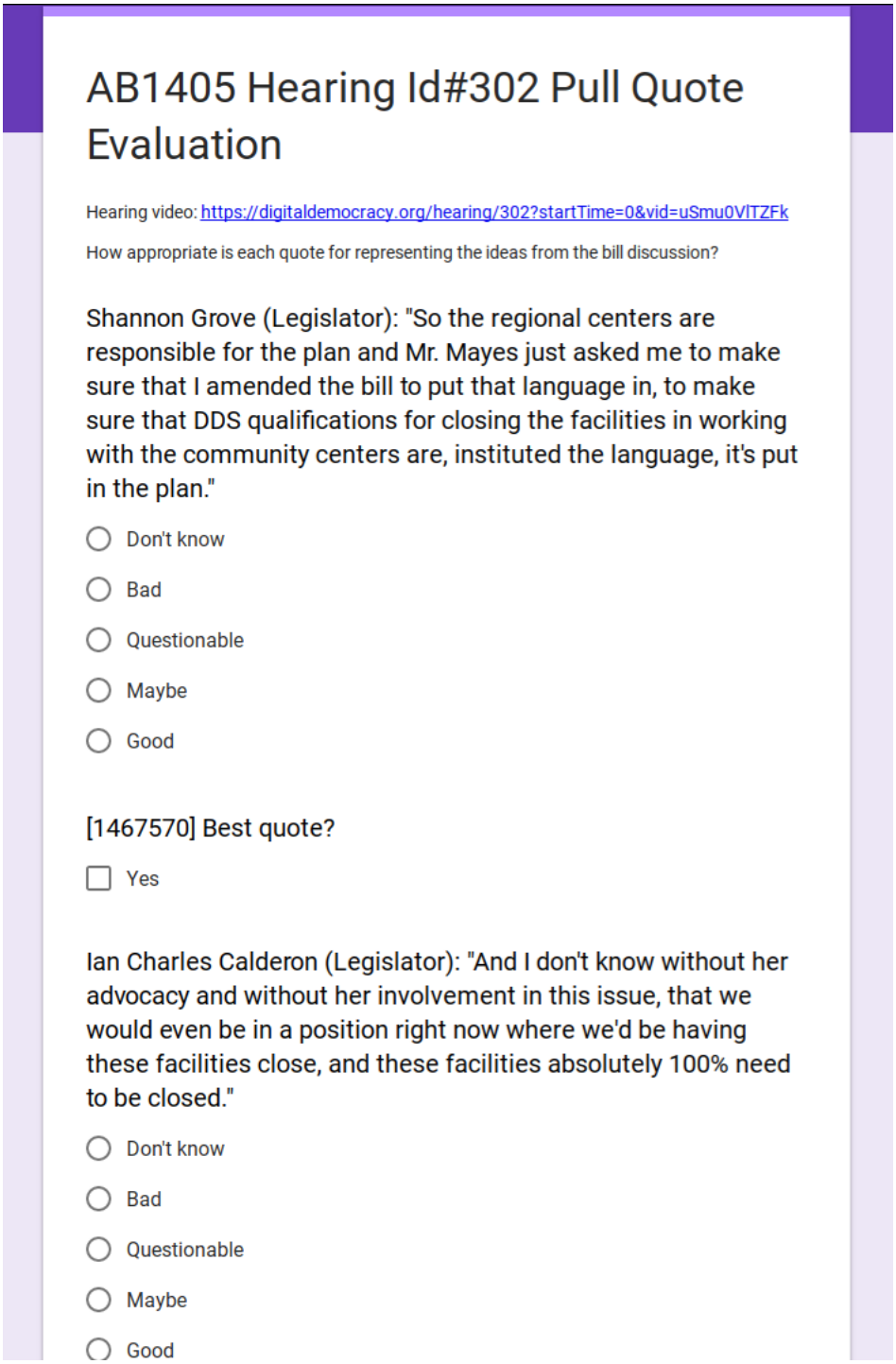

Figure B.1: ***Screenshot of the survey for AB1405 Hearing Id#302***

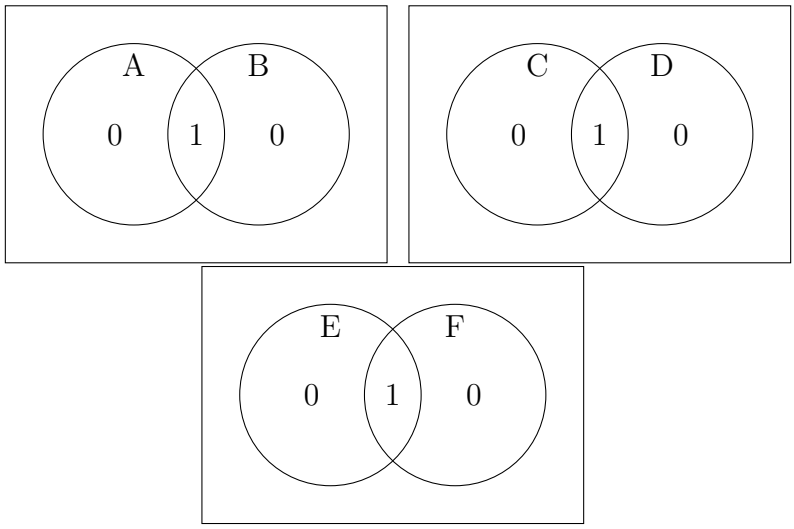

Figure B.2: ***Quote selection method comparison for AB1135 Hearing Id#1123.*** **Venn diagram for showing the number of shared pull quotes selected from AB1135 Hearing Id#1123 by different quote selection methods grouped by the graph model used**

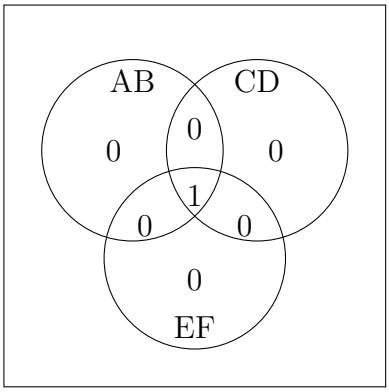

Figure B.3: ***Graph model comparison for AB1135 Hearing Id#1123.*** **Venn diagram for showing the number of shared pull quotes generated from AB1135 Hearing Id#1123 using different graph models and ignoring the quote selection method**

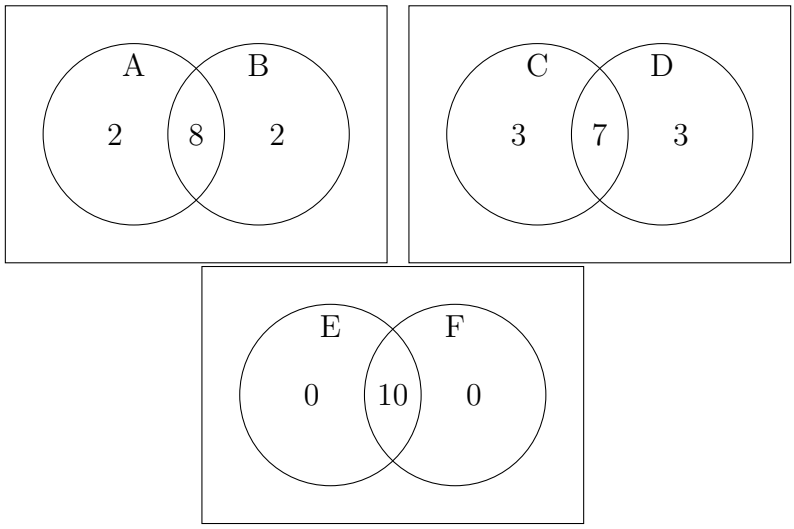

Figure B.4: ***Quote selection method comparison for AB1405 Hearing Id#302.*** **Venn diagram for showing the number of shared pull quotes selected from AB1405 Hearing Id#302 by different quote selection methods grouped by the graph model used**

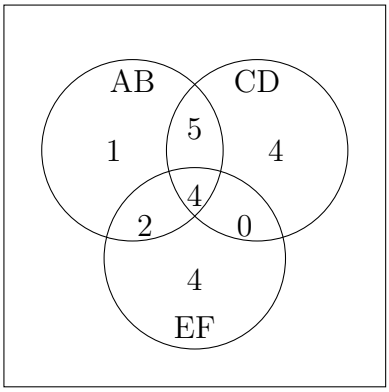

Figure B.5: ***Graph model comparison for AB1405 Hearing Id#302.*** **Venn diagram for showing the number of shared pull quotes generated from AB1405 Hearing Id#302 using different graph models and ignoring the quote selection method**

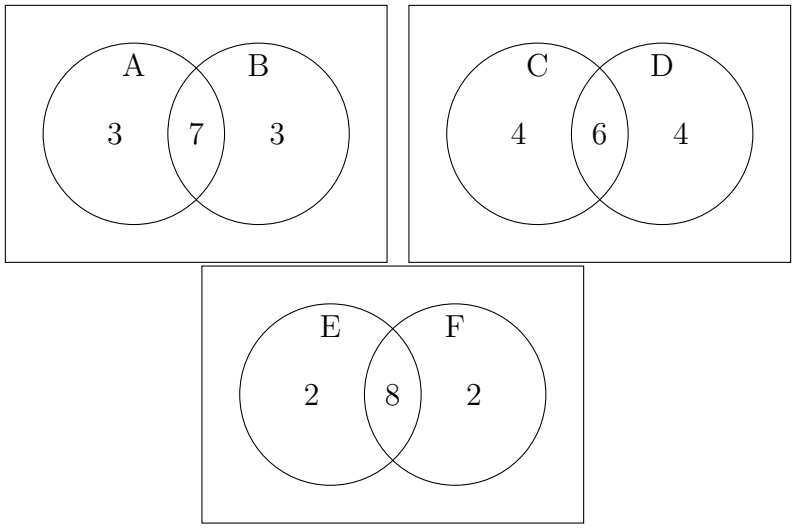

Figure B.6: ***Quote selection method comparison for AB66 Hearing Id#192.*** **Venn diagram for showing the number of shared pull quotes selected from AB66 Hearing Id#192 by different quote selection methods grouped by the graph model used**

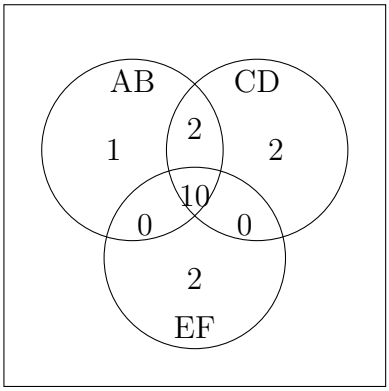

Figure B.7: ***Graph model comparison for AB66 Hearing Id#192.*** **Venn diagram for showing the number of shared pull quotes generated from AB66 Hearing Id#192 using different graph models and ignoring the quote selection method**

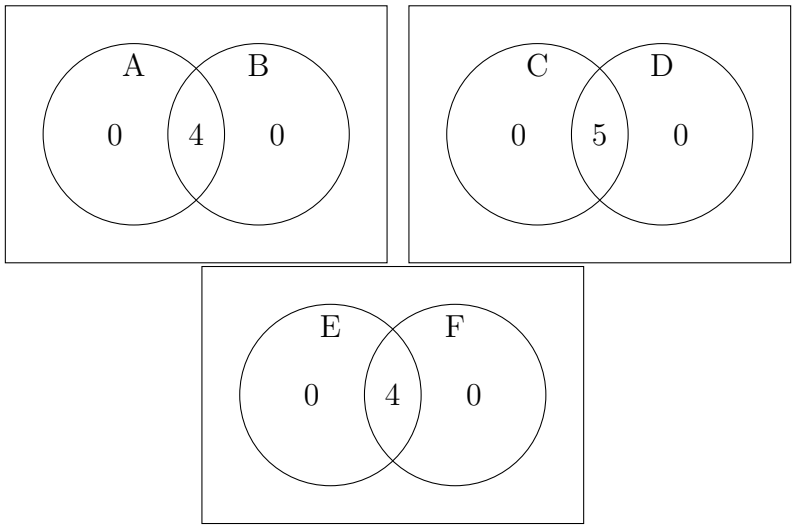

Figure B.8: ***Quote selection method comparison for AB884 Hearing Id#1266.*** **Venn diagram for showing the number of shared pull quotes selected from AB884 Hearing Id#1266 by different quote selection methods grouped by the graph model used**

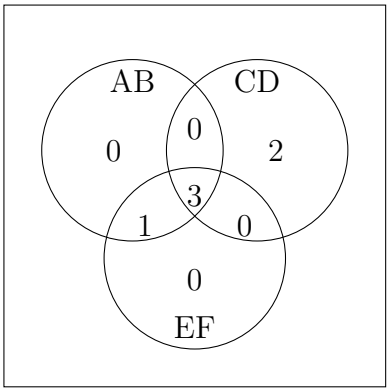

Figure B.9: ***Graph model comparison for AB884 Hearing Id#1266.*** **Venn diagram for showing the number of shared pull quotes generated from AB884 Hearing Id#1266 using different graph models and ignoring the quote selection method**

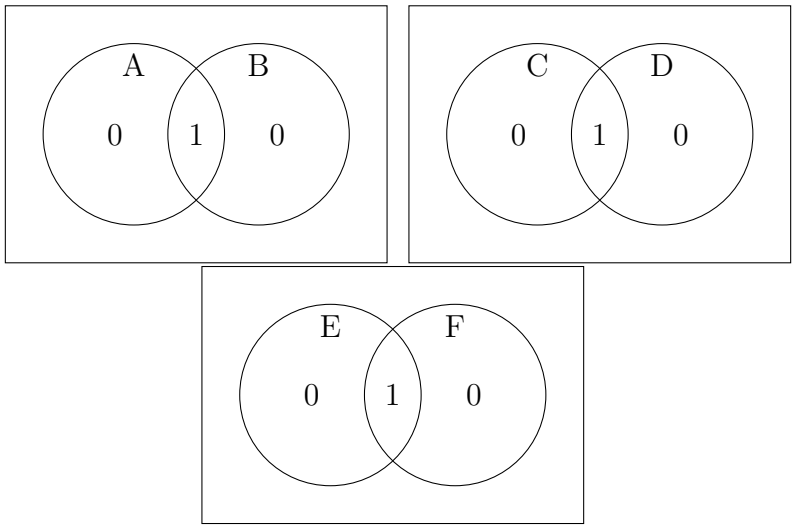

Figure B.10: ***Quote selection method comparison for AB884 Hearing Id#1284.*** **Venn diagram for showing the number of shared pull quotes selected from AB884 Hearing Id#1284 by different quote selection methods grouped by the graph model used**

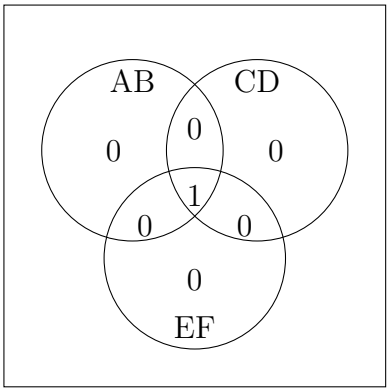

Figure B.11: ***Graph model comparison for AB884 Hearing Id#1284.*** **Venn diagram for showing the number of shared pull quotes generated from AB884 Hearing Id#1284 using different graph models and ignoring the quote selection method**

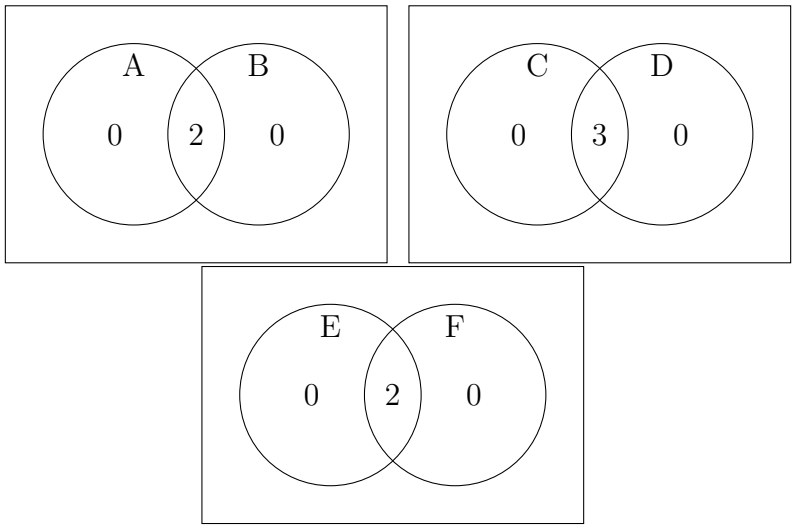

Figure B.12: ***Quote selection method comparison for SB10 Hearing Id#1048.*** **Venn diagram for showing the number of shared pull quotes selected from SB10 Hearing Id#1048 by different quote selection methods grouped by the graph model used**

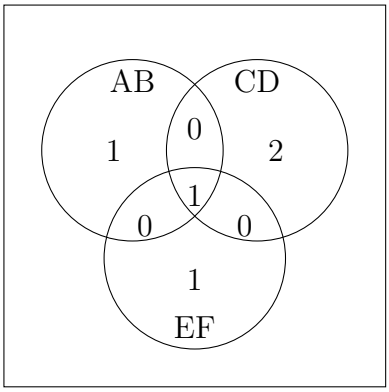

Figure B.13: ***Graph model comparison for SB10 Hearing Id#1048.*** **Venn diagram for showing the number of shared pull quotes generated from SB10 Hearing Id#1048 using different graph models and ignoring the quote selection method**

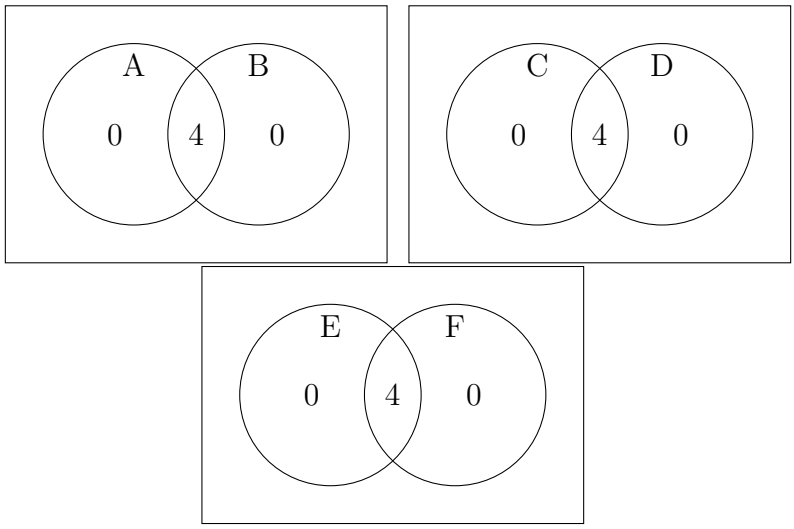

Figure B.14: ***Quote selection method comparison for SB1235 Hearing Id#1074.*** **Venn diagram for showing the number of shared pull quotes selected from SB1235 Hearing Id#1074 by different quote selection methods grouped by the graph model used**

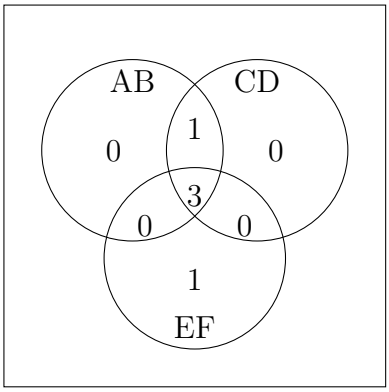

Figure B.15: ***Graph model comparison for SB1235 Hearing Id#1074.*** **Venn diagram for showing the number of shared pull quotes generated from SB1235 Hearing Id#1074 using different graph models and ignoring the quote selection method**

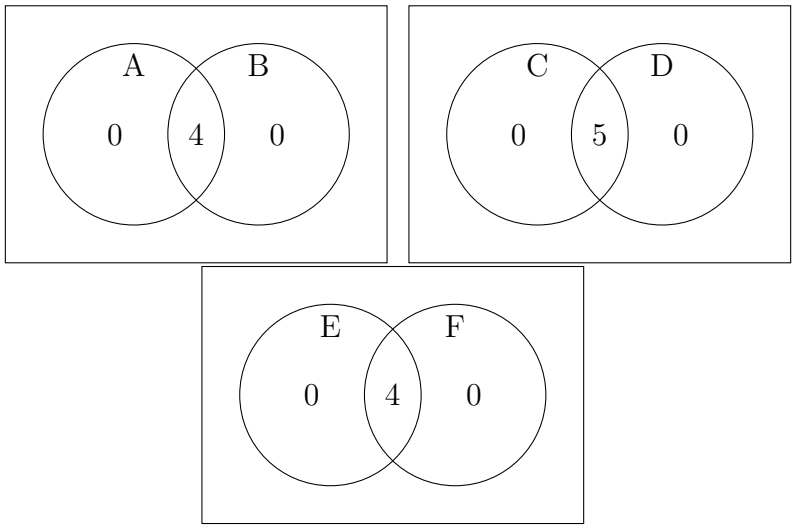

Figure B.16: ***Quote selection method comparison for SB1235 Hearing Id#1173.*** **Venn diagram for showing the number of shared pull quotes selected from SB1235 Hearing Id#1173 by different quote selection methods grouped by the graph model used**

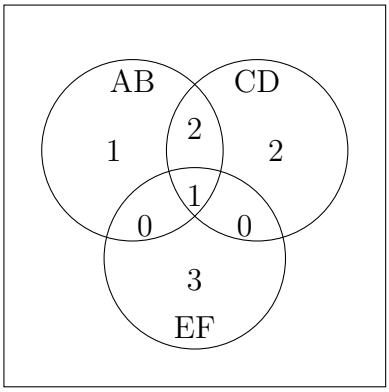

Figure B.17: ***Graph model comparison for SB1235 Hearing Id#1173.*** **Venn diagram for showing the number of shared pull quotes generated from SB1235 Hearing Id#1173 using different graph models and ignoring the quote selection method**

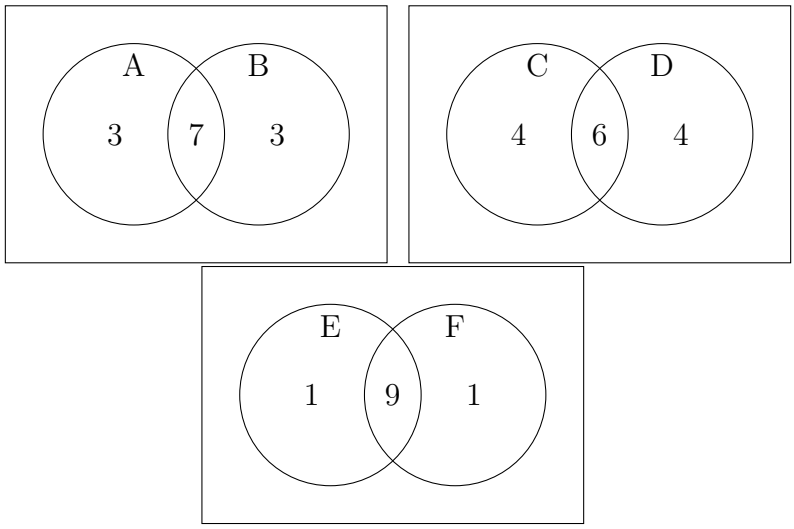

Figure B.18: ***Quote selection method comparison for SB128 Hearing Id#123.*** **Venn diagram for showing the number of shared pull quotes selected from SB128 Hearing Id#123 by different quote selection methods grouped by the graph model used**

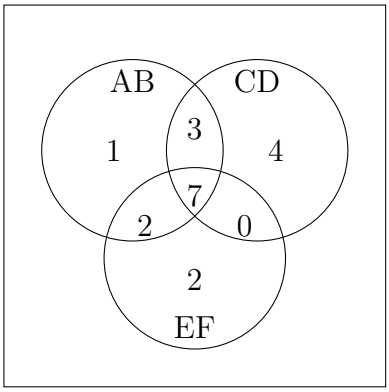

Figure B.19: ***Graph model comparison for SB128 Hearing Id#123.*** **Venn diagram for showing the number of shared pull quotes generated from SB128 Hearing Id#123 using different graph models and ignoring the quote selection method**

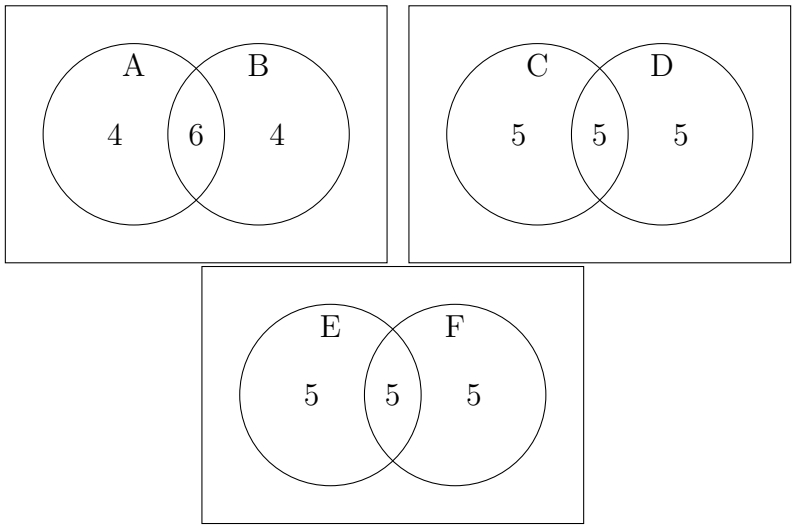

Figure B.20: ***Quote selection method comparison for SB128 Hearing Id#139.*** **Venn diagram for showing the number of shared pull quotes selected from SB128 Hearing Id#139 by different quote selection methods grouped by the graph model used**

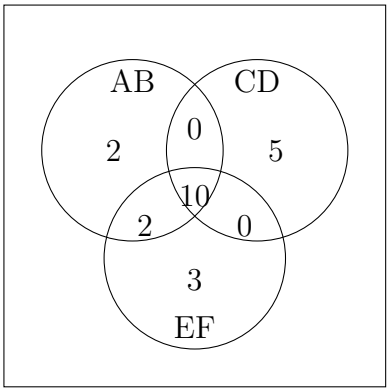

Figure B.21: ***Graph model comparison for SB128 Hearing Id#139.*** **Venn diagram for showing the number of shared pull quotes generated from SB128 Hearing Id#139 using different graph models and ignoring the quote selection method**

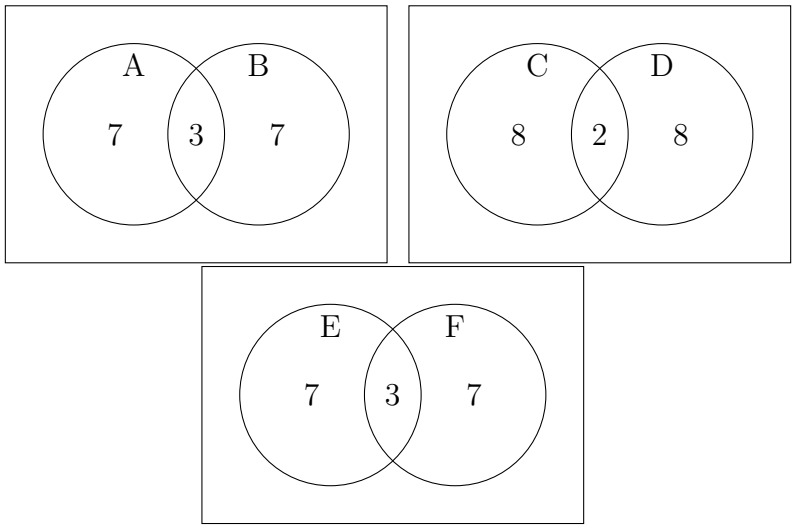

Figure B.22: ***Quote selection method comparison for SB277 Hearing Id#308.*** **Venn diagram for showing the number of shared pull quotes selected from SB277 Hearing Id#308 by different quote selection methods grouped by the graph model used**

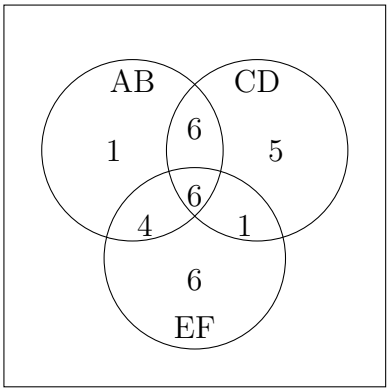

Figure B.23: ***Graph model comparison for SB277 Hearing Id#308.*** **Venn diagram for showing the number of shared pull quotes generated from SB277 Hearing Id#308 using different graph models and ignoring the quote selection method**

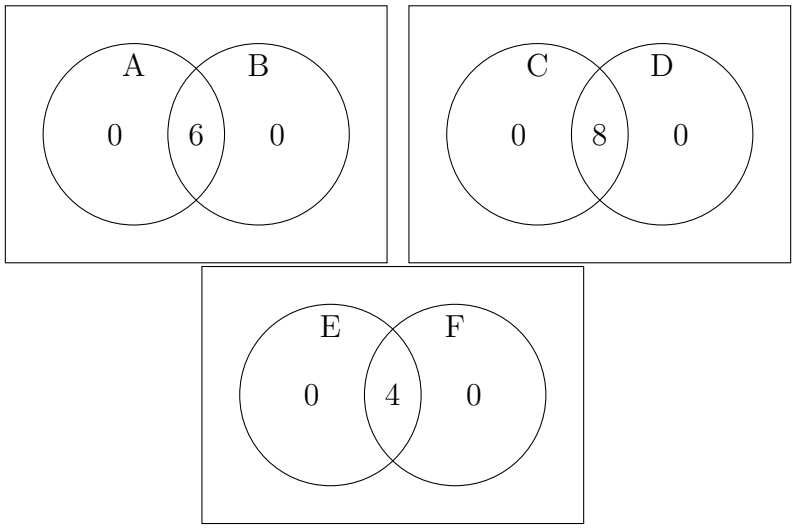

Figure B.24: ***Quote selection method comparison for SB329 Hearing Id#161.*** **Venn diagram for showing the number of shared pull quotes selected from SB329 Hearing Id#161 by different quote selection methods grouped by the graph model used**

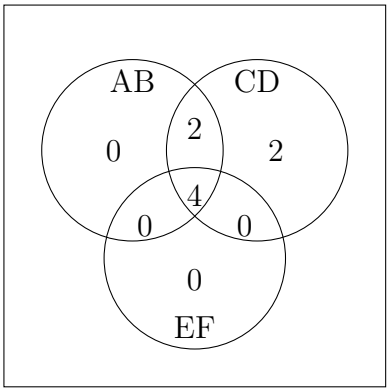

Figure B.25: ***Graph model comparison for SB329 Hearing Id#161.*** **Venn diagram for showing the number of shared pull quotes generated from SB329 Hearing Id#161 using different graph models and ignoring the quote selection method**

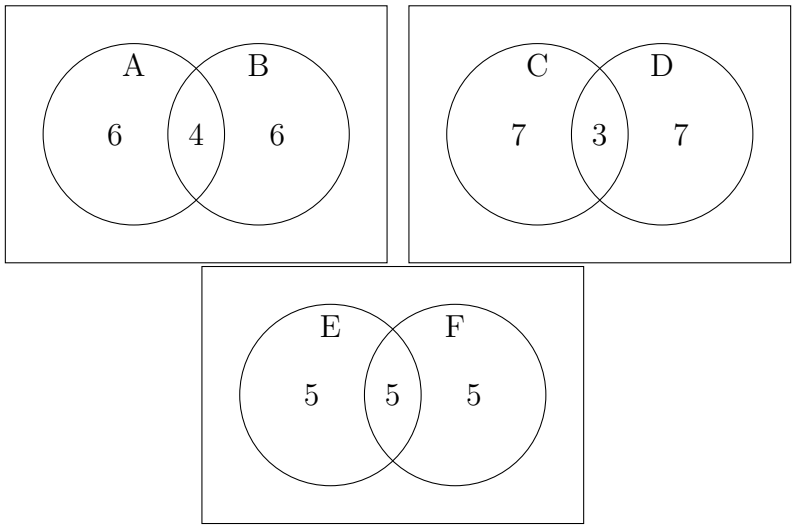

Figure B.26: ***Quote selection method comparison for SB350 Hearing Id#113.*** **Venn diagram for showing the number of shared pull quotes selected from SB350 Hearing Id#113 by different quote selection methods grouped by the graph model used**

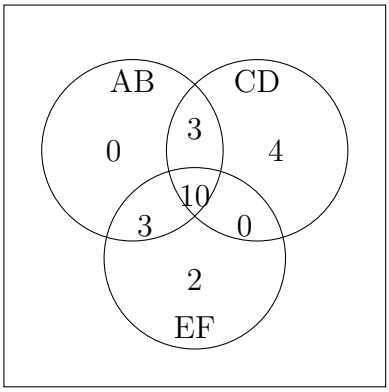

Figure B.27: ***Graph model comparison for SB350 Hearing Id#113.*** **Venn diagram for showing the number of shared pull quotes generated from SB350 Hearing Id#113 using different graph models and ignoring the quote selection method**

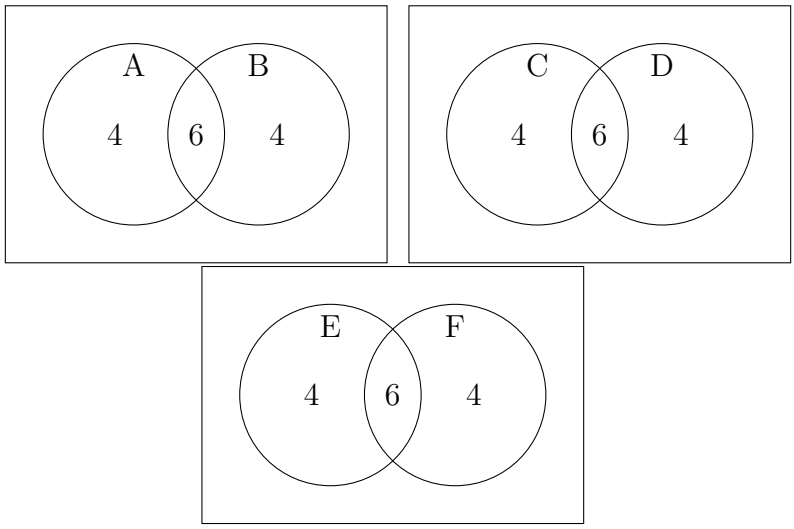

Figure B.28: ***Quote selection method comparison for SB350 Hearing Id#413.*** **Venn diagram for showing the number of shared pull quotes selected from SB350 Hearing Id#413 by different quote selection methods grouped by the graph model used**

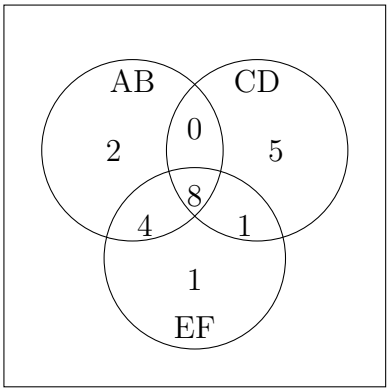

Figure B.29: ***Graph model comparison for SB350 Hearing Id#413.*** **Venn diagram for showing the number of shared pull quotes generated from SB350 Hearing Id#413 using different graph models and ignoring the quote selection method**

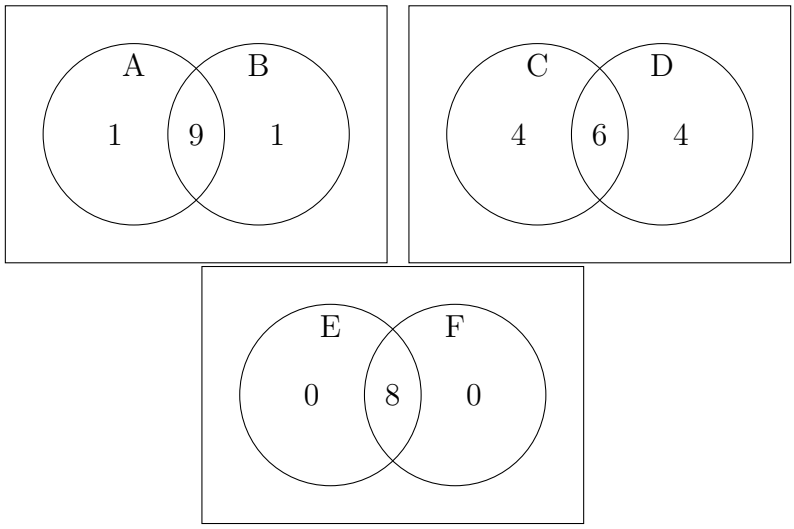

Figure B.30: ***Quote selection method comparison for SCA14 Hearing Id#1266.*** **Venn diagram for showing the number of shared pull quotes selected from SCA14 Hearing Id#1266 by different quote selection methods grouped by the graph model used**

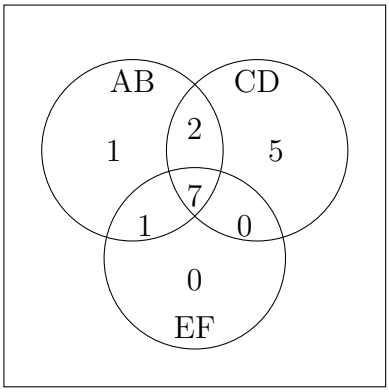

Figure B.31: ***Graph model comparison for SCA14 Hearing Id#1266.*** **Venn diagram for showing the number of shared pull quotes generated from SCA14 Hearing Id#1266 using different graph models and ignoring the quote selection method**

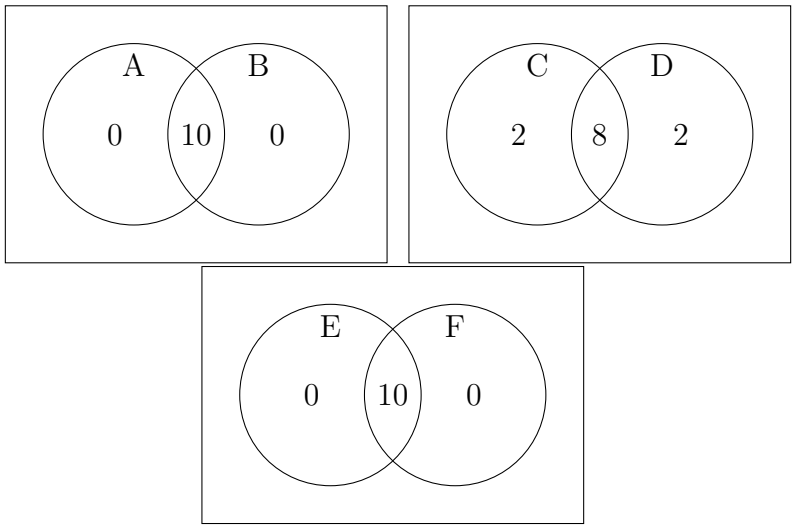

Figure B.32: ***Quote selection method comparison for SCA14 Hearing Id#1284.*** **Venn diagram for showing the number of shared pull quotes selected from SCA14 Hearing Id#1284 by different quote selection methods grouped by the graph model used**

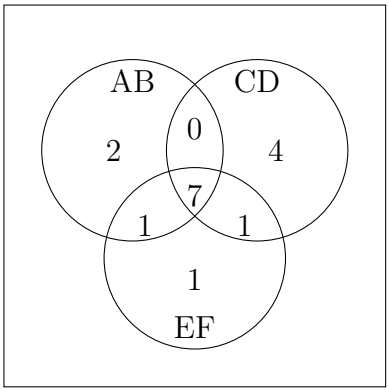

Figure B.33: ***Graph model comparison for SCA14 Hearing Id#1284.*** **Venn diagram for showing the number of shared pull quotes generated from SCA14 Hearing Id#1284 using different graph models and ignoring the quote selection method**

Id	Quote
0	So some have suggested that we're pushing this as an alternative to hospice care, palliative care, that's not true in fact we think this bill will help advance greater awareness in participation with existing end-of-life options in treatment and care.
1	And again we have to evaluate as legislators, a balance between a public policy good for people living with a terminal illness and the compelling testimony we've heard today.
2	It has to be fifteen days in addition to that diagnosis before someone can request medication.
3	And beyond that once the patient under what is proposed in SB 128 goes down this road, ends their life, the physician who is completing the death certificate is required in this law to indicate that the that the patient died of the underlying illness not by taking their own life.
4	SB 128 is aptly entitled the End-of-Life Option Act, it permits a physician to positively respond to the request of a terminally ill decisionally capable person for a prescription which taken as directed will enable him or her to determine the time and manner of an imminent an inevitable death.
5	Where assisted suicide is legal under this system, an heir , someone who stands to inherit or an abusive caregiver is allowed to steer the person towards assisted suicide, witness their request as was said, pick up the lethal dose, and in the end...

6	Hello, my name is Toni Broaddus, I am the California campaign director for Compassion and Choices and in case you haven't figured it out I just want to let you know that Compassion and Choices strongly supports this legislation on behalf of the two-thirds of Californians who want to have this option at the end of life, thank you.
7	And, in the discussion by our the attorney the representative, here I'm struck by the fact that in your argument he made no reference to the fact that these people had been told that they were terminally ill. Age, hopefully, it's not a terminal illness although I guess you could argue that it is.
8	Aid-in-dying under this law is not a replacement for palliative care or for hospice, it is an additional end-of-life option to be considered by the patient and by all physicians whether they be a hospice doc, palliative care specialist, or an oncologist.
9	Senior citizens, particularly those now in what I would call assisted living situations whether in residential communities for living with family or friends, in my experience these people aware that they were now depending on others more than they had previously, sometimes even for food and shelter and always, always for for emotional support and well-being... have been readily influenced by the desires, comments, criticisms, even facial expressions of others whether family members, nurses, doctors, social workers or yes even attorneys.We often times wait several weeks to hear back from clients on the available options they have and from my participation in those discussions and what I wish to share with you this afternoon, I have learned that a certain population particularly vulnerable to persuasion by others.

10	What is our job and where do we default in terms of the rights of folks to engage in making decisions?
11	This is of particular concern to the disability community which includes many people like John Norton who've out lived terminal diagnoses for years and even decades.
12	But, I totally support you in any decision you make about your life at this point.
13	For seventeen years the laws of Oregon have allowed a terminally ill mentally competent person to take control of their own dying process and decide for themselves the amount of suffering they may have to indoor and if at all they need to utilize this end-of-life option.
14	Those who maintain that thisis a euphemistically and misleading term for physician assisted suicide failed to acknowledge the significant transformation in attitudes and perspectives among many health care professionals over the last twenty years.
15	I think if the oncologist could tell you hundreds of stories of physicians who, I'm sorry, patients who told they have 6 months to live and they're still sitting here many years later.
16	Before she took her life using lethal drugs, yet the Director of compassion Choices Washington said that her situation represented quote none of the red flags.
17	Even having this concept out there even for a physician who does not participate we think poisons the physician patient relationship and, makes it that much more difficult to talk about the legal and what we believe more appropriate end-of-life options.

18	Again I believe the Oregon record is a very strong record of the built-in safeguards protecting against abuse or coercion.

Table B.1: *SB128 Hearing Id#123 pull quotes*

Id	Method Source						Rating Counts					
	A	B	C	D	E	F	DK	B	Q	M	G	Best Quote
0				✓			0	0	0	1	4	1
1	✓	✓		✓			0	1	1	0	3	0
2			✓				0	2	2	1	0	0
3	✓	✓	✓	✓	✓	✓	0	0	0	2	3	0
4	✓	✓	✓		✓	✓	0	0	0	0	5	2
5		✓		✓	✓	✓	0	0	1	2	2	0
6	✓	✓	✓	✓	✓	✓	0	1	1	3	0	0
7	✓	✓	✓	✓	✓	✓	0	2	3	0	0	0
8		✓			✓	✓	0	0	1	0	4	1
9	✓		✓		✓		1	1	0	1	2	1
10		✓					1	2	0	1	1	0
11	✓				✓	✓	0	2	2	0	1	0
12			✓	✓			0	5	0	0	0	0
13				✓			0	0	1	2	2	0
14	✓	✓	✓	✓	✓	✓	0	0	1	2	2	0
15					✓	✓	0	0	3	1	1	0
16						✓	0	3	2	0	0	0
17	✓	✓	✓	✓			0	1	2	1	1	0
18	✓		✓				0	0	1	3	1	0

Table B.2: *SB128 Hearing Id#123 method sources and rating counts*

(Rating abbreviations: DK – Don't know, B – Bad, Q – Questionable, M – Maybe, G – Good)

Id	Quote
0	I represent over 500 grassroots sponsors of over 120 Southern California mothers, that were able to come here and represent those who could not be, to oppose this bill.
1	In fact, let's go back to AV12109, my dear opponents here who are now supporting the bill, were right here testifying, saying that doctors were going to force people to get vaccinated by refusing to sign, by refusing to sign the form saying that they consulted.
2	It seems as though, if you want to send your child to public school and you have no other option to home school, you must sign the consent form.
3	Everybody has access to vaccines and it seemed unbelievable to them and scary that their young children their young babies who could not be vaccinated could be subject to these diseases that they have seen first hand in horrifying ways and so to say it's not a public health crisis when one of the closest urgent cares to my district has to be shut down.
4	The only other thing I wanted to say about Serrano and this issue of the compelling state interest is that at the end of the day, the core of the equal protection concern really comes down to whether the bill discriminates facially against the suspect class.
5	I understand AB 2109 has had some impact in some areas, capturing those who use PBEs because it was easier than going to the doctor perhaps, but in other areas, something else is happening, creating an uneven landscape of protection for our children and our society.

6	And the family physicians have been in support of this bill since its introduction, along with the entire physician community, because they are not only educating parents and patients about vaccines, and administering the vaccines, but they are also treating patients, when they are not vaccinated or when they develop diseases.
7	Then if they decide not to have their child immunized, under this bill, they would have to seek out other options like home schooling or independent study.
8	The purpose of AB 2109 was to get our immunization rates up high enough so we can protect herd immunity, community immunity in their schools, prevent the spread of outbreaks.
9	Now, we have worked with this committee to ensure that SB 277 protects our most vulnerable citizens, ensures that every child has an opportunity to receive an education, and protect our constituents from deadly, contagious, and preventable diseases.
10	Elisa Vargas from Meadow Vista, a single mother, a microbiologist, and a school teacher, and I strongly oppose this bill.
11	What I've seen with all the reading, the bottom line is despite originating in one of the most densely populated places in the country, the 2015 Disneyland measles outbreak was successfully contained, and only affected 0.00035% of the state's population.

12	And would like to be able to continue that discussion with you on ensuring the fact that the doctor would not have this, what I'm hoping is an expanded judgement around the medical exemption, but also an expanded judgement around alternate schedules beyond what is printed in the current regulations and that's something that I think is very important and critical as we look at the daycare and childcare issue.
13	In this case, the court won't even look at are vaccines a compelling interest.
14	It's a narrow analysis, and what the court looks at is, is there a compelling state interest to remove the personal belief exemption in light of what's going on.
15	My name is Paul Nelson, I'm a business owner from Huntington Beach, California, and I strongly oppose this bill.
16	And so we are certainly very concerned over that, especially with the continuing outbreak in Disneyland and other outbreaks that we are concerned about, that it's not, unfortunately, gonna do the job to prevent outbreaks in our community and again, this is a 19% decrease that followed a over 300% increase in PBEs that have occurred prior to that decrease.
17	I believe that current law states that a physician has complete, professional discretion over the writing of a medical exemption.
18	And this would not be a problem for those in the mainstream who choose to have their children vaccinated if the vaccines were 100 % effective, but they are not.

19	I'm a teacher and a member of the American Criminal Justice Association, and I have permission to speak on the behalf of over 100 of my colleagues to oppose this bill, as well as every member of my family.
20	Hello, my name is Cynthia Rivera from El Centro, California, mother of three fully vaccinated children, grandmother of partial vaccination and I strongly oppose this bill.
21	In fact, most recently, the state of Vermont, the governor who signed their bill to eliminate personal belief exemptions in the state of Vermont, one of the reasons he stated was there was a previous bill, was different than 2109, but they had a bill requiring education of parents.
22	It's been stated that there is a responsibility in the freedom of choice, and therefore, those choosing to limit vaccines face the consequence of not being able to send their child to school,
23	I am Scott Folsom, a parent leader in Los Angeles Unified School District and also a member of the LAUSD Trust for Children's Health, which operates on school clinics.
24	There was a lot of attempts to cause them to be medicated before they came to school, so the federal government and California followed, passed a law and said, you cannot condition a kid's right to attend school based on them taking any sort of medication.
25	If we want to stop outbreaks, we need to get our immunization rates higher, not just across the state, but also in individual communities as well because otherwise those communities have become the basis and the seeds for outbreaks that'll spread into other parts of the state that's around them.

26	This bill is about having your child enroll in school or potentially, the existing law for day care.
27	The amendments were here given verbally, the next step will just be, if it doesn't go to a probst, will be on the floor, and the input from the public is not going to be there, not going to have the opportunity to do what we just had here, and I think that doesn't serve us well.
28	That nothing in this section shall prohibit a pupil that qualifies for an IEP, pursuant to Federal Law and Section 56026 of the Education Code from accessing any special education and related services required by the Individualized Education Program.

Table B.3: *SB277 Hearing Id#308 pull quotes*

Id	Method Source						Rating Counts					
	A	B	C	D	E	F	DK	B	Q	M	G	Best Quote
0		✓		✓			0	2	2	1	0	0
1	✓		✓				0	2	1	2	0	0
2			✓		✓		0	0	2	2	1	0
3		✓		✓		✓	0	0	3	2	0	0
4		✓				✓	0	0	1	4	0	0
5					✓		0	0	2	3	0	0
6				✓			0	0	0	1	4	0
7	✓		✓				0	0	0	3	2	0
8				✓			0	0	0	1	4	1
9		✓		✓		✓	0	0	0	1	4	0
10						✓	0	2	2	1	0	0
11	✓				✓	✓	0	0	0	3	2	1
12		✓		✓		✓	0	2	1	1	1	0
13	✓		✓				0	3	1	1	0	0
14					✓		0	1	1	2	1	0
15					✓		0	3	1	1	0	0
16	✓	✓	✓	✓	✓	✓	1	1	1	1	1	0
17				✓			1	0	2	1	1	0
18	✓	✓	✓	✓			1	1	0	1	2	0
19	✓				✓		0	2	2	0	1	0
20						✓	0	2	2	1	0	0
21	✓	✓	✓		✓	✓	0	3	1	1	0	0
22				✓			0	0	1	2	2	0
23	✓		✓		✓		0	4	1	0	0	0

24		✓					0	1	2	2	0	0
25					✓		0	0	1	0	4	0
26	✓		✓				0	2	1	2	0	0
27			✓				0	4	1	0	0	0
28		✓				✓	0	3	2	0	0	0

Table B.4: *SB277 Hearing Id#308 method sources and rating counts*

(Rating abbreviations: DK – Don't know, B – Bad, Q – Questionable, M – Maybe, G – Good)

Id	Quote
0	But I just think, I can't stress enough how important it is that I think that this should be part of a negotiation, and I apologize to the proponents that this wasn't started a long time ago.
1	Indeed, over the years, legislators have authored nearly 10 different measures to require a bill to be in print for 72 hours before it can be voted on.
2	By abundant court precedent, in particular set by the litigation over the ill fated Legislature Reform Act of 1983, in the absence of a constitutional definition of the distinction between what a Committee of the Legislature might be and what a standing Committee of the Legislature might be, the distinction will default to the Legislature's own rules.
3	The purpose is, in my judgement, from my own personal view, I can't speak for the author.
4	Certainly I don't have the authority to take amendments now, but I think that certainly there's an opportunity for discussion to be able to avoid putting this on the, well it'll be put on the ballot as a constitutional amendment by the Legislature, which is a better thing to do.
5	There are times when it's not appropriate and not the best way to deal with issues and we think the ones at hand with these two bills and at hand with the initiative are the kinds of issues that are best dealt with through the legislative process.
6	Well, it's part of this tension that exists between the public's important right to be able to make the law or to question what we do and the power of the Legislature to act responsibly is this initiative process and we use to have in the law what was called the indirect initiative.

7	Let me finish by asking, I don't want you to have to speak for the author, but based on what you've said, is that your position that the goal, really, of SCA 14 is to put options on the ballot and not necessarily proceed to strive to get the best product on the ballot?
8	But I'll tell you for somebody who spent a year and a half of my life and know all those thousands of e-mails and you can ask A lot of the people here were testifying at all those endless meetings and conference calls we had creating the 1253, which was the Ballot Transparency Act.
9	The third element you talked about was the definition of committees, standing committees versus other committees.
10	So this measure in broad terms as you have before you will certainly talk about the details as we go forward, provides that the Legislature provide at least 72 hours notice of a measure in its final form before taking it up seeking to avoid what is commonly referred to in the legislative parliaments and gut and amends.
11	And I think but at the end of the day if what we produce is something that works, I think it's a fair and a reasonable solution which was the compromise that was just mentioned by the League of Women Voters that was envisioned in 1253, to try to bring people to the table to be able to create an honorable discussion, however late, between the legislative branch of government and those folks who are proponents of initiatives.
12	And I think that when you're talking about protecting the public's right to know, it's very important that it is constitutional and not at the whim of today's legislators or tomorrow's legislators.

13	I would just appeal to as long as, at the end of the day, they exercise good faith, negotiate with the lawyers in the Legislature and the staff in Legislature to try to come up with something that achieves the same objectives that they're achieving that they stand down and we have one measure going forward and we solve the problem which they have been solely the catalyst for making it happen.
14	SCA 14 does not protect the public's access to recordings of the Legislature's public proceedings.
15	And none has ever passed out of a policy Committee, let alone off the Senate and Assembly Floor.

Table B.5: *SCA14 Hearing Id#1266 pull quotes*

Id	Method Source						Rating Counts					
	A	B	C	D	E	F	DK	B	Q	M	G	Best Quote
0	✓	✓		✓			0	4	0	1	0	0
1			✓	✓			0	0	0	0	5	0
2		✓			✓	✓	0	0	2	2	1	0
3	✓		✓	✓			0	5	0	0	0	0
4	✓	✓		✓	✓	✓	0	2	0	3	0	0
5	✓	✓	✓	✓	✓	✓	0	1	1	3	0	0
6	✓	✓		✓	✓	✓	0	2	1	1	1	0
7	✓	✓	✓	✓	✓	✓	0	0	2	1	2	0
8	✓	✓	✓	✓	✓	✓	1	2	1	1	0	0
9			✓				0	3	0	1	0	0
10	✓	✓	✓	✓	✓	✓	0	1	0	2	2	0
11	✓	✓	✓		✓	✓	0	1	1	1	2	0
12	✓	✓					0	0	0	4	1	1
13			✓				1	1	1	1	1	0
14				✓			0	0	0	0	5	0
15			✓				0	2	1	2	0	0

Table B.6: *SCA14 Hearing Id#1266 method sources and rating counts*

(Rating abbreviations: DK – Don't know, B – Bad, Q – Questionable, M – Maybe, G – Good)