October 2021

Natural Language Processing for Lexical Corpus Analysis

Abram Kaufman Handler
University of Massachusetts Amherst

NATURAL LANGUAGE PROCESSING FOR LEXICAL CORPUS ANALYSIS

A Dissertation Presented

by

ABRAM HANDLER

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2021

College of Information and Computer Sciences

NATURAL LANGUAGE PROCESSING FOR LEXICAL CORPUS ANALYSIS

A Dissertation Presented

by

ABRAM HANDLER

Approved as to style and content by:

Brendan O'Connor, Chair

Brian Dillon, Member

Mohit Iyyer, Member

Narges Mahyar, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

ACKNOWLEDGMENTS

Thanks to Steve Myers, Joe Foxhood and especially Vassil Roussev for helping me understand that I was actually a computer scientist. Thanks to Katherine A. Keith and Su Lin Blodgett, who have been thoughtful, creative, and encouraging labmates throughout my time as a Ph.D. student. Thanks to my friend Javier Burroni for extensive mathematical help and advice over many years. Thanks to Joe Susnick and Sam Handler for practical coding guidance and real-world perspective on computing research. Thanks to Nick Eubank and Oren Ziv for helping guide me through the academic world; and thanks also to Nick for telling me to get a dog. Thanks to current and former University of Massachusetts graduate students and NLP reading group members for years of helpful suggestions and discussions, especially Nader Akoury, Haw-Shiuan Chang, Andrew Drozdov, Jeffrey Flanigan, John Foley, Neha Nayak Kennard, Kalpesh Krishna, Mahmood Jasim, Nicholas Monath, Sheshera Mysore, Emma Strubell, Pat Verga and Tu Vu. Thanks to my advisor Brendan O'Connor for many years of support and guidance. Finally, thanks to my friends, family and partner Eliana Bronstein for helping me undertake so many years of dedicated study.

ABSTRACT

NATURAL LANGUAGE PROCESSING FOR LEXICAL CORPUS ANALYSIS

SEPTEMBER 2021

ABRAM HANDLER

B.A., COLUMBIA UNIVERSITY

M.S., UNIVERSITY OF NEW ORLEANS

M.S., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS, AMHERST

Directed by: Professor Brendan O'Connor

People have been analyzing documents by reading keywords in context for centuries. Traditional approaches like paper concordances or digital keyword-in-context viewers display all occurrences of a single word from a corpus vocabulary amid immediately surrounding tokens or characters, to show readers how individual lexical items are used in bodies of text. We propose that these common tools are one particular application of a more general approach to analyzing documents, which we define as lexical corpus analysis. We then propose new natural language processing techniques for lexically-focused corpus investigation, and demonstrate how such methods can be used to create new user-facing tools for analyzing corpora.

Our contributions are divided into three parts. In Part I, we consider how to represent a corpus lexicon to best reflect human mental and linguistic models of

a domain, and propose a natural language processing (NLP) method for enriching a unigram corpus vocabulary with multiword phases. In Part II, we consider how lexical systems might show query terms in context to best satisfy user search need, and offer several new techniques focused on summarizing mentions of a query term in context. Finally, in Part III, we apply our proposed NLP methods towards new user-facing systems for lexical corpus analysis, and present user studies with journalists and historians which investigate how new lexical tools can help such users in their work.

TABLE OF CONTENTS

PART I: HOW TO REPRESENT THE LEXICON

PART II: HOW TO SHOW LEXICAL ITEMS IN CONTEXT

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

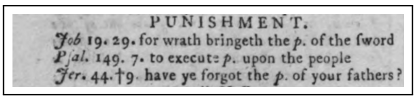
INTRODUCTION

1.1 The concordance

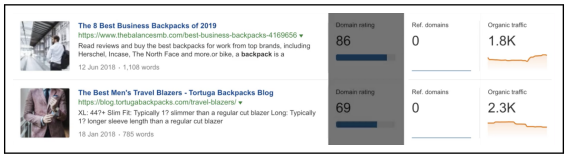
In the first half of the thirteenth century, Dominican monks at the Saint-Jacques monastery in Paris compiled the first concordance of the Christian Bible, a resource listing each occurrence of some 10,000 word types within the Latin text. Working from notes and drafts in the different handwriting of different men from the monastery, Rouse and Rouse [289] describe how individual monks likely copied the locations of words within the corpus onto folded sheets devoted to specific portions of the alphabet (e.g. a single monk likely composed one leaflet devoted to words from *stabile* to *structor*) and then organized the hand-written sheets into small volumes for distribution. In medieval Europe, priests and scholars used these and other similar references to compose sermons, impose uniform doctrine, and inform theological debates [289, 31].[1]

Much like such early religious resources, contemporary printed and digital concordances (also called Keyword-in-Context indexes [208] or KWIC viewers) help users analyze document collections which are too large to read, by showing keywords amid immediately surrounding text (Figure 1.1). Over the past two centuries, people have used concordances to analyze corpora in diverse fields like history [356], sociology

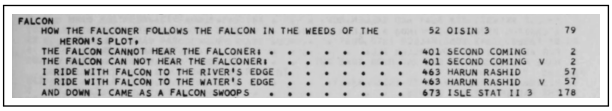
[1]The first Saint Jacques concordance indexed the locations of each word in the text, without showing use in context. Soon after, Dominican monks in England compiled a similar volume showing a full sentence of context for each occurrence of a word type [289]. Hugh of St. Cher, an administrator at Saint-Jacques, is sometimes described as either the single author [172, 157] or overseer [199] of the first concordance; authoritative work from Rouse and Rouse reports that there is little evidence for these claims. Even earlier religious texts also list occurrences of word types within biblical text, in alphabetical order [91].



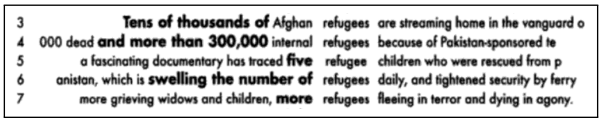
(a) An 18th-century concordance, showing occurrences of the word "punishment" (abbreviated as p) in the Christian Bible. *"Hereby many important things may be observed at one view,"* the author writes [78], *"without the trouble of turning over several volumes."*



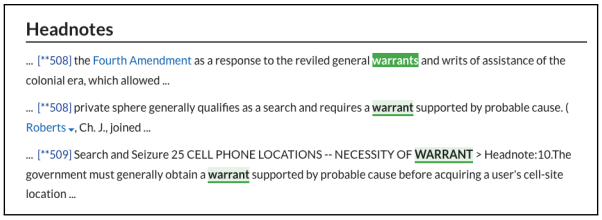
(b) The Ahrefs Content Explorer, a 21st-century faceted, web-scale concordance for online marketing. *"We can use Context Explorer to find relevant guest blogging opportunities,"* explains a tutorial [1]. *"Because these blogs are mentioning your target keyword ... [among other criteria], they are clearly publishing content that is relevant to your niche."*



(c) A 20th-century printed concordance of the works of William Butler Yeats [267]. *"Consider the advantages of cataloging the ... birds that beat ... through Yeats's poems*," one scholar writes. *"I count, for a beginning, some 8 hawks, 21 owls, 6 bats..."*



(d) A concordance of British newspapers, used to analyze media descriptions of refugees [17]. *"In line 7 they are 'fleeing',"* one author writes, before going on to describe other similar concordance lines. *"This pattern suggests that refugees are often described in terms of their movement."*



(e) The KWIC view feature in Lexis Advance [72], a contemporary commercial concordance used for legal research. A tutorial explains [197] that the KWIC view feature *"allows you to determine the context of the search terms, and whether or not the document is pertinent."*

Figure 1.1: Uses of concordances across centuries and domains. This work proposes generalizations of this enduring and widely-used text analysis technique.

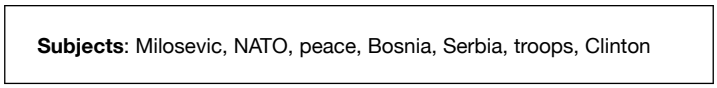
[280, 110], classics [301, 346], politics [234, 222], forensics [75], literary scholarship (Figure 1.1c), media analysis (Figure 1.1d), marketing (Figure 1.1b) and law (Figure 1.1e). Concordances are also particularly important to lexicography [183], because uses of a word in context give information about the word's meanings and grammatical roles in a corpus.[2] At the time of writing (January 2021), one popular concordance tool [196] has been cited over 1,200 times on Google scholar.

1.1.1 Uses of the concordance

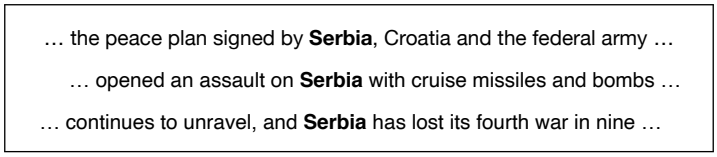
We propose that concordances are useful because they offer multiple perspectives into the contents of a corpus that is too large to read (Figure 1.2). Most obviously, concordances show mentions of some query keyword within the context of immediately surrounding tokens or characters. By reading mentions of the keyword in context (e.g. Figure 1.2b), a reader can quickly gain a sense of a keyword's use across a corpus. We thus say that concordances offer a *token-level perspective* on a corpus, as they show keywords amid immediately surrounding token spans from text. For instance, Hammo et al. [137] review keywords in context to study changes in the meaning of words across time and Zinn [356] similarly reviews keywords in context to study institutional shifts in the United Kingdom. In the literature from corpus linguistics, reviewing token spans showing some query keyword in context is sometimes described as reading the "lines" of a concordance [230].

However, because concordances index the locations of each word type in a text, concordances can also help people quickly review (and sometimes count) the concepts or topics in a corpus. For instance, Parrish [267] uses a printed concordance to

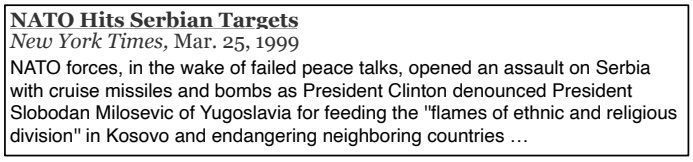
[2]Some theories of semantics propose that the meaning of a lexical item is determined by the distribution of co-occurring words [146, 111]. From this perspective, reading some co-occurring words in context is an informal sample of the true co-occurrence distribution, which defines a word's semantics. Writing from the corpus linguistics tradition, McEnery and Hardie [231] describe the process of identifying and building linguistic intuition using concordances as "collocation-via-concordance," a complement to more formal statistical techniques.



(a) Concordances offer a *type-level perspective* on a corpus. By reviewing (and sometimes counting) word types in the concordance index, the user may gain an overview of the subjects in a body of text. This figure shows a list of subjects which occur frequently in a mock corpus focused on Balkan wars of the 1990s, which might be identified using a concordance index.



(b) Concordances offer a *token-level perspective* on a corpus, by showing token spans mentioning a keyword in documents. In this figure, the term "Serbia" is shown in token spans that fit within 150 pixels of screen space (respecting word breaks).



(c) Concordances offer a *document-level perspective* on a corpus, by guiding users to underlying text. In this case, the figure shows the start of a full document mentioning "Serbia".

Figure 1.2: Concordances help users gain three perspectives on a large corpus.

manually count references to birds and animal types in the works of Yeats (Figure 1.1c), and García-Marrugo [119] uses an electronic concordance to count references to military actors (e.g. "guerrillas" vs. "paramilitaries") in Colombian newspapers.[3] Thus we say that concordances also offer a *type-level perspective* into a corpus. This use of a concordance may be interpreted as a form of exploratory data analysis [328], as it helps a user gain an overview of unfamiliar documents.

[3]Some work seemingly unrelated to concordance tools uses word importance scores to rank word types a corpus lexicon [262, 241]. If we interpret concordances as a data structure mapping word types to their locations in a corpus, much like the traditional inverted index described in Manning et al. [216], we can use a concordance to compute many traditional word importance scores, which are often based on statistics about the frequencies of words in context (e.g. pointwise mutual information, as in O'Connor [262]).

Finally, many traditional printed or hand-written concordances include references to underlying documents. For instance, early Dominican concordances pointed users to the locations of words within the Christian bible, and some electronic KWIC viewers (e.g. Figure 1.1e) allow users to click to access underlying sources.[4] Therefore, we say that concordances also offer a *document-level perspective* of a corpus, by helping users navigate to underlying source text.

Overall, it is possible to interpret the *type-level perspective*, *token-level perspective* and *document-level perspective* as offering a unified and lexically-oriented view of a corpus, with varying levels of context. In the *type-level perspective*, word types from the lexicon are presented or analyzed without surrounding context. In the *token-level perspective*, some occurrences of some query word from the lexicon are shown within the context of immediately surrounding tokens. Finally in the *document-level perspective*, words within the lexicon are shown within the context of an entire document.

1.2 Introducing lexical corpus analysis

Concordances have many clear advantages as tools for helping people form and answer questions[5] from large repositories of unlabeled digital text:

- Concordances do not require domain-specific annotation and are not beholden to any given linguistic formalism (e.g. abstract meaning representation [24]). Concordances thus work reliably, without supervision, across heterogeneous text. Many NLP tools do not enjoy this advantage (see, for instance, work from Bamman [22]).

[4]In information retrieval (IR), search engines are said to offer query-biased snippets [325], single-document summaries shown on a search engine results page (SERP). Such summaries are designed to help users decide if they should navigate to a query-responsive document. Concordances lines can be seen as a particular kind of query-biased snippet, which always show keywords amid immediately surrounding text.

[5]In the HCI literature, this process is sometimes described as "sensemaking" [277].

- Concordances are simple and transparent; meaning the user can easily and accurately understand how a concordance organizes documents, without having to learn a new way to think about textual data (e.g. learning to interpret t-SNE plots [333]).[6]

- Concordances run quickly. Decades of research in information retrieval (IR) offers time and space efficient techniques for indexing the locations of words in documents (see Manning et al. [216]). Concordances can apply this work towards scalable, low-latency, interactive data analysis. Slow interfaces are known to hinder a user's work [205].

- Concordances help people simplify corpora by breaking documents into categories or groups (e.g. all documents containing the word "falcon", Figure 1.1c), without requiring the labor-intensive work of constructing an ontology (which may be brittle or incomplete [329]).

- Because concordances do not attempt statistical inference about unstructured text, the tools run reliably without inevitable mistakes from probabilistic natural language processing, which introduces tricky design challenges (see Amershi et al. [10]).[7]

Inspired by such advantages and noting the broad and long-lasting uses of traditional concordance tools, this work proposes a wider study of lexically-focused corpus investigation. **We argue that modern natural language processing (NLP) methods can be used to create new, digital tools for analyzing text through** **Thesis**

[6]See Doshi-Velez and Kim [90] for a broader discussion of transparency in machine learning.

[7]Some of these admirable properties can be described in terms of Nielsen's well-known usability heuristics [256], which emphasize the need for control, consistency and error prevention in user interfaces. For instance, concordances can be said to favor recognition over recall (a Nielsen heuristic) because the user does not have to learn a new mental model of text (e.g. topics, from LDA [38]) to use a concordance.

(1) reviewing a corpus lexicon and (2) examining items from that lexicon in context. We describe this process as lexical corpus analysis.

A tool for lexical corpus analysis both defines a corpus lexicon and shows items from the lexicon amid surrounding text. By this definition, traditional hand-written, printed or electronic concordances are one possible tool for lexical corpus analysis, showing single words from a unigram lexicon amid some number of immediately surrounding words, characters or pixels of context.[8] However, other ways of extracting and presenting lexical items in context may be better suited to different users in different circumstances. For instance, a user studying conflict mediation might need to review mentions of multiword units, which express key subjects from the domain (e.g. "peace treaty"). But such units would not be indexed in a traditional 1-gram concordance. Similarly, a historian studying media depictions of a female astronaut will want to review descriptions of the astronaut's appearance in news stories (Chapter 7). But such descriptions may not fall within the immediately surrounding context shown in a traditional KWIC viewer (Figure 1.3). Thus while concordances are powerful tools, this work proposes that other lexically-oriented systems may better serve specific groups of users with particular information needs.

1.2.1 Exploratory and query-focused lexical corpus analysis

In this user-centered investigation, we distinguish between two particular kinds of lexical corpus analysis, which reflect different user goals and objectives.[9] In **exploratory lexical corpus analysis**, the user will seek to gain "insights, understanding, and surprises" [45] by reviewing a *type-level perspective* on a corpus. For instance,

---

[8] Different implementations of concordances define the context using different units [286, 207, 262].

[9] This distinction is inspired by Tukey's division between exploratory and confirmatory statistics [328]. We consider query-focused analysis to be more confirmatory than exploratory, because the user begins with a query which expresses their question. However, this thesis does not consider methods for confirming or rejecting formal quantitative hypotheses about text.

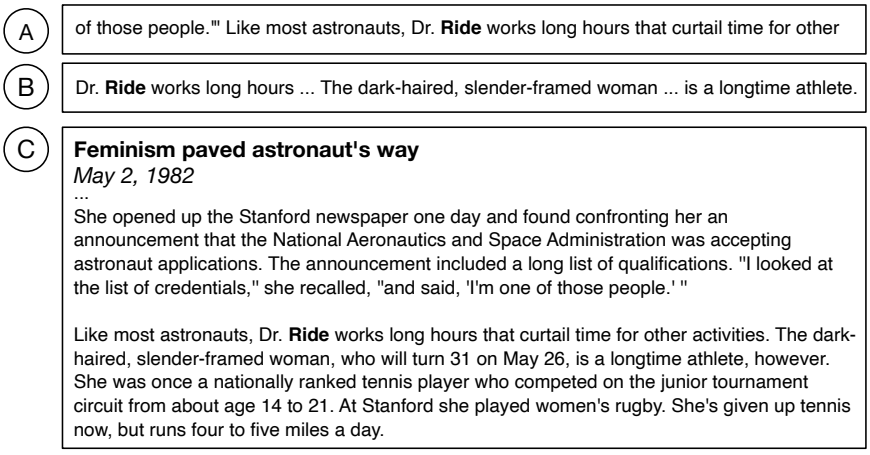


Figure 1.3: A standard concordance line (A), and a line from an alternative tool (B), each showing a mention of Sally Ride from a news article (C). Both A and B are roughly the same length. But B includes descriptions of Ride's attributes (e.g. "slender-framed"), and thus would be more helpful for a historian studying media descriptions of the astronaut. This example comes from Chapter 7.

Chapter 6 describes a system which helps journalists explore important subjects in a corpus, by reviewing unusually frequent word types from particular time periods.[10] Note that in exploratory lexical corpus analysis, the user may not yet have a fully-formed research question.

In **query-focused lexical corpus analysis**, the user already has a hypothesis or research question, which they can express with a query (drawn from the corpus lexicon). The user investigates by reviewing a *token-level perspective* on a collection of documents, which shows mentions of the lexical query in context. For instance, in Chapter 7, we document how a tool built around showing lexical items in context helps historians investigate questions in archives. The term query-focused lexical analysis is inspired by the term query-focused summarization, in which a system

[10]It is possible to interpret reviewing such word types as roughly similar to reviewing quantitative or graphical summaries to explore structured data [328], or reviewing corpus contents in a specialized interface during exploratory search [219]. Other prior work on user-facing text analysis (e.g. Termite [63] or TIARA [203]) also focuses on word types while helping users explore a body of text.

must construct a textual summary based on user's query (see Nenkova and McKeown [247], and Section 1.3.2).

Although we distinguish between the exploratory and query-focused use cases, these two approaches are closely intertwined. For instance, in Chapter 6, we present a system which supports both exploratory and query-focused lexical analysis, using a unified interface. The system first shows frequent lexical items to encourage exploration, and then allows the user to query for individual lexical items in context. Together, borrowing terminology from HCI [277], these exploratory and query-focused phases of investigation can be described as a kind of sensemaking process.

Part I: How to represent the lexicon
Chapter 2. Phrase extraction (NPFST)
Part II: How to show lexical items in context
Chapter 3. Relationship summarization
Chapter 4. Text simplification (clause deletion)
Chapter 5. Text simplification (vertex addition)
Part III: How to design lexical systems for specific user groups
Chapter 6. ROOKIE
Chapter 7. CLIOQUERY

Table 1.1: Lexical corpus analysis is based on defining a corpus lexicon and showing items from the lexicon in context. In Part I of this work, we focus on how to represent the corpus lexicon, and in Part II we focus on how to show items from the lexicon in context. In Part III we apply NLP methods from Part I and Part II in user-facing tools, which examine how to design lexical systems for specific user groups. Section 1.3 describes each part of this work in greater detail.

1.3 Research questions in lexical corpus analysis

In the next three subsections, we review what we assert to be the three major research questions in lexical corpus analysis: how lexical systems might represent the corpus lexicon to reflect user mental models of a domain, how to show lexical items in context based on user search need, and how to design lexical systems for specific user

groups. In presenting each question, we also briefly describe our work towards partial solutions, offering a very short preview of this thesis. Parts I, II and III consider these same questions in detail.

1.3.1 How to represent the lexicon (Part I)

Traditional concordances often index and list occurrences of single word units [261], which we describe as unigrams. On the surface, this approach seems to allow the user to name and investigate any concept from a corpus. For instance, a user might look up the word "falcon" review all mentions of the concept of falcons in Yeats (Figure 1.1c). However, flat lists of single words do a poor job representing many concepts in a domain. For example, meeting minutes from the U.S. Federal Reserve will include discussion of the concept "interest rates". But because "interest rates" consists of two words, this concept will not be included in a unigram concordance index; a user simply can not look up mentions of "interest rates" using a 1-gram concordance.

Poor representation of multiword phrases is one example of a much broader phenomenon: flat vocabulary lists do not capture many aspects of human mental models of a domain. For instance, some concepts in a corpus may have multiple names (e.g. "George Bush" and "Dubya"), use different grammatical forms (e.g. "frequent fliers" vs. "people who fly frequently"; an example from Stubbs [321]), stand in for other concepts (e.g. metonymy like "Brussels pressured London"), share properties with other concepts (e.g hypernymy like "fish" vs. "trout") or draw their meaning from complex lexical composition (e.g. "vegan shoes"). Traditional data structures for representing occurrences of words in context (such as the inverted index described in Manning et al. [216]) do not capture such complexity.

This disconnect limits the utility of current concordances. Some shortcomings are obvious. For instance, if multiword phrases are not included in the corpus lexi-

con, a concordance will not show some concepts from a domain (e.g. "peace treaty"). Similarly, a concordance will not show some occurrences of concepts which have different names (e.g. references to "Dubya" while reviewing mentions of "Bush"). Yet other limitations are more subtle. For instance, the linguist García-Marrugo [119] uses a concordance to manually organize and count complex hierarchical relationships between different names for illegal actors in the Colombian press (e.g. "Front" vs. "FARC" vs. "Tirofijo"[11]). Automated lexical systems which similarly organized the corpus vocabulary might help less experienced users outside linguistics perform such analysis. Improved representation of the corpus lexicon could also support whole new kinds of lexically-focused corpus investigation. "Suppose ... I could simply type in the command supernatural nominatives," one literary scholar writes [157], "and almost immediately receive a list of apparitions, devils, ghosts, ghouls, monsters, mummies, ogres, phantoms, shades, shadows, spirits, vampires, and witches [in a work from Joseph Conrad]."

Thus, motivated by the possibilities of improved lexical systems which better reflect how people think, write and talk about subjects described in text, in Part I of this work, we introduce a high-level research question:

R1: How can lexical systems represent a corpus vocabulary to reflect human mental and linguistic models of a domain?

In Chapter 2 we describe progress towards a partial solution.[12] We propose a computationally-efficient method for enriching a corpus lexicon with multiword phrases (e.g. "interest

[11] A nickname for guerrilla leader Manuel Marulanda.

[12] Some work in user-facing text analysis can also be seen as taking steps to better reflect user mental models of a domain. For instance, the WordSeer system [243, 2] supports searching by root (e.g. for "read" and "reading") to help users find mentions of the same word across grammatical forms, and the SPIKE system allows a user to interactively find aliases using specialized queries [322]. Outside of NLP, successful user-facing digital tools such as the influential Smalltalk programming language [168] and the commonly-used Tidyverse data analysis software [344] also attempt to reflect the ways that people think about their work.

rates"), and compare this technique to baseline phrase extraction strategies. Yet while we show progress towards addressing R1, our work towards answering this broad question is incomplete. We describe many directions and possibilities for future work in Chapter 8, where we also consider future challenges in applying related work from NLP in our user-facing setting.

Note that throughout this work, we use the notation Q to reflect a user's query during lexical corpus analysis. Q is an item from a corpus lexicon. In different chapters, Q will take on different grammatical types, based on different representations of the corpus vocabulary. For instance, Q will sometimes be a single word (e.g. "army" in Chapter 7), a noun phrase (e.g. "human rights" in Chapter 6) or a noun phrase pair (e.g. "peace treaty"–"Bill Clinton" in Chapter 3). We often employ our proposed phrasefinding method from Chapter 2 to enrich a unigram corpus lexicon with noun phrases. Chapter 4 and 5 present NLP methods which would in principle apply to any arbitrary (and potentially gappy) value of Q, regardless of its lexical type.

1.3.2 How to show lexical items in context (Part II)

In query-focused lexical corpus analysis, a user reviews mentions of a query word Q in context. This opens a natural question: how should a system show mentions of a query in documents? KWIC viewers offer one particular choice for choosing what to show from a corpus; they always display the tokens or characters immediately surrounding Q, wherever it appears in text. But there are many other possible ways to show query words in context. Figure 1.3, Figure 1.4, and Figure 1.5 suggest a few different possibilities.

In analyzing the many possible choices, we use the term **search need** to refer to the reason that users of a particular application investigate a lexical query,[13] and assume that different users of different applications will have different search needs. For instance, Chapter 7 describes a historian studying how the news media depicts Sally Ride. This historian might wish to use an application which displayed physical descriptions of the female astronaut (see Figure 1.3). But similar emphasis on appearance might be less appropriate in a tool for users trading financial securities. Noting that search need will vary by application, we then organize Part II around a central research question:

> **R2: How might lexical systems show query terms in context, to best satisfy user search need?**

In addressing R2, we argue that regardless of search need, applications will need to strive for extractions which (1) adhere to some hard or soft brevity constraint, (2) show mentions that sound natural when removed from the corpus and (3) include the most important information, based on the search needs of a particular user group. We describe these as the **brevity**, **acceptability** and **importance** criteria, and detail each criterion below.

Brevity. Any lexical application will have to contend with limited screen space and user attention. Thus we assume there is some application-specific limit to how much context can and should be presented to the user. We also assume that different applications may need to show shorter or longer mentions of Q in context. For ex-

[13]We distinguish between the user's search need and their query intent. We use search need to refer to the reason why a person turns to a particular specialized search tool. We use the IR term [44] query intent to refer to the (typically unobserved) reason a user performs a particular keyword search using a general-purpose, open-domain document search engine. We typically assume or observe a search need, but do not attempt to infer query intent. In industrial settings, inferring query intent seems to rely on heterogeneous, large-scale and proprietary datasets (e.g. user location data from the Android operating system [349]). In principle, our methods for meeting search needs could be applied in general purpose systems which first correctly infer query intent.

ample, Luhn's original KWIC index [208] allowed for 60 characters per concordance line (including the keyword). But some lexicographers report the need for need multiple lines of context to provide a syntactically-complete understanding of word use [261].[14] We use the term **context budget** to define the number of tokens, characters or pixels of context allotted per given lexical type in the corpus.[15] For instance, a KWIC view showing J separate 60-character spans containing Q would have an implied context budget of $60 * J$ characters. (Formally, we say the character length of Q counts towards the budget.)

Acceptability. A human or machine may string together any arbitrary sequence of word tokens. But some sequences will sound natural to native speakers (e.g. "Aristide fled Haiti in 2004"); others will not (e.g. "Aristide Haiti in"). In this work, applying terminology from linguistics [298], we describe sequences which sound natural as "acceptable." We then assume that users wish to review well-formed or acceptable output extractions from a lexical system,[16] with an understanding that acceptability might need to be balanced against other goals.

Based on this assumption, we introduce several methods concerned with creating well-formed mentions of Q in context. In Chapter 3, we propose an approach to identifying token spans that sound natural when removed from documents. In Chapter 4, we study how to shorten sentences mentioning Q to create well-formed output (among other goals). Finally, in Chapter 5, we show how to efficiently create natural-sounding extractions from single sentence. In each case, our goal is to identify

---

[14]Showing more context for each mention of a word makes it harder to look over many mentions of a lexical item in context. Rouse and Rouse report that one 13th century concordance showed so much context that the work ended up an "impossibly long" and "expensive failure" [289].

[15]Our term *context budget* is based on the related term *summary budget* in NLP; e.g. in Martins and Smith [224], or Lin and Bilmes [201].

[16]This assumption is based on evidence from information retrieval, which suggests that users prefer snippets which sound natural over jumbled text fragments [66]. A highly-cited textbook on creating search engines advises creating snippets which read fluidly when extracted from documents [77].

(possibly gappy) spans of text which will sound natural when they are extracted from a corpus and shown to a user.

Importance. Because lexical application designers seek to best satisfy user search need under brevity constraints, we argue that they must consider how to fill the context budget with what a given class of users will consider to be the most important information from a corpus, based on their search need. Thus, throughout this work, we present several different methods for defining and identifying important information from documents.

In Chapter 3, we offer a new supervised approach to identifying the most important descriptions of relationships between pairs of lexical query terms.[17] In Chapter 4, we assume that the user's query term is the only known important information in a sentence, and offer a method for creating well-formed shortened sentences which contain Q. In Chapter 5, we present a method for efficiently extracting salient information in sentences, based on natural supervision from newspaper headlines (which serve as a proxy for user salience judgments). Finally, in Chapter 7, we study what one particular user group considers the important information in a document or paragraph.

These efforts towards identifying and extracting important information from a corpus are closely related to work on **extractive summarization** [82] in natural language processing, which seeks to choose some small selection of natural-sounding text from a collection of one or more documents to summarize their contents. Our

[17]Our interest in summarizing relationships follows from the capacity of modern text processing, which offers new techniques for identifying and representing patterns which were not available to the authors of traditional concordances. New text processing tools like dense vectors [236, 198, 87], topic models [38] and pairwise co-occurrence statistics [262] both offer new opportunities for lexically-focused corpus exploration (e.g. the Termite system [63] or the t-SNE method [333]) and present new questions about *why* certain pairs of lexical items co-occur together in a text (e.g. why does "Slobodan Milosevic" co-occur with "air strikes"?). Note that for some techniques (e.g. t-SNE), the distance between some pair of lexical items is determined by a global objective considering all pairs of words; examining co-occurrences of a single lexical pair can build intuition, but not offer full explanation.

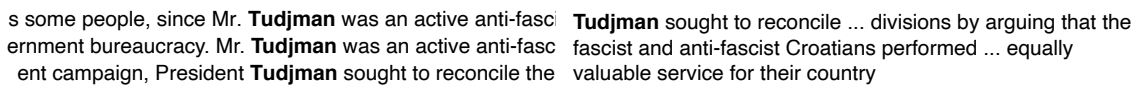


Figure 1.4: Two choices for showing the lexical item "**Tudjman**" in context, using an allotted budget of 150 characters. The left interface shows a traditional KWIC view. The right interface shows a (manually-constructed) extractive summary, optimized for traditional summarization criteria such as salience, readability and lack of repetition. This work revisits KWIC views from the perspective of extractive summarization in NLP.

efforts may be considered a form of extractive summarization, with an added and atypical hard constraint that summary output must contain the user's query.

1.3.3 How to design lexical systems for specific user groups (Part III)

In Part III of this work, we consider how to design, build and evaluate new user-facing systems for lexical corpus analysis, sometimes through applying NLP methods from Part I and Part II. Creating user-facing lexical systems presents many complex and interrelated design and engineering choices. At minimum, building any lexical application requires some consideration of how to organize a corpus lexicon based on the user's mental and linguistic models of a domain (Part I), and how to present lexical items in context based on user search need (Part II). But user-facing applications also present many other design questions, such as how many occurrences of a query to show on screen (see Chapters 6 and 7 for detailed discussions). Because we assume that is impossible to answer such questions in a principled manner without a clear understanding of the specific needs of specific user groups, in Part III we turn to research methods from human–computer interaction (HCI) and user-centered design to address a third research question:

R3: How can we build new lexical systems for specific user groups?

To help answer, we describe the design, development and evaluation of two systems designed for specific use cases: the ROOKIE system designed for journalists learning

about new topics (Chapter 6) and the ClioQuery system designed for historians and archivists performing news search (Chapter 7). We validate these approaches with qualitative and quantitative user studies, which show that Rookie helps journalists learn about new topics and that ClioQuery helps historians conduct archive research.[18]

	Lexicon	**Context**	**Users**
Rookie	Unigrams/NPs	Sentences	Journalists
ConceptMap	NPs (in pairs)	Relationship summaries	General users
ClioQuery	Unigrams	Sentence summaries	Historians/archivists

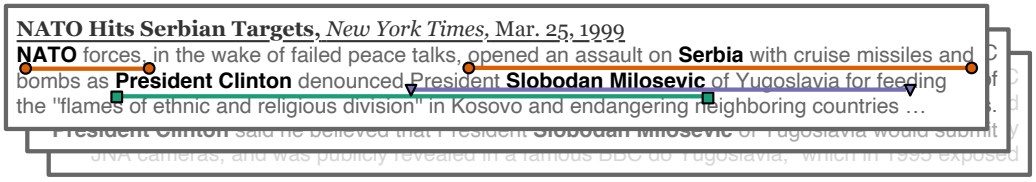
Table 1.2: This work discusses three possible interfaces for lexical corpus analysis, ConceptMap(s) (§3.5), Rookie (§6) and ClioQuery (§7). These sytems must grapple with three central questions: (1) how will the system represent the lexicon (2) how will the system select context surrounding each lexical item and (3) how do we design for the user's search needs? We abbreviate noun phrases as "NPs" above.

1.4 Related work

Our work on lexical corpus analysis lies at the intersection of three subfields of computer science: natural language processing (NLP), information retrieval (IR) and human–computer interaction (HCI). Throughout this work, we also apply methods from psycholinguistics (Chapter 4), and closely investigate specialized search behavior in journalism, history and library science (Chapters 6 and 7). Because we adopt a cross-disciplinary approach, our efforts extend and draw from a diversity of related work, beyond connections with exploratory data analysis described throughout this chapter.

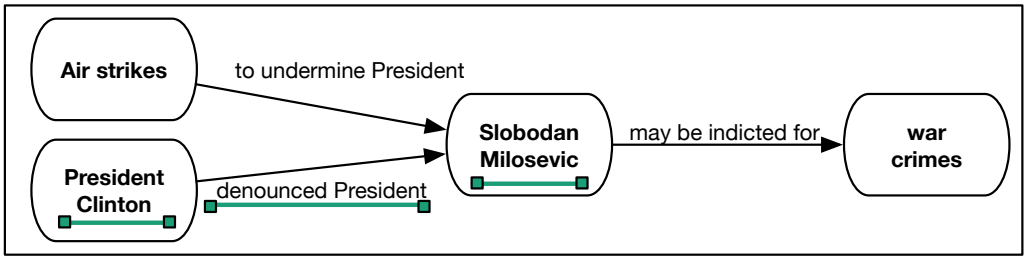
[18]Our Rookie and ClioQuery systems focus on users who are interested in lexical items as a means towards answering broader research questions. For instance, one historian examined mentions of the Iraqi city of Fallujah in U.S. newspaper editorials to better understand public perception of the second U.S.–Iraq war (Chapter 7). However, future systems might serve linguists, lexicographers, and NLP researchers studying lexical items themselves, by helping such users formalize word senses [238], explain temporal changes in lexical semantics [136], construct lexical resources [16], or identify lexical relations [288].

Figure 1.5: A news corpus (top subfigure), along with several possible systems for lexical corpus analysis (bottom three subfigures). Each system shows mentions of one or more lexical items (bolded) in context (underlined, in color). Different systems will be appropriate for different search needs.

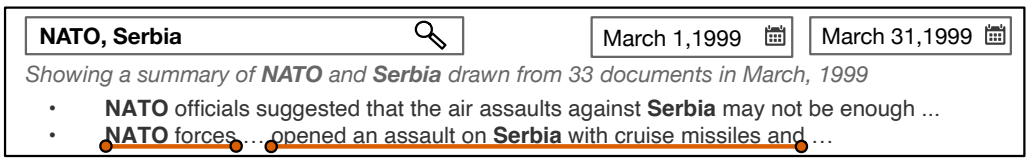


(a) A news corpus. The inferfaces below show different lexical items from this corpus in context. The first document in the corpus is shown in front of other (stacked) documents.

(b) A traditional keyword-in-context (KWIC) interface, showing three mentions of the single-word lexical item Q ="**Milosevic**" with 20 characters of surrounding context. The first line in the KWIC interface (underlined) is drawn from the first document in the corpus.



(c) A ConceptMap interface showing pairs of noun phrases (e.g. Q ="**President Clinton**"/"**Slobodan Milosevic**") in context. The system selects from mentions of both phrases in a corpus to summarize their relationship (see Chapter 3 for a comparison with relation extraction). Some mentions of Clinton and Milosevic in the corpus (shown stacked beneath the first document in the corpus) are not selected for display.



(d) A sample interface for lexical corpus analysis showing the single-word lexical items Q =["**NATO**," "**Serbia**]" in context. This interface uses text simplification techniques (see Chapters 4 and 5) to show query terms in context.

Most obviously, our efforts are closely related to extensive prior study of **user-facing text analysis systems** from HCI, such as Termite [63] and Jigsaw [126]. In Chapters 6 and 7, we discuss the strengths and weaknesses of these prior tools and other existing approaches, and compare them to our proposed lexical systems, ROOKIE and CLIOQUERY. Chapter 6 also explains why we avoid topic modeling [38] and entity tagging [100, Chp. 17], which are sometimes used in prior work.

Our research is also closely related to **information retrieval** [216], which most traditionally focuses on developing search engines that return ranked lists of documents in response to a user's query. Like these traditional keyword search tools, throughout much of this work (e.g. in Chapters 3, 5, 6 and 7), we assume that a user wishes to search for some keyword Q in a corpus. We are also mindful of the computational costs associated with indexing (Chapter 2) and presenting (Chapter 5) mentions of Q across a corpus. Similar concern with computational efficiency is common in IR.[19] Our focus on creating well-formed, shortened query mentions is also similar to efforts towards constructing **query-biased snippets** [325] for search engine results pages.

However, unlike in IR, our work typically responds to keyword input by showing users mentions of Q in context, rather than by ranking documents. (In this sense, our work is more similar to recent efforts towards **extractive search** [322] in NLP, which explores the idea of searching over patterns in dependency graphs instead of documents.) Additionally, our work allows Q to take on different grammatical types (e.g. a noun phrase pair, Chapter 3), and sometimes focuses on exploiting structured grammatical representations of sentences (e.g. dependency parses, in Chapter 5). This grammatical orientation is less common in IR, which typically represents language using a bag-of-words [216].

[19]For instance, one popular textbook details the time and space costs associated with many different ways of representing the locations of words in text [216].

As we describe in Section 1.3.2, our efforts towards showing Q in context may also be considered a particular form of query-focused, **multi-document summarization** [82, 100]. Many of the methods presented in Part II specifically build on and respond to the text summarization literature from NLP. In particular, Part II presents several new approaches to **sentence compression**, a subliterature from text summarization concerned with summarizing individual sentences (see Knight and Marcu [189]). We describe related work from sentence compression in Chapters 4 and 5. Chapter 3 presents new summarization techniques focused on summarizing relationships, in which Q is a pair of noun phrases. In Chapter 3, we include a detailed discussion explaining the ways in which this effort is both similar to and different from extensive prior work on **relation extraction** [100, Chp. 17].

1.5 Evaluation criteria

Our work on lexical corpus analysis is inspired by the traditional concordance, a computationally-efficient, highly-interpretable, highly-trustworthy and domain-independent corpus analysis tool, which has proven useful for many centuries. Because we aim to design other lexical tools with similar properties, we often evaluate our work in terms of computational efficiency, domain independence, usability, transparency and trustworthiness.[20] We explain the importance of each evaluation criterion below.

To begin, because latency is known to make user interfaces less effective [205], and because many practitioners work with limited computational resources (e.g. a single laptop), we often measure the **computational efficiency** of our proposed

[20]Of course, measuring our contributions by such criteria does not mean we fully achieve each goal. This work describes progress towards these objectives, but there is more work to be done (see Chapter 8).

techniques.[21] For instance, in Chapter 5, we evaluate the latency of our proposed VertexAddition sentence shortening technique on a CPU, to simulate performance on an ordinary laptop. While we sometimes experiment with neural network methods (e.g. in Chapter 5), our interest in efficient computation often leads us to focus on techniques which do not require GPUs. Securing special computation (e.g. setting up a cloud server) presents a barrier to quick data analysis, and some users (e.g. historians, Chapter 7) may be unable to configure or pay for special computation.

Similarly, because we hope to make tools which work reliably on new corpora, and because baseline approaches such as concordances and traditional keyword document search engines work well across different kinds of text, we also measure the **domain independence** of many methods proposed in this work. While we can't test our contributions in *every* domain, we do take care to often evaluate our efforts across heterogeneous corpora. For instance, in Chapter 2, we evaluate our proposed NPFST method using both 17th century criminal court proceedings and social media posts.[22]

Additionally, because we aim to develop tools which help groups of users in their work, we also measure the **usability** of our proposed systems via user studies. For instance, Chapter 6 includes both qualitative and quantitative evaluation of the usability and efficacy of our proposed Rookie system. Note that systems from Part III often employ NLP methods from Part I and Part II, demonstrating how our proposed NLP techniques can be applied in measurably useful applications.

Finally, evaluating by usability also forced us to implicitly evaluate and emphasize the **transparency** and **trustworthiness** of our proposed systems, particularly

[21] With exceptions [299], computational efficiency is typically overlooked in contemporary NLP. For instance, the popular SuperGLUE [338] benchmark evaluates based on F1, accuracy and exact matching rather than latency or memory use; researchers evaluated based on this common metric have little incentive to develop low-latency or memory-efficient techniques.

[22] Our interest in domain independence also leads us to sometimes avoid supervised techniques, which will perform less reliably in cases when the underlying data distribution changes from training to test time. In corpus analysis, such domain shift is all but guaranteed; the goal is to analyze *new* corpora, where the distribution is almost necessarily unknown.

during the course of developing iterative prototypes.[23] For instance, speaking with historians and archivists about early versions of ClioQuery (Chapter 7) revealed that some such experts do not trust uninterpretable summarization techniques, and speaking with journalists while developing the Rookie system (Chapter 6) revealed the importance of simple and reliable visualizations, which users could understand.[24] Therefore, we suspect that user studies focused on overall usability also implicitly measured transparency and trustworthiness; when users did not trust or understand early prototypes, they offered negative feedback.[25]

1.6 Summary of contributions and results

Our work offers a collection of systems and methods which together measurably support users undertaking lexical corpus analysis. We briefly summarize our contributions below, distinguishing between contributions towards exploratory analysis and contributions towards query-focused analysis. (These terms are first described in Section 1.2.1).

---

[23]Following prior work, we say that a system is **transparent** if the user understands the computer's output (sometimes described as interpretability [90]). We say that a system is **trustworthy** if the user believes the system has the capacity to help them undertake a task, and the integrity to adhere to domain-specific standards. Capacity and integrity are commonly applied criteria for assessing the trustworthiness of an interface in HCI (e.g. SMILY [50]), based on earlier ideas from Mayer et al. [226] in organizational management.

[24]In Part III, we thus avoid abstractive summarization methods [82] which generate output summary token spans that do not occur in input source text. Abstractive techniques are more flexible than extractive methods, but may make it harder for users to trust summary output, because the provenance of information shown a summary (i.e. where the information comes from, in underlying documents) can not be directly observed. Moreover, abstractive techniques are also prone to factual errors [191], which may undermine user trust. Both Zhang and Cranshaw [352, Section 3.3.2] and Chapter 7 of this work discuss issues related to trust and summarization in greater detail.

[25]Our experience designing trustworthy NLP systems is consistent with other work [10], which suggests that designing AI systems (e.g. using NLP) requires accommodating inevitable mistakes from probabilistic inference, including work considering the relationship between agency and automation [152, 130], and work seeking to combine the strengths of human and machine intelligence in so-called mixed-initiative [163] or AI-infused [10] user interfaces.

1.6.1 Exploratory contributions

In exploratory lexical corpus analysis, a user investigates a new body of text by examining the corpus vocabulary. Therefore, in this work we offer both tools and methods focused on providing a *type-level perspective* for exploring a corpus, including:

- The ROOKIE system (Chapter 6), which suggests noteworthy items from a corpus lexicon during a particular time period, based on a user's initial free text query. In a qualitative usability evaluation, journalism students described how this form of user interface could help provide a *"snapshot"* of a news archive in cases where *"you aren't...familiar [with] the history of the topic"* and *"want to build some context first."* We describe a quantitative evaluation of ROOKIE's query-focused contributions in Section 1.6.2.

- The NPFST method (Chapter 2), which extracts noun phrases from text for use in downstream corpus exploration. In a quantitative evaluation, we demonstrate that NPFST efficiently extracts many important and interpretable phrases (e.g. "social security") across three heterogeneous domains (news, social media and historical criminal proceedings), without returning too many unimportant or uninterpretable phrases (e.g. "and the"). Reliability across domains is important for corpus analysis, as tools for exploring bodies of documents should work well on different kinds of text. We employ companion software to NPFST in the ROOKIE system, offering evidence that the approach can be successfully applied in user-facing software.

- A new method for summarizing relationships, applied in CONCEPTMAP interfaces (Chapter 3). CONCEPTMAPs are graphical summaries displayed as directed graphs, which describe the relationships between lexical items in a corpus to help a user explore a new area. Our work on concept maps extends our

efforts towards constructing lexicons (Chapter 2) and presenting noteworthy co-occurring items from lexicons to users (Chapter 6), because concept maps can help explain *why* noteworthy phrases may co-occur in a corpus. We evaluate our proposed approach on heterogeneous datasets, to minimize the risk of overfitting to a particular domain. Although concept maps are designed for exploratory analysis, our method for summarizing relationships also achieves high yield, which helps in query-focused analysis. Our technique allows for the summarization of roughly two times more query pairs than baseline relation extraction methods, allowing the user to investigate more relationships in a corpus.

Having now briefly reviewed our chief exploratory contributions, we present our primary query-focused contributions in the next section.

1.6.2 Query-focused contributions

In query-focused lexical corpus analysis, a user investigates a specific item from the lexicon Q, by reviewing its use in context. To help with this use case, we offer tools and methods focused on providing a *token-level perspective* on a corpus, including:

- The ROOKIE system, which helps users investigate lexical queries across documents. In a quantitative evaluation, we found that by showing lexical items in context, ROOKIE helped users answer a historical question 37% faster than a traditional search interface. ROOKIE also suggests important lexical items, for corpus exploration (see Section 1.6.1).

- The CLIOQUERY system, which helps users investigate lexical queries by presenting all mentions of Q in a corpus, using NLP methods from Chapters 3 and 4. In a qualitative expert interview evaluation and field study evaluation, historians articulated the advantages of the CLIOQUERY interface over a traditional keyword search engines for investigating lexical queries.

- A low-latency, CPU-based VertexAddition algorithm for simplifying mentions of Q to fit within available screen space. In Chapter 5, we compare this linear method to an earlier approach based on an ILP objective, which requires worst-case exponential computation. Our implementation of VertexAddition runs 11x faster than the ILP, which is consistent with theoretical results. Recall that low latency is important, because lags are known to hinder interfaces for data analysis [205]. We also evaluate our approach based on well-formedness and token-level F1 score. Our proposed method achieves similar or higher scores than prior work in such automatic evaluations.

- A study of sentence simplification via clause deletion (Chapter 4), which offers (1) a new corpus of human judgments of the well-formedness of shortened sentences (2) a computational model of such judgments, and (3) a sentence simplification system based on our model of human perceptions of well-formedness. Our work in Chapter 4 is motivated by weaknesses in traditional supervision for the sentence simplification task, which implicitly defines the important information in a sentence by specifying a single "gold" shortening. By contrast, our proposed approach allows a practitioner to define the important information which should be included in an shortened sentence (e.g. inclusion of Q), which reduces dependence on domain-specific supervision (a broad goal for this work). We evaluate our model based on how well it can predict human judgments of well-formedness, measured by accuracy and ROC AUC.

We present these contributions in much greater detail in the remaining chapters of this work. We then end with a discussion of future research directions in lexical corpus analysis (Part IV).

PART I: HOW TO REPRESENT THE LEXICON

CHAPTER 2

NOUN PHRASE EXTRACTION

This chapter is adapted from *Bag of What? Simple Noun Phrase Extraction for Text Analysis* [142].

Synopsis

Traditional concordances index and display single word units called unigrams. For example, Figure 1.1a shows how a reader might use a traditional concordance to review mentions of the unigram "punishment" in the Christian Bible. Yet, some concepts in a corpus are best represented by multiword phrases, rather than single word units. For instance, the concept "social security" is better represented as a two-word phrase ("social security") than as the single words "social" and "security". A reader who wants to understand how U.S. politicians discuss "social security" needs to review mentions of the words "social" and "security" as they occur together in a body of text.

In this chapter, we offer the NPFST method to help efficiently identify such multiword units for lexical corpus analysis. NPFST uses a part-of-speech tagger and a finite state transducer to extract multiword noun phrases from a corpus. We compare this approach to alternative n-gram and parsing methods, and find that NPFST achieves lower yield and higher recall. We also observe that NPFST efficiently extracts interpretable phrases without configuration on many different kinds of English text, which is helpful for corpus analysis (Section 1.5). Open-source software based on NPFST is available on the web at: `https://github.com/slanglab/phrasemachine`.

2.1 Introduction

In exploratory lexical corpus analysis (see Section 1.2.1) a user makes sense of a corpus by reviewing items from a lexicon. Yet some concepts from a corpus are best expressed with multiword phrases (e.g. "social security"), rather than single word units (e.g. "New York" vs. "New" and "York"). Therefore, in this chapter, we consider how to extract meaningful multiword phrases, without also collecting unimportant or nonsensical extractions (e.g. "it went to"). Lexically-focused systems such as ROOKIE (Chapter 6) can then show such multiword phrases to users during exploratory corpus analysis.

Throughout this chapter, we refer to the process of identifying all multiword units in a corpus as an extraction method. We describe and define several common extraction methods, and present a new method, **NPFST**, which uses a part-of-speech tagger and a finite state transducer to efficiently extract noun phrases from a corpus. We then evaluate and compare extraction methods in terms of yield, recall, interpretability, and computational efficiency, where:

- Yield refers to the number of phrases that a method extracts; a lower yield is desirable because it requires fewer computational and human resources to process extracted phrases.[1]

- Recall refers to a method's ability to recover relevant or important phrases, as determined by a human. We measure this quality by computing the recall of named entities, which are clearly important phrases in any domain.

[1] **A note on measuring precision.** In this chapter, we use yield as an approximation of precision. Measuring precision requires defining all important phrases in a domain in order to record how many extracted phrases are in fact important. This is a tricky task which is *not* our focus in this work. Instead, we use yield as a coarse but clear-cut approximation of precision. We assume that if a method extracts many phrases (i.e. high yield), it will likely extract many unimportant phrases (i.e. low precision).

- Interpretability is a qualitative property of extracted phrases. If a person reading an extracted phrase can easily understand which concept the phrase refers to in text, we say the the phrase is interpretable. For instance, we consider the phrase "social security" to be interpretable. We consider the phrase "of the social" to be not interpretable.

- Computational efficiency refers to the wall clock speed of an extraction method. Future work might extend this definition to also include a method's memory footprint.

Overall, we show that the NPFST strategy efficiently extracts many interpretable multiword units without extracting a large number of overall phrases. Qualities like interpretability and efficiency are important for lexical corpus analysis (Section 1.5).

2.2 Background: Baseline extraction methods

We review several baseline extraction strategies below.[2]

2.2.1 n-gram methods

The simplest extraction method is **AllNGrams(K)**. This method extracts all n-grams, up to length K, from tokenized, sentence-segmented text, excluding n-grams that cross sentence boundaries. It is commonly used to extract features for text classification (e.g., Yogatama et al. [350]), but extracts fragmentary lexical units that cross sentences constituents and may not be meaningful when removed from context. For example, the text of the Affordable Care Act includes the hard-to-interpret 4-gram, "the Internet website of." This phrase plainly does not refer to any concept in

[2]We do not evaluate statistical collocation methods (e.g., Dunning [96] or Hannah and Wallach [145]), another possible approach. These methods focus on within-n-gram statistical dependence. In informal analyses, we found their recall unsatisfying for low-frequency phrases, but defer a full comparison for future work.

the domain. Additionally, although AllNGrams(K) has high recall (provided that K is sufficiently large), it suffers from a high yield (requiring more resources to process and store extracted phrases).

2.2.2 Parsing methods

Another possible extraction method uses syntax to collect phrases which correspond to sentence constituents. In this chapter, we assume that the user wishes to extract noun phrases (a particular kind of constituent) because unlike verb, prepositional, or adjectival phrases, NPs often are meaningful as standalone phrases even when stripped from their surrounding context (e.g., *[Barack Obama]*$_{NP}$ vs. *[was inaugurated in 2008]*$_{VP}$).

There are many methods for extracting NPs. Given the long history of constituent parsing research in NLP, one obvious approach is to run an off-the-shelf constituent parser and then retrieve all NP non-terminals from the trees.[3] We refer to this method as **ConstitParse**. Because the major sources of English training data, such as the Penn Treebank [221], include determiners within the NP and non-nested flat NP annotations,[4] we find that this method leads to low recall in our context (see Section 2.4). (Since modern parsers rely on these sources of training data, it is very difficult to change this behavior.)

2.2.3 Shallow parsing methods

Another possible extraction method, proposed by Justeson and Katz [178], is to use part-of-speech (POS) patterns to find and extract NPs, a form of shallow partial

[3]Another type of syntactic structure prediction is NP chunking. This produces a shallower, non-nested representation.

[4]The English Web Treebank (LDC2012T13) has some more nesting structure and OntoNotes (version 5, LDC2013T19) includes a variant of the Penn Treebank with Vadas and Curran [331]'s nested NP annotations. We look forward to the availability of constituent parsers trained on these data sources.

parsing [3]. Researchers have used this approach in a variety of different contexts [33, 116, 185, 62, 23]. Informally, extracting phrases in this manner requires defining a part-of-speech pattern and then finding all token spans which match the pattern. More formally, a pattern-based extraction method can be specified in terms of a triple of parameters: (G, K, M), where G is a grammar, K is a maximum length, and M is a matching strategy. The grammar G is a non-recursive regular expression that defines an infinite set of POS tag sequences (i.e., a regular language); the maximum length K limits the length of the extracted n-grams to $n \leq K$; while the matching strategy M specifies how to extract text spans that match the grammar.

The simplest grammar that we consider is

$$(A \mid N) * N(PD * (A \mid N) * N)*$$

defined over a coarse tag set of adjectives, nouns (both common and proper), prepositions, and determiners. We refer to this grammar as SimpleNP. The constituents that match this grammar are bare NPs (with optional PP attachments), N-bars, and names. We do not include any determiners at the root NP.

There are a number of possible matching strategies which might be used to find strings matching G of length less than K. We consider three baseline matching strategies, each of which can (in theory) be used with any G and K:

- FilterEnum enumerates all possible strings in the regular language, up to length K, as a preprocessing step. Then, at runtime, it checks whether each n-gram in the corpus is present in this enumeration. This matching strategy is simple to implement and extracts all matches up to length K, but it is computationally infeasible if K is large.

- FilterFSA compiles G into a finite-state automaton (FSA) as a preprocessing step. Then, at runtime, it checks whether each n-gram matches this FSA.

Like FilterEnum, this matching strategy extracts all matches up to length K; however, it can be inefficient if K is large.

- GreedyFSA also compiles G into an FSA, but uses a standard greedy matching approach at runtime to extract n-grams that match G. Unlike the other two matching strategies, it cannot extract overlapping or nested matches, but it can extract very long matches.[5]

In their original presentation, Justeson and Katz [178] defined a grammar that is very similar to SimpleNP and suggested using 2- and 3-grams (i.e. $K=3$). With this restriction, their grammar comprises seven unique patterns. They also proposed using FilterEnum to extract text spans that match these patterns. We refer to this method as **JK** = (SimpleNP, $K=3$, FilterEnum). Many researchers have used this method, perhaps because it is described in a popular NLP textbook by Manning and Hinrich Schütze [215].

2.3 Our proposed NPFST method

We propose a new pattern-based extraction method: **NPFST** = (FullNP, $K=\infty$, RewriteFST). In §2.3.1, we define the FullNP grammar, and in §2.3.2, we define the RewriteFST matching strategy.

2.3.1 FullNP Grammar

FullNP extends SimpleNP by adding coordination of pairs of words with the same tag (e.g., *(VB CC VB)* in *(cease and desist) order*); coordination of noun phrases; parenthetical post-modifiers (e.g., *401(k)*, which is a 4-gram because of common NLP tokenization conventions); numeric modifiers and nominals; and support for the Penn

[5] We implemented both FilterFSA and GreedyFSA using standard Python libraries—specifically, *re.match* and *re.finditer*.

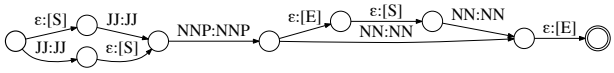


Figure 2.1: Composed rewrite lattice $L = I \circ P$ for input I = (JJ NNP NN). Five spans are retrieved during lattice traversal.

Treebank tag set, the coarse universal tag set [274], and Gimpel et al. [123]'s Twitter-specific coarse tag set. We provide the complete definition in the appendix.

2.3.2 RewriteFST Matching Strategy

RewriteFST uses a finite-state transducer (FST) to rapidly extract text spans that match G—including overlapping and nested spans. This matching strategy is a form of finite-state NLP [285], and therefore builds on an extensive body of previous work on FST algorithms and tools.

The input to RewriteFST is a POS-tagged[6] sequence of tokens I, represented as an FSA. For a simple tag sequence, this FSA is a linear chain, but, if there is uncertainty in the output of the tagger, it can be a lattice with multiple tags for each position.

The grammar G is first compiled into a phrase transducer P,[7] which takes an input sequence I and outputs the same sequence, but with pairs of start and end symbols—[S] and [E], respectively—inserted to indicate possible NPs (see Figure 2.1). At runtime, RewriteFST computes an output lattice $L = I \circ P$ using FST composition;[8] since it is non-deterministic, L includes all overlapping and nested spans, rather than just the longest match. Finally, FilterFST traverses L to find all edges with a [S]

[6]We used the ARK POS tagger for tweets [123, 265] and used Stanford CoreNLP for all other corpora [326, 217].

[7]We used `foma` [167, 29] to compile G into P. `foma` was designed for building morphological analyzers; it allows a developer to write a grammar in terms of readable production rules with intermediate categories. The rules are then compiled into a single, compact FST.

[8]We implemented the FST composition using *OpenNLP* [7] and *pyfst* (`http://pyfst.github.io/`).

Matching Strategy	All Matches?	Large K?
FilterEnum	yes	infeasible
FilterFSA	yes	can be inefficient
GreedyFSA	no	yes
RewriteFST	yes	yes

Table 2.1: RewriteFST matching strategy versus the matching strategies described in §2.2.3. Like FilterEnum and FilterFSA, RewriteFST extracts all matches up to length K; in contrast, GreedyFSA cannot extract overlapping or nested matches. Like GreedyFSA, RewriteFST can extract long matches; in contrast, FilterEnum and is infeasible and FilterFSA can be inefficient if K is large.

symbol. From each one, it performs a depth-first search to find all paths to an edge with an [E] symbol, accumulating all [S]- and [E]-delimited spans.[9]

In Table 2.1, we provide a comparison of FilterFST and the three matching strategies described in §2.2.3.

2.4 Evaluation of NPFST compared to baselines

In this section, we provide experimental results comparing NPFST to the baseline extraction methods described in §2.2 in terms of yield, recall, efficiency, and interpretability. NPFST has a low yield and high recall, and efficiently extracts highly interpretable phrases.

2.4.1 Yield and Recall

Yield refers to the number of phrases extracted by a method, while recall refers to a method's ability to recover the most relevant or important phrases, as determined by a human. Because relevance and importance are domain-specific concepts that are not easy to define, we evaluate recall using three named-entity recognition (NER)

[9]There are alternatives to this FST approach, such as a backtracking algorithm applied directly to the original grammar's FSA to retrieve all spans starting at each position in the input.

datasets (named entities are undoubtedly relevant and important phrases in a given domain):

- Mentions of ten types of entities on Twitter from the WNUT 2015 shared task [19]
- Mentions of proteins in biomedical articles from the BioNLP shared task 2011 [184]
- A synthetic data set of named entities in New York Times articles [292], identified using Stanford NER [217]

For each data set, we defined a method's yield to be the total number of spans that it extracted and a method's recall to be the percentage of the (labeled) named entity spans that were present in its list of extracted spans.[10]

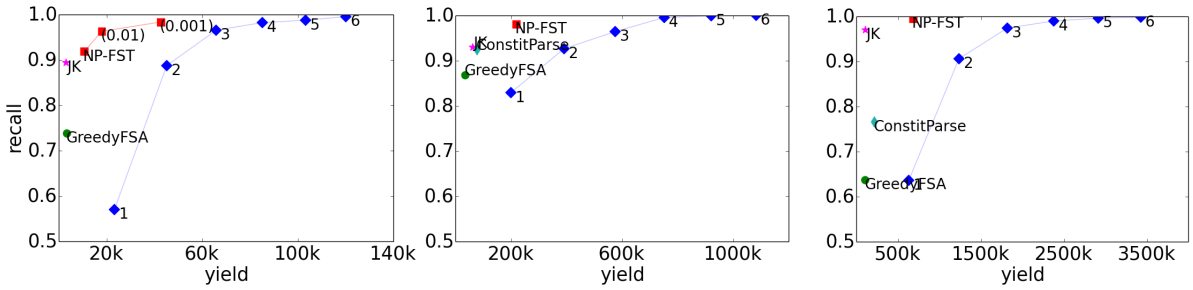
Figure 2.2 depicts recall versus yield[11] for NPFST and the following baseline methods: AllNGrams(K) with different values of K, ConstitParse,[12] JK, and (SimpleNP, $K=\infty$, GreedyFSA).

A good method should have a low yield, but high recall—i.e., the best methods are in the top-left corner of each plot. The pattern-based methods all achieved high recall, with a considerably lower yield than AllNGrams(K). ConstitParse achieved a lower yield than NPFST, but also achieved lower recall. JK performed worse than NPFST, in part because it can only extract 2- and 3-grams, and, for example, the BioNLP data set contains mentions of proteins that are as long as eleven tokens

[10]We assumed that all methods extracted all unigram spans. Because the yield and recall values for (SimpleNP, $K=3$, FilterFSA) are the same as those of JK, we omit these values from Figure 2.2. We also omit yield and recall values for (FullNP, $K=\infty$, FilterEnum) and (FullNP, $K=\infty$, FilterFSA) because they are identical to those of NPFST. Finally, we omit yield and recall values for (FullNP, $K=\infty$, GreedyFSA) because our implementation of GreedyFSA (using standard Python libraries) is too slow to use with the FullNP grammar.

[11]The WNUT data set is already tokenized; however, we accidentally re-tokenized it in our experiments. Figure 2.2 therefore only depicts yield and recall for the 1,278 (out of 1,795) tweets for which our re-tokenization matched the original tokenization.

[12]We used the Stanford CoreNLP shift–reduce parser.



(a) WNUT (b) BioNLP (c) NYT

Figure 2.2: Recall versus yield for AllNGrams(K) with $K = 1, \ldots, 6$, ConstitParse, JK, (SimpleNP, $K = \infty$, GreedyFSA), and NPFST. A good method should have a low yield, but high recall. Thus, the best methods are in the top-left corner of each plot. For visual clarity, the y-axis starts at 0.5. We omit yield and recall values for AllNGrams(K) with $K > 6$ because recall approaches an asymptote. For the WNUT data set, we omit yield and recall values for ConstitParse because there is no reliable constituent parser for tweets. As described in §2.4.1, we also show yield and recall values for NPFST run on input lattices (denoted by 0.01 and 0.001).

(e.g., "Ca2+/calmodulin-dependent protein kinase (CaMK) type IV/Gr"). Finally, (SimpleNP, $K = \infty$, GreedyFSA) performed much worse than JK because it cannot extract overlapping or nested spans.

For the WNUT data set, NPFST's recall was relatively low (91.8%). To test whether some of its false negatives were due to POS-tagging errors, we used NPFST's ability to operate on an input lattice with multiple tags for each position. Specifically, we constructed an input lattice I using the tags for each position whose posterior probability was at least t. We experimented with $t = 0.01$ and $t = 0.001$. These values increased recall to 96.2% and 98.3%, respectively, in exchange for only a slightly higher yield (lower than that of AllNGrams(2)).

2.4.2 Computational efficiency

We used ten articles from the BioNLP data set to compare the methods' preprocessing and runtime costs. Table 2.2 contains timing results[13] for AllNGrams(∞), ConstitParse, JK, (SimpleNP, $K=3$, FilterFSA), and (SimpleNP, $K=\infty$, GreedyFSA), and NPFST. We omit results for (FullNP, $K=\infty$, FilterEnum), (FullNP, $K=\infty$, FilterFSA), and (FullNP, $K=\infty$, GreedyFSA) because they are too slow to compete with the other methods.

POS tagging is about twenty times faster than parsing, which is helpful for users in fields like social science and journalism who may not have high-performance computers. (As described in Section (§1.2) to this work, throughout this work we strive for efficient computational techniques for lexical corpus analysis.) NPFST is slightly slower than the simpler pattern-based methods; however, 80% of its time is spent constructing the input I and traversing the output lattice L, both of which are implemented in Python and could be made faster.

Method	Time
AllNGrams(∞)	44.4 ms
ConstitParse	825.3 ms
JK	45.3 ms
(SimpleNP, $K=3$, FilterFSA)	46.43 ms
(SimpleNP, $K=\infty$, GreedyFSA)	39.34 ms
NPFST	82.2 ms

Table 2.2: Timing results for AllNGrams(∞), ConstitParse, JK, (SimpleNP, $K=3$, FilterFSA), (SimpleNP, $K=\infty$, GreedyFSA), and NPFST on ten articles from the BioNLP data set; AllNGrams(∞) is equivalent to AllNGrams(56) in this context. The pattern-based methods' times include POS tagging (37.1 ms), while ConstitParse's time includes parsing (748.0 ms).

Data Set	Method	Ranked List
Twitter	unigrams	snow, #tcot, al, dc, gore
	JK	al gore's, snake oil science, snow in dc, mine safety
	NPFST	al gore's, snake oil science, 15 months, snow in dc, *bunch of snake oil science
Old Bailey	unigrams	jacques, goodridge, rust, prisoner, sawtell
	ConsitParse	the prisoner, the warden, the draught, the fleet, the house
	JK	middlesex jury, public house, warrant of attorney, baron perryn, justice grose
	NPFST	middlesex jury, public house, warrant of attorney, baron perryn, *middlesex jury before lord loughborough
NYT	unigrams	will, united, one, government, new
	ConstitParse	he united states, the government, the agreement, the president, the white house
	JK	united states, united nations, white house, health care, prime minister
	NPFST	united states, united nations, white house, health care, *secretary of state warren christopher

Table 2.3: Ranked lists of representative terms for unigrams, ConstitParse, JK, and NPFST. For NPSFT, we include the highest-ranked phrase of length four or more (on its own line, denoted by *) in order to highlight the kinds of longer phrases that JK is unable to extract. For the Twitter data set, we omit results for ConstitParse because there is no reliable constituent parser for tweets.

2.4.3 Interpretability

To assess the interpretability of the phrases extracted by each method, we used three existing datasets: tweets about climate change, written by (manually identified) climate deniers;[14] transcripts from criminal trials at the Old Bailey court in London during the 18th century;[15] and New York Times articles from September, 1993. For each data set, we extracted phrases using ConstitParse, JK, and NPFST and produced a list of top terms for each method, ranked by count. To create a top terms list for the NPFST method, we merged overlapping phrases (e.g. "social security" and "security act") using the term-merging algorithm described in the next section. Ta-

[13]We used Python's *timeit* module.

[14]`https://www.crowdflower.com/data/sentiment-analysis-global-warmingclimate-change-2/`

[15]`http://www.oldbaileyonline.org/`

ble 2.3 contains these lists, demonstrating that NPFST produces highly interpretable phrases.[16]

Additional details: Merging related terms

As described in §2.3.2, NPFST extracts overlapping and nested spans. For example, when run on a data set of congressional bills about crime, NPFST extracted the phrase "omnibus crime control and safe streets act," as well as the nested phrases "crime control" and "safe streets act." Although this behavior is generally desirable, it can also lead to repetition when returning a list of high-ranking terms.

We therefore outline an high-level algorithm for merging high-ranked terms. The input to our algorithm is a small ordered list L, ordered by some term-level importance criterion. (In §2.4.3, L is a ranked list of the most frequent terms in the corpus.) The output is a list of C top terms, where C is less than the length of L. To merge nested phrases in L, our algorithm iterates through L, starting with the highest-ranked term. At each iteration, the algorithm either places some term t in L in a new (singleton) cluster or adds t to an existing cluster, based on some user-defined similarity criterion. (In §2.4.3, we add t to some cluster if t is a substring of some t' in the cluster, or if some t' in the cluster is a substring of t.) The algorithm stops iteration once it has generated C clusters. The algorithm then selects a single term to represent each cluster according to some user-defined criterion. (In §2.4.3, we select the longest term in each cluster, by character length.) Finally, the algorithm returns the clusters' representative terms to return a list of C top terms. By starting with the highest-ranked term and terminating after forming C clusters, this algorithm avoids

[16]We excluded domain-specific stopwords and any phrases that contained them. For example, for the tweets, we excluded phrases that contained "climate" and "warming." For the Old Bailey transcripts, we excluded phrases that contained "st." or "mr." (e.g., "st. john" or "mr. white"). We also used a regular expression to exclude apparent abbreviated names (e.g., "b. smith") and used a stopword list to exclude dates like "5 of february." For the New York Times articles, we excluded phrases that contained "said."

the quadratic cost of comparing all pairs of top terms from the set of extracted lexical items.

2.5 Conclusion

In this chapter, we present NPFST, a new method that efficiently extracts noun phrases for inclusion in a corpus lexicon. We then compare our method to several baselines, demonstrating that our proposed approach has low yield, high recall, and efficiently extracts interpretable phrases across many different kinds of English text. NPFST is particularly appropriate for lexical corpus analysis, as the method does not require the user to undertake specialized configuration or annotation to investigate new corpora. We apply NPFST for user-facing exploratory lexical analysis in Chapter 6.

Open-source companion software is available at:
`https://github.com/slanglab/phrasemachine`

2.6 Appendix

2.6.1 FullNP Grammar

The following `foma` grammar defines the rewrite phrase transducer P:

```
# POS tag categories. "Coarse" refer to the Petrov Univeral tag set.
# We directly use PTB tags, but for Twitter, we assume they've been
# preprocessed to coarse tags.
# CD is intentionally under both Adj and Noun.
define Adj1     [JJ | JJR | JJS | CD | CoarseADJ];
define Det1     [DT | CoarseDET];
define Prep1    [IN | TO | CoarseADP];
define Adv1     [RB | RBR | RBS | CoarseADV];
# Note that Twitter and coarse tags subsume some of this under VERB.
define VerbMod1 [Adv1 | RP | MD | CoarsePRT];
# PTB FW goes to CoarseX, but we're excluding CoarseX since for Gimpel et al.'s
# Twitter tags, that's usually non-constituent-participating things like URLs.
define Noun     [NN | NNS | NNP | NNPS | FW | CD | CoarseNOUN | CoarseNUM];
define Verb     [VB | VBD | VBG | VBN | VBP | VBZ | CoarseVERB];
define AnyPOS   [O  |  Adj1|Det1|Prep1|Adv1|VerbMod1|Noun|Verb |
    CoarseDOT|CoarseADJ|CoarseADP|CoarseADV|CoarseCONJ|CoarseDET|
    CoarseNOUN|CoarseNUM|CoarsePRON|CoarsePRT|CoarseVERB|CoarseX
]
define Lparen   ["-LRB-" | "-LSB-" | "-LCB-"];  # Twitter doesnt have this.
define Rparen   ["-RRB-" | "-RSB-" | "-RCB-"];
# Ideally, auxiliary verbs would be VerbMod, but PTB gives them VB* tags.

# single-word coordinations
define Adj      Adj1 [CC Adj1]*;
define Det      Det1 [CC Det1]*;
define Adv      Adv1 [CC Adv1]*;
define Prep     Prep1 [CC Prep1]*;
define VerbMod  VerbMod1 [CC VerbMod1]*;

# NP (and thus BaseNP) have to be able to stand on their own.  They are not
# allowed to start with a determiner, since it's usually extraneous for our
# purposes. But when we want an NP right of something, we need to allow
# optional determiners since they're in between.
define BaseNP   [Adj|Noun]* Noun;
define PP       Prep+ [Det|Adj]* BaseNP;
define ParenP   Lparen AnyPOS^{1,50} Rparen;
define NP1      BaseNP [PP | ParenP]*;
define NP       NP1 [CC [Det|Adj]* NP1]*;

regex NP -> START ... END;
write att compiled_fsts/NP.attfoma
```

PART II: HOW TO SHOW LEXICAL ITEMS IN CONTEXT

CHAPTER 3

RELATIONSHIP SUMMARIZATION

This chapter is adapted from *Relational Summarization for Corpus Analysis* [139] and *Summarizing Relationships for Interactive Concept Map Browsers* [144].

Synopsis

Early paper concordances showed readers how single words were used in context (e.g. Q="punishment", *Figure* 1.1a). But today, computational techniques for identifying co-occurring lexical items allow for new forms of lexically-focused corpus investigation (e.g. Termite [63], which shows users co-occurring lexical items, using a topic model of text [38]). Identifying and presenting related terms suggests a clear next question: why do any given pair of lexical items often appear together in a corpus?

To help answer, we define the task of relationship summarization, in which the goal is to show two co-occurring lexical items in context, in a way that expresses a summary of their relationship. We then make progress in solving the task by (1) offering and evaluating a new method for finding natural language sentences which express relationships between pairs of lexical items and then (2) applying the method towards constructing summaries for ConceptMap interfaces, which are graphical corpus overviews designed to help with lexically-oriented corpus exploration (see Section 1.2.1). In automatic evaluation, we show that our method for identifying descriptions of relationships between lexical items allows for summarization of more than two

times more query pairs than baseline relation extractors, while returning measurably more readable output across heterogenous corpora.

3.1 Introduction

Computational text analysis offers many methods for identifying co-occurring lexical items in a corpus, such as word vectors [236, 198, 87], topic models [38] and pairwise co-occurrence statistics [262]. These methods allow for new forms of lexically-focused corpus exploration (e.g. ROOKIE, from Chapter 6) organized around showing statistically-related words. Yet identifying such co-occurring terms raises a natural question: what is the nature of the relationship between items which often appear together in the text? For instance, why does "Aristide" often co-occur with "liberation theology" in the Haiti corpus from Chapter 6?

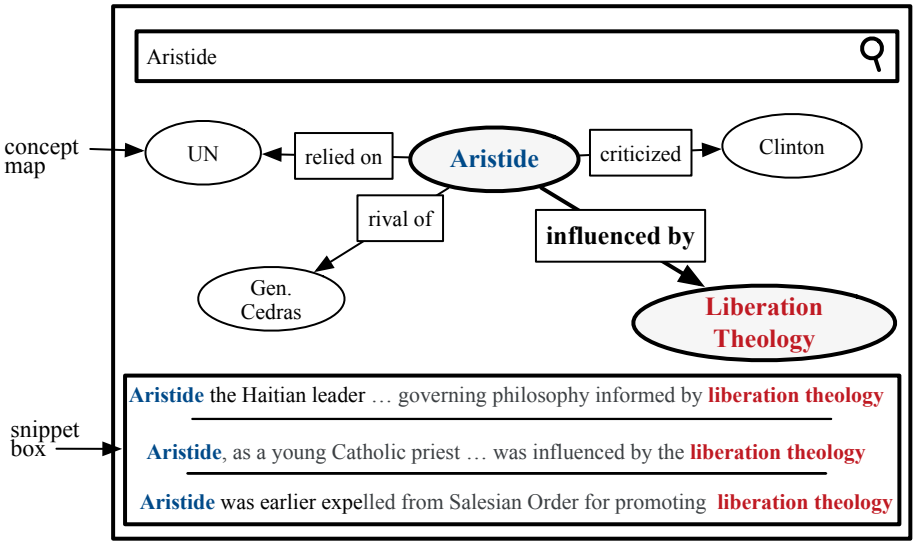


Figure 3.1: An example interface which uses relationship summarization to help explain lexical co-occurrence. The user has queried for the entity "Aristide". The interface shows a CONCEPTMAP (top), displaying short summaries of relationships between Aristide and other concepts (noun phrases, Chapter 2) which frequently co-occur with Aristide in the corpus. The user has drilled down to see a more detailed summary of Aristide's relationship with liberation theology.

This chapter seeks to help users understand such relationships by defining and taking steps to solve the problem of **relationship summarization**. For instance, the ConceptMap browser in Figure 3.1 applies relationship summarization to explain why the lexical item "Aristide" often co-occurs with the lexical item "Liberation Theology" in a corpus of *New York Times* articles mentioning "Haiti" (see §6). Unlike previous efforts at summarizing relationships [103], in this chapter, we focus on answering lexical user queries about the connections between two particular terms, without referencing a knowledge graph [336].[1] In order to answer such queries we:

- Formally define the relationship summarization problem (§3.2), which we divide into two subtasks: the **candidate set generation** task and the **summary construction** task.
- Offer a new method for the candidate set generation task based on naturally-occurring supervision from news headlines (§3.4). We find that this method returns more readable output and allows for the summarization of more query pairs than baseline relation extraction techniques (§3.4.2).
- Offer a new method for the summary construction task, for use in ConceptMap browsers. Our approach is based on a modeling a new annotated dataset, assembled for this task (§3.5).
- Analyze and discuss the summary construction task for future work (§3.6), arguing that different summarization techniques are likely most appropriate for different pairs of lexical query items.

Together, this effort supports new natural language processing methods for corpus analysis, focused on analyzing relationships between lexical items in text.

[1]Relational summaries are intended for general-purpose corpus analysis. Existing knowledge bases do not cover topics discussed in many corpora, such as historical court records [158]. Therefore, our approach does not assume access to a knowledge base.

3.2 Defining relationship summarization

Relationship summarization attempts to summarize the relationship between two lexical query terms, (t_1) and (t_2), in a corpus. In this chapter, we assume that (t_1) and (t_2) are noun phrases, a syntactic category which encompasses both traditional named entities like people and places, as well as less concrete, but important, entities and concepts like "liberation theology" (§2). We sometimes describe (t_1) and (t_2) as "lexical query terms" or just "terms" (for brevity). In the terminology of lexical corpus analysis, (t_1) and (t_2) are a 2-place query Q, consisting of two NPs.

We define the set of all sentences within a corpus which contain (t_1) and (t_2) as the **mention set**. A relationship summary is a synopsis of the mention set, which consists of K relation statements, each displayed on its own line. Relation statements are natural language expressions which begin with (t_1) and end with (t_2), and occur at least once as a token span in the corpus. Hence, each relation statement is a span of tokens within some sentence in the mention set; thus, we say that relationship summarization displays (t_1) and (t_2) in context (i.e. a span of token from a sentence in the corpus).

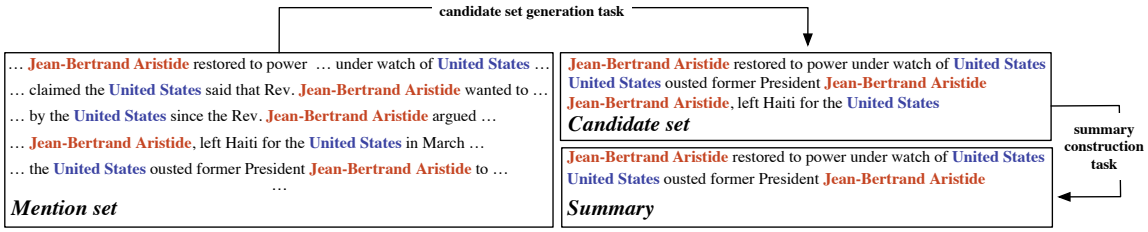


Figure 3.2: A relationship summary is a synopsis of all sentences which mention two lexical query terms, denoted (t_1) and (t_2). We refer to such sentences as a *mention set*. In the figure above (t_1) is **Jean-Bertrand Aristide** and (t_2) is **United States**. To create a summary first requires identifying all statements in the mention set which coherently describe some relationship between (t_1) and (t_2). This **candidate set generation task** is a prerequisite for the subsequent **summary construction task**, which selects the top K candidates to create a summary. In this work, we offer a method for the first task and show how the second task will likely require a diversity of summarization techniques (§3.6).

We use specific terminology and notation to describe relation statements. We refer to the span of tokens in between $\boldsymbol{(t_1)}$ and $\boldsymbol{(t_2)}$ as a *relation phrase*, and we use the notation $\boldsymbol{(t_1)}$ r $\boldsymbol{(t_2)}$ to denote a relation statement, which consists of $\boldsymbol{(t_1)}$, $\boldsymbol{(t_2)}$ and a relation phrase r. In the relation statement, "**Aristide** fled **Haiti**", r is the token "fled", $\boldsymbol{(t_1)}$ is the token **Aristide**, and $\boldsymbol{(t_2)}$ is the token **Haiti**.[2]

Relation statements, which are strings intended for human readers, are similar to the 3-tuples, "*relations*", from prior work on information extraction [25]. However, in this work, we show that the assumptions underlying the extraction of 3-tuple "*relations*" for machines (§3.3) leads to poor performance in summarizing mention sets for people (§3.4.2).

In this chapter, we present a strictly extractive method for generating relation statements; each relation statement must be constructed by deleting tokens from some context sentence in the mention set.[3] Some relation statements constructed by deleting tokens from a sentence make sense; others do not. We refer to any $\boldsymbol{(t_1)}$ r $\boldsymbol{(t_2)}$ which sounds well-formed to a human reader as **acceptable**.[4] Table 3.1 shows examples of acceptable and unacceptable relation statements, constructed by token deletion.

As in traditional summarization [82], a good relationship summary should (i) be readable, (ii) include the most important aspects of the relationship between $\boldsymbol{(t_1)}$ and $\boldsymbol{(t_2)}$, (iii) avoid redundancy, and (iv) cover the full diversity of topics in the mention set.

More concretely, relational summaries might be presented with different kinds of user interfaces. In cases where a user seeks to browse many relationships, a summary

---

[2]In this case, **Aristide** and **Haiti** are single-word NPs.

[3]In subsequent studies of relation extractors (§3.4.2), we allow extractors to lightly introduce new tokens, such as adding the word "is" in relations expressed as noun phrases.

[4]Linguists sometimes use the term "acceptability" to refer to human judgements of the well-formedness of utterance. See Sprouse and Schütze [314] for an overview.

s_1	Aristide fled Haiti ~~in 2004~~. (t_1) r (t_2)
s_2	~~For instance~~ Bush told Aristide ~~to leave~~. (t_1) r (t_2)

Table 3.1: Two relation statements constructed by deleting tokens from source sentences, s_1 and s_2. The relation statement extracted from s_1 is well-formed; the statement extracted from s_2 is not.

might be displayed as a ConceptMap [103], where the two terms are vertexes in a directed graph and their relationship is printed along the edge label between them. In cases where user wants to investigate a specific relationship, a relationship summary might be displayed as a *snippet box*: a short list of sentences which begin and end with the two lexical items. Figure 3.1 shows a snippet box and ConceptMap. In a snippet box, both the number of lines in the summary and the length of the lines in the summary is longer than in a ConceptMap.

3.2.1 Tasks: candidate set generation and summary construction

In this chapter, we approach the problem of constructing a relationship summary in two steps. The first step is to identify the set of all possible well-formed relation statements, called the **candidate set** and denoted $\mathcal{C}$. We refer to this task of identifying all acceptable relation statements as the **candidate set generation task**. Identifying a candidate set presents a subsequent problem of choosing the best collection of K relation statements from $\mathcal{C}$ to create a summary. We refer to this second step as the **summary construction task**. In this chapter, we propose and evaluate methods for the candidate set generation task and the summary construction task.

3.3 Related work

Relationship summarization intersects with a diversity of prior work from natural language processing, including work on **relation extraction**, **summarization** and **sentence compression**.

Traditionally, the goal of **relation extraction** is to cull structured facts for knowledge databases from unstructured text. Often, such facts take the form of a 3-tuple which defines a relationship between two arguments, such as (arg1="Angela Merkel", rel=met with, arg2="Theresa May"). If extractors do not make use of a predefined schema, the task of finding relations is called Open Information Extraction (OpenIE). OpenIE systems[5] provide an off-the-shelf method for generating a candidate set for a relationship summary. Their output can easily be linearized to $\boldsymbol{(t_1)}$ r $\boldsymbol{(t_2)}$ statements by simply concatenating the three arguments of the 3-tuple to form a string.

However, we find that the recall of relation extractors is often too low to summarize many mention sets. (We measure this disadvantage extensively in section §3.4.4.) One reason for their poor performance might be that extractors have goals and assumptions which are poorly suited to relationship summarization. In relation extraction, the aim is to find relation strings that recur for many different entity pairs, which allows such systems to build knowledge databases. For instance, relation extraction might be used to build tables of world leaders who rel="met with" other world leaders in order to analyze international politics. From this perspective, long, sparse, heterogenous and detailed relation strings which might apply only to a pair of specific arguments are undesirable, as they make it difficult to find general patterns across many different entity pairs. For example, the influential ReVerb OpenIE system [102] excludes "overly-specific relation phrases" which apply only to two entities.

[5]There are many available OpenIE systems; Stanovsky and Dagan [316] offer an overview.

One way to help ensure that relations generalize across entity pairs is to strive for arguments which are as short as possible, a common goal in OpenIE [316].[6]

Our method for generating a candidate set is closer to approaches from **sentence compression** [189, 67, 107, 109], an NLP task which seeks to make a source sentence shorter while preserving the most important information and producing readable output. We show that our compression-based approach allows us to achieve higher readability than off-the-shelf relation extractors (§3.4.2).

Sentence compression is often used in traditional extractive **summarization** to make more efficient use of a budgeted summary length. We discuss summarization further later in this Chapter (§3.5, §3.6), where we consider how existing work on summarization might be applied to the problem of selecting K statements from the candidate set.

Finally, in this chapter, we describe the collection of a new supervised dataset for summarizing relationships in ConceptMap interfaces (§3.5). Our dataset has a different focus from an existing ConceptMap resource from Falke and Gurevych [103], who seek to create the best overall ConceptMap for a given *topic*. In our case, we seek to find the best summary relationship for a given *relationship* which a user might wish to investigate. Therefore, unlike Falke and Gurevych [103], our dataset includes labels for the most readable and informative statement describing the relationship between any $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$ query pair.

3.4 An approach to the candidate set generation task

This section proposes and evaluates a method for candidate set generation.

---

[6]Methods from the relation extraction literature which seek to deduce facts from extracted relations, such as Riedel et al. [284], might also help identify useful summaries in future work. Relations which imply that other relations are true might make good summaries.

Unacceptable statement	**Auburn police** are investigating the death of a **Tuskegee** ~~woman who died ...~~
Acceptable statement	~~Drug firm~~ **Glenmark** has opened its new facility in **Argentina** ~~which would ...~~

Table 3.2: Token spans between named entities which are selected at random usually do not express relationships between the two entities. We sample such spans at random (top) to generate examples of relation statements that are not well-formed. Gold compressions from the Filippova and Altun [107] sentence compression corpus that start and end with a named entitiy serve as examples of well-formed relation statements (bottom). In this figure, named entites are shown in bold.

3.4.1 Candidate generation using headline-based supervision

Traditionally, relation extraction begins with a fixed notion of what constitutes a desirable "relation" between two arguments, defined by a predefined schema, a syntactic template [102], or a collection of seed examples [12]. The relation extraction task is then to correctly identify spans in which arguments are joined by a relation.

The relationship summarization problem is somewhat different: we begin with a pair of lexical query terms, $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$, and we wish to learn the nature of their relationship. Therefore, any well-formed statement which describes any relationship between the two query terms Q is potentially of interest, even if it does not match prior expectations of what constitutes a relation.

We thus approach the candidate set generation task as a specialized form of sentence compression: we attempt to predict if a sentence from the text can be compressed to a well-formed statement form $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$. Table 3.2 shows examples of sentences which can and cannot be shortened to this form.

We use gold standard sentence–compression pairs from the Filippova and Altun [107] benchmark sentence compression dataset constructed automatically from the headlines of news articles to supervise this prediction.[7] In this corpus, gold standard compressions must be acceptable sentences. Therefore, compressions from the dataset which happen to begin and end with a named entity, once extracted from source

[7] `https://github.com/google-research-datasets/sentence-compression`

sentences, can serve as positive examples of acceptable relation statements. On the other hand, randomly chosen spans of the form $\mathbf{(t_1)}$ r $\mathbf{(t_2)}$, which happen to arise in source sentences, are very often not well-formed as standalone sentences. These randomly sampled spans can serve as examples of unacceptable relation statements. We then predict well-formedness with supervision from known gold well-formed and sampled, presumed garbled (not well-formed) examples.[8]

Filtering the original dataset in this manner[9] yields 17,529 positive and 30,266 negative sentences. We then downsample negative training examples to create two balanced classes of equal size, and use 81% of data for training, 9% for validation and the remaining 10% for testing.

$p(c = 1\|s,$ $\mathbf{(t_1)}$ r $\mathbf{(t_2)})$	$\mathbf{(t_1)}$ r $\mathbf{(t_2)}$
.005	**Jean-Bertrand Aristide** that the **United States**
.010	**United States** since the Rev. **Jean-Bertrand Aristide**
...	...
.894	**United States** ousted former President **Jean-Bertrand Aristide**
.976	**Jean-Bertrand Aristide**, left Haiti for the **United States**

Table 3.3: Highest and lowest well-formedness predictions from the set **United States – Jean-Bertrand Aristide**

Let $p(wf = 1 \mid s,$ $\mathbf{(t_1)}$ r $\mathbf{(t_2)})$ indicate the probability that a span of form $\mathbf{(t_1)}$ r $\mathbf{(t_2)}$ extracted from sentence s is well-formed. We model $p(c = 1|s,$ $\mathbf{(t_1)}$ r $\mathbf{(t_2)})$ using

[8]We manually inspect 100 negative examples, selected at random, and find that roughly 80% are in fact garbled.

[9]We also exclude randomly chosen spans which happen to encompass the entire source sentence and exclude randomly chosen spans where $\mathbf{(t_1)}$ and $\mathbf{(t_2)}$ are joined by only edges of type compound in the dependency graph of the compression (e.g. "Coup leader Cedras ..."). We use CoreNLP version 3.8 to extract *enhanced++* Universal Dependencies [217, 296, 259]. We also filter positive and negative examples where the span between $\mathbf{(t_1)}$ and $\mathbf{(t_2)}$ is longer than J=75 characters, to simulate a space constraint in a user interface. Finally, we remove all punctuation from the end of the sentence for both positive and negative examples because all gold positive compressions end in punctuation marks. For positive examples, if the compressed version of a sentence deletes tokens between t_1 and t_2, we replace the span between t_1 and t_2 in the source sentence with the compression.

logistic regression, with features based on the position of part-of-speech tags and dependency edges in s. Specifically, each sentence in the filtered dataset contains a span of the form $(\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$. We refer to the tokens in this span as *in the compression* because a user would see these tokens in a relation statement compressed from s. Each sentence also contains spans of tokens which are *outside of the compression* because they are deleted from the original source sentence to create a relation statement. Table 3.2 displays examples.

Our feature vector records the counts of how many times each part-of-speech tag in the tagset occurs in the compression and also independently records the counts of how many times each part-of-speech tag occurs out of the compression. We refer to the count of each part-of-speech tag in the compression and the count of each part-of-speech tag out of the compression as $\boldsymbol{\Phi}$. We also count the occurrence of each possible dependency edge label in the compression, and the count of each possible dependency edge label out of the compression. If a label's dependent lies in the compression, we consider the label in the compression.[10] We refer to these dependency edge counts as $\boldsymbol{\Psi}$. Our final feature vector, $\boldsymbol{\Omega}$, is defined as the concatenation of $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$.

Features	Accuracy
$\boldsymbol{\Phi}$ (pos)	.858
$\boldsymbol{\Psi}$ (deps)	.892
$\boldsymbol{\Omega}$ (deps & pos)	.896

Table 3.4: Test accuracies

We implement our model with *scikit-learn* [272] and manually tune the inverse regularization constant to the setting, $c = 1$, which achieves the highest accuracy on the validation set. For evaluation, a sentence is presumed well-formed if $p(c = 1|s, (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})) > .5$. Using the feature vector $\boldsymbol{\Omega}$ we achieve an accuracy of .896 on the

[10]Enhanced dependencies allow for a token to have more than one incoming edge (i.e., multiple parents). If there is more than one incoming edge, we pick an edge at random.

test set. We also present results using only the $\mathbf{\Psi}$ and $\mathbf{\Phi}$ features (Table 3.4) because reliable dependency parses are not available in some settings [40, 22].

Two limitations of this approach suggest areas for future work. First, in some cases, the relationship between $\boldsymbol{(t_1)}$ and $\boldsymbol{(t_2)}$ might not be expressed in the form, $\boldsymbol{(t_1)}$ r $\boldsymbol{(t_2)}$, as in "**Russia** and **France** signed an agreement". In order to generate candidate relation statements it would be helpful to lightly rewrite the sentence, as in "**Russia** signed an agreement with **France**". Additionally, a sentence might express a relationship between two lexical items but be too long to display on a ConceptMap or a snippet box. In these cases, it would be helpful to compress the sentence to create a more concise relation statement.

3.4.2 Evaluating headline-based candidate set generation

Any relationship summarization system should deliver a high-quality summary when a user queries for Q, a tuple of two lexical items. Therefore, ideally, a system should generate the largest possible candidate set, without selecting any garbled relation statements as candidates. We thus evaluate our query-focused generation method in terms of both readability and yield (total relation statements recalled). Our method generates three times more relation statements than OpenIE systems, allowing for summarization of two times more query pairs. We also achieve higher scores in a test of human acceptability judgements (Table 3.5).

3.4.3 Creating a corpus

We evaluate our compression-based method for generating candidate sets against two relation extractor baselines on two very different corpora: (1) all comments from the "relationships"[11] subreddit from June, 2015 – September, 2017[12] and (2) a col-

[11]"relationships" refers to interpersonal relationships

[12]`https://medium.com/@jason_82699/pushshift-reddit-api-md-c2d70745c270`

lection of *New York Times* articles from 1987 to 2007 which mention the country "Haiti" [292]. For each corpus, we first find a collection of multi-word phrases using the `phrasemachine` package (§2) which extracts all multi-word, noun phrase lexical items from the corpus. We require all relation statements be less than or equal to $J = 75$ characters, which excludes overly verbose relation statements which are unsuitable for many user interfaces.

3.4.4 A yield evaluation

After extracting all multiword phrases with `phrasemachine`, we determine the top 100 noun phrases, by count. We then examine all possible combinations of two top terms and record which pairs have a non-empty mention set of sentences which mention two lexical query terms (§3.2). We examine all mention sets because an investigator should be able to investigate any entity she chooses while analyzing a corpus.

We compare the yield of our query-focused generation method to off-the-shelf relation extractors. Such extractors generate 3-tuples from each mention set. Some but not all of those 3-tuples might have one argument which is equal to $\boldsymbol{(t_1)}$ and another argument which is equal to $\boldsymbol{(t_2)}$. Each such 3-tuple can be linearized into a string of the form $\boldsymbol{(t_1)}$ r $\boldsymbol{(t_2)}$ which will generate a candidate set.

More precisely, we identify the 3-tuples which an OpenIE system extracts from a mention set such that exactly one argument from the triple is equal[13] to $\boldsymbol{(t_1)}$ and exactly one argument from the triple is equal to $\boldsymbol{(t_2)}$. We refer to these 3-tuples as "matching". We then count (1) the total number of mention sets which contain

[13]Note that OpenIE systems might not extract the literal string $\boldsymbol{(t_1)}$ or $\boldsymbol{(t_2)}$ as arguments. For instance, if $\boldsymbol{(t_1)}$ is "Merkel" the OpenIE system might extract the argument "Angela Merkel". If some term and some argument from a relational triple share the same head token in the dependency parse of the sentence we say that they are equal. Falke and Gurevych [104] employ a similar equality criterion. We tokenize with CoreNLP. In extremely rare cases, tokenization mismatches between CoreNLP and ClausIE make it impossible to apply this criterion.

at least one matching 3-tuple and (2) the total number matching 3-tuples across all mention sets. We refer to such counts as the *yield* of a candidate generation system.

We measure the yield of Stanford OpenIE [12] and ClausIE [85] on the *New York Times* and *Reddit* corpora, and compare each system to our compression-based approach (§3.4).[14] We measure these two relation extractors because Stanford OpenIE is included with the popular CoreNLP software and ClausIE achieves the highest recall in two systematic studies of relation extractors [316, 353].

We find that using extractors to generate candidate sets achieves a low yield (total number of extracted relations), which is undesirable both because it limits the number of mention sets which may be summarized and generates fewer relation statements from which to select an optimal relationship summary. Additionally, we find that, for the great majority of sentences, relation extractors do not extract any relations between $\boldsymbol{(t_1)}$ and $\boldsymbol{(t_2)}$. Moreover, for many mention sets, the number of relations extracted with off-the-shelf systems is often zero. We show these results in Table 3.5.

This suggests that although relation summarization is superficially similar to relation extraction, off-the-shelf extractors are poor tools for creating textual units to summarize mention sets. Very often, two terms are related to each other in ways which are simply not captured by relation extractors.

3.4.5 An evaluation with human acceptability judgments

Our compression-based method achieves higher yield than off-the-shelf relation extractors. However, because all sentences in a mention set include $\boldsymbol{(t_1)}$ and $\boldsymbol{(t_2)}$, it is always possible to generate a maximally large candidate set by simply extracting all spans between $\boldsymbol{(t_1)}$ and $\boldsymbol{(t_2)}$ from the mention set, regardless of whether such relation statements are well-formed. We examine if gains in yield come at the expense of

[14] For our compression-based approach, we count all cases where $p(wf = 1 \mid s, \boldsymbol{(t_1)}\ r\ \boldsymbol{(t_2)}) > .5$ as extracting a relation statement.

	Yield				Well-formedness	
	Total non-empty pairs		Total rel. stmts.		Mean judgment	
	Haiti	Reddit	Haiti	Reddit	Haiti	Reddit
ClauseIE	128	1,121	279	3,949	3.67	3.71
StanfordOIE	443	1,488	972	5,605	3.69	2.97
This work	**739**	**3,766**	**2,954**	**21,495**	**3.94**	**3.85**
Upper bound	2,472	4,496	43,051	123,760	Range: 1-5	

Table 3.5: We compare Stanford OpenIE, ClausIE and our headline-based compression method for the candidate set generation task on two different corpora (Haiti articles from *New York Times*, and the *Reddit* relationships forum) in terms of (1) how many entity pairs have a non-empty candidate set, (2) how many total relation statements are generated, and (3) the average human judgment of well-formedness (§3.4.5). For yield measures, the upper bound on the left shows the total number of non-empty entity pairs (i.e. how many pairs actually cooccur in at least one sentence, out of all $\binom{100}{2} = 4950$ theoretically possible pairs) and the upper bound on the right shows the total number of sentences in the corpus which mention at least two of the terms. Our method summarizes more entity pairs across both corpora, and achieves the highest well-formedness scores among all techniques (§3.4.5).

acceptability by presenting randomly selected relation statements to workers on the platform Figure Eight[15] (formerly Crowdflower) and asking workers to rate the extent to which they agree or disagree as to whether a relation statement is a "coherent English sentence" on a scale from 1 to 5. Each relation statement is shown to three workers in total.[16] Our approach is broadly similar to the readability experiments reported in Filippova and Altun [107].

We solicit 481 total judgements from workers and calculate the mean acceptability score, by method and corpus (Table 3.5). Our method achieves the highest mean acceptability score for both corpora.

[15] `https://www.figure-eight.com/`

[16] We use seven test questions to filter out careless or bad faith responses. Workers must answer 70% of test questions correctly to be included in a task's results. We construct test questions blindly, without knowledge of the system which generated the relation statement.

Additionally, aggregating judgments across corpora, we observe a statistically significant (p=8 x 10^{-4}) difference between our method ($\mu = 3.89$, $\sigma = 1.38$) and Stanford OpenIE ($\mu = 3.33$, $\sigma = 1.46$) in a two-tailed t-test. Our method also achieves a higher mean score than ClausIE ($\mu = 3.69$, $\sigma = 1.44$), although the difference is not significant.

3.5 An approach to the summary construction task

After a relationship summarization system generates a candidate set, the next task is selecting the top K candidate statements for inclusion in a summary. The value of K will vary by application. Some applications will allow for longer summaries, while others will need more terse explanations of relationships. Therefore, as a case study, this section proposes and evaluates an annotated method for the summary construction task in ConceptMap browsers, described below.

3.5.1 Introducing ConceptMap browsers

ConceptMaps are are visual summaries, structured as directed graphs (Figure 3.1). Important concepts from a corpus are shown as vertexes. Natural language descriptions of the relationships between concepts are shown as textual labels, along the edges on the map. Initial attempts to generate English-language ConceptMaps within natural language processing [103] have focused on creating static diagrams which summarize collections of documents. However, in interactive settings, users will want to query relationships with a ConceptMap browser interface, rather than simply read over fixed output from a summarization system. For instance, in the ConceptMap browser[17] shown in Figure 3.1, a user has queried for Aristide and the system identifies some co-occurrence relationship between Aristide and Liberation Theology. In this case, the system needs to describe a relationship between $(\boldsymbol{t_1})$ ="Aristide" and $(\boldsymbol{t_2})$ ="Liberation Theology" at query time, and describe their

[17]We use the term ConceptMap *browsers* to refer to interactive concept map interfaces.

relationship using a labeled edge in the graph ("Bertrand Aristide influenced by Liberation Theology" in 3.1). Later, we discuss this approach to summary construction (§3.6).

3.5.2 Annotated summary construction for ConceptMaps

Because there is no existing supervision to help select the best summary statement from the candidate set for ConceptMaps, we collect a new dataset of annotated summary statements, which we use to supervise a new model for this task.

We construct our annotated dataset from news stories focusing on the Balkan Peninsula in the 1990s.[18] Political scientists use rich news archives from this complex period to better understand conflict [295]. We describe our two-step approach to creating the dataset below.

Step One. For each country, we once again (§3.4.3) use the package `phrasemachine` (§2) to identify the 100 highest-frequency noun phrases within articles which mention that country, and examine all pairs of highest-frequency noun phrases which co-occur at least twice in articles about the country. Then we sample a total of 689 non-empty pairs with more than two extracted candidates. In total there are 5,214 candidate statements across 689 sampled sets.[19] On average there are 7.56 statements per set ($\sigma = 10.6$). A sample candidate set is shown in Table 3.6.

---

[18] We create our dataset from *New York Times* articles [292] published from 1990–1999, which mention at least one country from the Balkans. The countries are: Kosovo, Albania, Serbia, Croatia, Montenegro, Macedonia, Bulgaria, Romania, Moldova and Bosnia. We exclude the former Yugoslavia; its landmass included other countries on our corpus.

[19] **Additional notes.** `phrasemachine` sometimes returns overlapping phrases, leading to duplicate sets. We merge duplicates with a heuristic which uses hand-written rules based on (*i*) token overlap between concepts and (*ii*) overlapping sentences between sets. We exclude pairs which are very obviously unrelated to the Balkans (e.g. "Chinatown" and "Little Italy"). Our annotation budget determined the number of annotated sets. As in earlier portions of this chapter, we also allow statements which begin with $(\boldsymbol{t_2})$ and end with $(\boldsymbol{t_1})$; the order of query concepts is important in interfaces which display ConceptMaps, but beyond the scope of this work. We limit statements to a maximum of 75 characters.

To create the dataset, we first identify candidate summary statements beginning with $(\boldsymbol{t_1})$ and ending with $(\boldsymbol{t_2})$ which read as a fluid and well-formed sentence when extracted from naturally-occurring text. As in earlier experiments, we define all spans between $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$ with a probability of well-formedness greater than .5 to be the **candidate set** for the pair $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$.

	A1	A2	A3
c_1 **General Grachev**'s favor is his loyalty to **Mr. Yeltsin**	-	W	-
c_2 **Mr. Yeltsin** openly accused **General Grachev**	-	-	-
c_3 **General Grachev**, Defense Minister by dint of his loyalty to **Mr. Yeltsin**	W	-	W
c_4 **General Grachev**'s plea today will do nothing to help **Mr. Yeltsin**	-	-	-
c_5 **Mr. Yeltsin** might also appear weak if he had to replace **General Grachev**	B	B	B

Table 3.6: A candidate set for $(\boldsymbol{t_1})$ = "**Mr. Yeltsin**" and $(\boldsymbol{t_2})$ = "**General Grachev**," along with decisions from three annotators (A1, A2 and A3) selecting the best (B) and worse (W) summary statement in the set. All annotators agree that c_5 is the best, so $\alpha(c_5)$ =3. (During annotation, the order of all sets was randomized).

Step Two. After defining candidate sets we collect annotations to choose the best summary statement from each candidate set. Our annotation procedure assigns a score $\alpha(c) \in \{-3, -2, ...+3\}$ to each candidate $c = (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2})$ in a candidate set, which is intended to reflect how well s summarizes a particular relationship. We use this supervision to train a model to predict $\alpha(c)$. We propose that the statement with the highest predicted $\alpha(c)$ score should be displayed on a ConceptMap.

Some candidate sets in our dataset are easy for a person to judge and rank. For instance, it is possible to quickly read over the small set shown in Table 3.6 and identify statements which are clearly better and clearly worse synopses of the relationship between "General Grachev" and "Mr. Yeltsin".

However, other candidate sets in our dataset are too large and too complex to read and analyze quickly. (The largest candidate set in our dataset contains 143 statements in total). We accommodate both large and small sets with a "low-context" [103] annotation technique; we split candidate sets into one or more subsets, and ask

annotators to rank the best and worst summary statements in each subset. Then we aggregate these local judgements about the best and worst candidates within each subset to create a global score. This global score, $\alpha(c)$, attempts to capture the overall quality of a given a candidate summary statement c. This method of soliciting local judgements about subsets and then aggregating into an overall score is known as Best-Worst Scaling [206]; it has been shown to make more efficient use of human judgements for a natural language task than traditional techniques [186].

The annotation process assigns a score $\alpha(c) \in \{-3, -2, -1, 0, 1, 2, 3\}$ to each c in each candidate set. A higher value of $\alpha(c)$ indicates greater agreement among annotators that a given c should summarize the candidate set.

3.5.3 Annotation: Additional details

We present all candidate sets to three different non-native English speakers, hired via a professional annotation firm. All annotators completed graduate work in either linguistics or the humanities, and were based in the Middle East. For each annotator, we divide each candidate set into J random tuples (a tuple consists of up to eight candidate statements), and ask the annotator to choose the best and worst from each tuple. Annotators are instructed that the best statement should be the one that both sounds the most natural and that most helps them understand the history and politics of the Balkan region. They are instructed that the most unnatural sounding and least informative statement should be chosen as worst. In total, each candidate statement is shown to each annotator exactly once.[20] After annotators have judged each individual set, we aggregate with Orme [264]'s counting formula: we set the score

[20]Unlike in traditional Best-Worst annotation, the number of candidates in each tuple may vary depending on the size of the candidate set. If a candidate set has a cardinality of less than eight, the size of the tuple is set to the size of the candidate set; otherwise the size of a tuple is capped at eight. We make this choice because many candidate sets have a small cardinality, and it does not make sense to break up small sets (e.g. 5 or 6 candidates) into very small tuples.

$\alpha(c) \in \{-3, -2, \ldots, +3\}$ of each summary statement c to be the number of times c was chosen as the best, minus the number of times it was chosen as the worst.

Following prior work [186], we evaluate inter-annotator agreement via split-half reliability. For each candidate set, we randomly split annotators into two groups, and compute the score for each c using each group of annotators. Then we compute the Spearman correlation (ρ) between the two sets of scores, yielding an average of $\rho = 0.495$ across 1000 random splits.

3.5.4 Modeling annotated data

The previous section describes a procedure for assigning a score, $\alpha(c)$ for each c in our dataset. We use these scores to train a model, $p(\alpha(c)|s)$. During modeling, we divide the dataset into training and test sets at the entity level, ensuring that there are no relationships between concepts in the training and test set. Ensuring that there are no relationships shared across sets is important because a model might use knowledge about relationships gleaned from training data (e.g. "Milosevic led Serbia") to make inferences about relationships in the test data (e.g. "Milosevic led the Serbian Socialist party"). 627 candidates are used for training; the remaining 62 are for testing.[21]

We model $p(\alpha(c)|s)$ using ordinal regression, implemented with the MORD package [273]. We use unigram features, morphological features, part-of-speech-tag features and binary features (e.g. s includes punctuation mark) to represent the candidate statement. We also include $p(wf = 1 \mid s, (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2}))$ as a feature in our model, along with the token length of a summary statement. We tune MORD's regulariza-

[21]To implement the train–test split, we form an initial provisional division of concepts into two sets. For all relationships between concepts that cross the two sets, we move the entity from the test set to the training set. All scored summary statements between concepts in the training set are used for training; the remainder are for test. We manually tune the size of the initial split so that 10% of concepts are in the final test set.

tion penalty parameter to maximize 5-fold, cross-validated Spearman's ρ using the training set.[22]

3.5.5 Model evaluation

We use the test set to measure the extent to which our model's predictions correlate with gold scores, achieving a Spearman's $\rho = 0.443$ between our model's predictions and the gold scores. This is close to the $\rho = 0.495$ computed to measure inter-annotator agreement (see §3.5.3).

We instructed annotators to select summary statements that were both informative and grammatically wellformed. We use the probability of grammatical well-formedness $p(wf = 1 \mid s, (\boldsymbol{t_1})\ r\ (\boldsymbol{t_2}))$ as a feature in our model. This measure appears to partially reflect annotator judgements: there is a Spearman's $\rho = 0.154$ between the two metrics across the dataset. Research into human perceptions of grammatical well-formedness [314, 341] could be applied to make better predictions in the future.

Model	Spearman's ρ
$\boldsymbol{p(\alpha(c)\|s)}$ (Ordinal regression)	**0.443**
Logistic regression	0.304
Inter-annotator agreement	0.495

Table 3.7: Spearman's ρ for our ordinal regression model $p(\alpha(c)|s)$, compared both to the inter-annotator agreement and a simpler logistic regression model.

Predicting annotator perceptions of informativeness is more challenging. For instance, annotators preferred "Mr. Milosevic has been formally charged with war crimes" ($\alpha(c) = 3$) to "President Slobodan Milosevic may be indicted for war crimes" ($\alpha(c) = 1$). The former expresses a completed action which arguably entails the lat-

[22]We examine 10^i for $i = -3, -2..2, 3$ and select $10^1$for the final value of the hyperparameter. Additionally, the MORD API implements several variants of ordinal regression. We use the LogisticSE variant because it achieves the highest cross-validated ρ on the training set.

ter, hypothetical action. How to best model [42], formalize [210] and even study [134] such complex semantic relationships is an unsolved problem in NLP.

We use the number of tokens in a summary statement (subtracting out the length of query concepts) as a feature. We observe a Spearman's $\rho = .337$ between $\alpha(c)$ and the token length of s. We hypothesize that this feature might serve as a very coarse proxy for informativeness: although not instructed to do so, annotators might choose longer statements ahead of shorter statements because they express more about the Balkans.

3.6 A discussion and analysis of the summary construction task

The previous section describes a the collection and modeling of a new annotated resource for selecting statements from the candidate set to summarize relationships. Our approach adopts the basic supervised paradigm underlying much current work on summarization [156, 132]. We collect human judgements of salience and well-formedness (in our case, judgements are expressed via Best-Worst Scaling), and then train a model to best replicate such judgements. Results are mixed. We find that shallow cues like statement length and grammatical well-formedness are helpful for identifying good summary statements, but also that representing deeper semantic relationships (e.g. entailment) remains an ongoing challenge for automatically building ConceptMaps. Thus in this section, we analyze the summary construction task

more deeply, by analyzing mention sets and articulating how their properties might affect future supervised and unsupervised approaches to summary construction[23]

To begin, we observe that mention sets are inherently heterogenous. Some describe a single, temporally-focused event. Others describe a consistent, unchanging relationship. Still others describe intricate sagas unfolding across time. For instance, within the Haiti corpus, one mention set describes events in 1994 when "General Cedras fled to the Dominican Republic". This mention set is quite different from a set of sentences in the Reddit corpus in which users assert that "video games are a deal breaker" in interpersonal relationships. Figure 3.6 displays hand-crafted summarizes for these mention sets.

In general, the properties which guide how a mention set should be summarized are its **size**, **topical diversity**, **temporal focus** and the degree to which the set expresses **states or events**. In this section, we use the notation $\boldsymbol{(t_1)}$ – $\boldsymbol{(t_2)}$ to refer to a mention set. For instance, *New York – London* would refer to all sentences from a corpus which contain the names of both of these cities.

Size. In general, because many word types in a corpus occur infrequently [357], the number of sentences which mention $\boldsymbol{(t_1)}$ and $\boldsymbol{(t_2)}$ is often small. For instance, of the 320,670 total sentences in the Haiti corpus, only 160 mention "Jean-Bertrand Aristide" and the "United States," which is nonetheless among the larger mention sets in the corpus. In general, larger sets often describe complex and noteworthy relationships, which are more difficult to summarize (Figure 3.5). Note that although individual mention sets are often small enough to simply read (unlike in some multi-

[23]Because such supervision is costly and difficult to collect, carries risks of annotation artifacts [134] and might transfer poorly to new domains, future work might explore if other forms of task-based supervision and task-based evaluation [176] may be better suited to the specialized task of automatic ConceptMap summarization. For instance, instead of asking a human to identify better and worse summary statements, we might examine how well a user (or model) presented with summary statement s can answer if other summary statements s' are true or false [101]. If some s helps users identify many other true s', then s is (potentially) a good summary.

Figure 3.3: Mention sets are heterogeneous, requiring a diversity of summarization techniques. In this work, we analyze the diversity of mention sets towards future attempts that the relationship summarization problem.

video games and I don't want that to be a **deal breaker**
video games was a **deal breaker**
video games is a **deal breaker**

Figure 3.4: A hand-crafted summary for the mention set **video games–deal breaker**. The mention set contains many stative descriptions of the relationships between the two terms, indicating that a summary might focus on presenting fixed relationships rather than evolving events.

General Cedras ... last week fled to the **Dominican Republic**
Dominican Republic ... will not permit permanent residence by **General Cedras**

Figure 3.5: A hand-crafted summary for the mention set **General Cedras–Dominican Republic**. The set has a high number of mentions which all fall within a several month span, hinting at a relationship focused on a particular event at a particular point in time.

United States supports the restoration of ... **Jean-Bertrand Aristide** (Aug. 1994)
Jean-Bertrand Aristide was restored ... a year ago under the watch of **United States** (Oct. 1995)
United States ... withheld contributions, hoping to spur President **Jean-Bertrand Aristide** (Sep. 2002)
Jean-Bertrand Aristide asserted ... he had been driven from power by the **United States** (Mar. 2004)

Figure 3.6: A hand-crafted summary for the mention set **Jean-Bertrand Aristide–United States**, one of the largest in the Haiti corpus. The mention set describes a complex, shifting relationship; at different times over several decades, Aristide was a beneficiary, opponent and critic of the United States.

document summarization settings), summarization of mention sets is still quite useful, as practitioners will often seek to understand many different relationships as they investigate a new topic (e.g. Figure 3.1).

Topical diversity. In general, some mention sets are focused on a single topic, others are more diffuse. For instance, after losing power in a second, 2004 coup Haiti's Jean Bertrand Aristide was forced into exile in South Africa. The mention set for *Jean Bertrand Aristide – South Africa* contains twelve sentences which (mostly, but not exclusively) describe Aristide's removal from power and life in exile in South Africa from 2004 onwards. Detecting and including diverse or complex topics is a classic aim of traditional multi document summarization (e.g. Lin and Hovy [200]), which might be applied in this new setting.

Temporal focus. In timestamped corpora such as news archives or social media posts, some mention sets are focused within certain time periods; others are spread across the span of the corpus. For instance, in the Haiti corpus, *General Cedras – Dominican Republic* are only mentioned together during a few months of 1994 (Figure 3.4). A good summary for this mention set should describe a central event from this time period: when General Cedras fled to the Dominican Republic. On the other hand, *Jean-Bertrand Aristide – United States* are mentioned together in 67 months in the corpus, covering a number of important events spread across decades (Figure 3.5). For this mention set, a narrow summary focusing on a single event would be inappropriate.

Many existing methods specialize in detecting [54], tracking [6] and summarizing evolving topics in timestamped documents. Some systems focus specifically on summarizing event "spikes": both in news (e.g. Alfonseca et al. [5]) and on social media (e.g. Nichols et al. [254]). In some cases, the event described in a mention set will even match the loose form of a common narrative template [53], such as when $(\boldsymbol{t_1})$ and $(\boldsymbol{t_2})$ are codefendants at a trial.

Mention sets which are more temporally diffuse are also more challenging. Update summarization refers to summarizing changes introduced by new documents, possibly from a high volume stream [180]. This form of summarization is important in cases when a relationship shifts or changes through time, as in figure 3.5.

States or events. Mention sets may be coarsely divided into cases where the set expresses a stable state or property of the world in the eyes of the author (e.g. "England is a close ally of the US" or "video games are a deal breaker") and cases where the relation statement expresses a change or event (e.g. "Gen. Cedras fled to the Dominican Republic" or "dad left mom"). In many interesting cases, the mention set contains a mix of stative and eventive relation statements which express a narrative,

such as "America is an ally of South Korea" and "America sent a destroyer to South Korea".

Defining [281], extracting [4] and determining relationships between events [164] is a challenging research area. But a better understanding of states and events would improve future work. For instance, if a summary includes the event "Jolie divorced Pitt", it does not need to include the stative relation phrase "Jolie was married to Pitt". To our knowledge, there is no prior work which considers how fine-grained relations between states and events might be employed for summarization. MacCartney and Manning [210] offer a framework for enumerating possible relationships between propositions, which might serve as a useful starting place.

3.7 Conclusion

This chapter presents a new NLP method for summarizing relationships between lexical items, which can help explain why particular lexical items co-occur in text. We show that our method can summarize more query pairs than baseline relation extraction systems, and analyze one application of our approach in ConceptMap interfaces. In Chapter 7, we also show how relationship summarization can help in a user-facing system designed for historians and archivists.

CHAPTER 4

TEXT SIMPLIFICATION (CLAUSE DELETION)

This chapter is adapted from a preprint entitled *Human acceptability judgements for extractive sentence compression* [143].

Synopsis

In query-focused lexical corpus analysis (see Section 1.2.1), the user investigates a particular lexical item (or group of lexical items) in a body of text. But because the user's query Q may be mentioned many times within a corpus, reading all mentions of Q within long passages may be burdensome. We thus propose applying text simplification methods from natural language processing to help shorten sentences containing user query words, to speed the review of lexical items in context. Specifically, in this chapter, we consider how to simplify English sentences by collecting and modeling human judgments of simplified text, and then use a model of such judgments in a new sentence simplification system. Later, we apply findings from this effort in the ClioQuery interface (Chapter 7).

4.1 Introduction

In natural language processing, *sentence compression* refers to the task of automatically shortening a longer sentence [189, 67, 109]. This technique has obvious applications in lexical corpus analysis, because it reduces the amount of text a user must read to review mentions of a lexical query Q in a corpus. Therefore, in this chapter, we investigate how to shorten sentences containing Q. Specifically, we collect a crowdsourced dataset of human judgments describing the well-formedness of shortened sentences (§4.4), present and evaluate a model of such human judgments (§4.5), and use this model in a new sentence compression technique (§4.6), which predicts the well-formedness of a shortening. We then show how this approach allows a practitioner to identify well-formed shortenings which include Q. Later, we apply a version of our clause-deletion approach in the CLIOQUERY system (Chapter 7).

4.2 Related work

Researchers have been studying extractive sentence compression for over two decades [189, 67, 109]. Recent approaches are often based on a large compression corpus,[1] which was automatically constructed by using news headlines to identify "gold standard" shortenings [107]. State-of-the-art models trained on this dataset [109, 11, 339] can reproduce gold compressions (i.e. perfect token-for-token match) with accuracy higher than 30%.

However, learning from parallel corpus supervision has clear downsides for lexical corpus analysis. Parallel corpus supervision implicitly assumes a given rate of compression, which may not be appropriate for a given application (e.g. maybe a user needs very short compressions), and encodes assumptions about the distribution

[1] `https://github.com/google-research-datasets/sentence-compression`

of Q within sentences (e.g. is Q usually the subject, does it appear in embedded clauses?).

Therefore, in this Chapter, we investigate the use of human acceptability judgements, as a new and more flexible form of supervision for the sentence compression task. Our approach is based on prior research which seeks to model human judgements of the well-formedness of English sentences [154, 314, 195, 341]. However, unlike more general studies, our work is strictly concerned with human perceptions of shortened sentences (for application in lexical corpus analysis).[2] Unlike prior work, our study also solicits human judgements of shortenings from naturally-occurring news text, instead of sentences drawn from syntax textbooks [315, 341] or created via automatic translation [195].

We note that our effort focuses strictly on anticipating the well-formedness of extractive compressions, rather than identifying compressions which contradict or distort the meaning of the original sentence. Identifying which compressions do not modify the meaning of source sentences is closely connected to the unsolved textual entailment problem, a recent area of focus in computational semantics [42, 271, 227]. In the future, we hope to apply this evolving research to the compression task. Some current compression methods use simple hand-written rules to guard against changes in meaning [67], or syntactic mistakes [175].

Finally, following much prior work, this study approaches sentence compression as a purely extractive task. Closely related work on abstractive compression [68, 291, 214] and sentence simplification [355, 347] seeks to shorten sentences via paraphrases or reordering of words. Despite superficial similarity, extractive methods typically use different datasets, different evaluation metrics and different modeling techniques.

[2]We compare our model to Warstadt et al. [341] in §4.5.

Sentence	Pakistan launched a search for its missing ambassador to Afghanistan on Tuesday, a day after he disappeared in a Taliban area.
Headline	Pakistan searches for missing ambassador.
"Gold" compression	Pakistan launched a search for its missing ambassador.
Alternate 1	Pakistan launched a search for its missing ambassador to Afghanistan on Tuesday. (A(c) = -1.367, BREVITY = 84 characters max., QUERY = 1)
Alternate 2	Pakistan launched search Tuesday. (A(c) = -6.144, BREVITY = 59 characters max., QUERY = 0)

Table 4.1: A sentence, headline and "gold compression" from a standard sentence compression dataset [107], along with two alternate compressions constructed with a system supervised with human acceptability judgements (§4.6). The alternate compressions reflect different BREVITY requirements and different adherence to an application-specific, lexical QUERY criterion. In this case, the brevity requirement is expressed with a hard maximum character constraint and the importance criterion is expressed with a binary score, indicating if a sentence includes a user's query term Q="Afghanistan". We use our A(c) $\in (-\infty, 0]$ metric (§4.6) to measure the ACCEPTABILITY of each compression; a higher score indicates a compression is more likely to be well-formed. Alternate 2 is neither entirely garbled nor perfectly well-formed, reflecting the gradient-based nature of acceptability (§4.4).

4.3 Compression via subtree deletion

Any sentence compression technique requires a framework for generating possible shortenings of a sentence, s. (In §5 we denote each input sentence with an uppercase S to emphasize that S is a set. In this chapter, we use a lower case s, because s refers to a dependency parse graph, rather than a vertex set.) We generate compressions with a subtree deletion approach, based on prior work [189, 108, 107]. To generate a single compression (from among all possible compressions) we begin with a dependency parse of s.[3] Then, at each timestep, we prune a single subtree from the parse. After M subtrees are removed from the parse (one at a time, over M timesteps) the remaining

[3]We use Universal Dependency [259] trees (v1), parsed using CoreNLP [217].

vertexes are linearized in their original order. Formally pruning a subtree refers to removing a vertex and all of its descendants from a dependency tree; pruning singleton subtrees (one vertex) is permitted.

We find that is possible to construct 88.2% of the gold compressions in the training set of a standard corpus [107] by only pruning subtrees. Therefore, we only examine prune-based compression in this work.[4]

4.4 Human acceptability judgements for sentence compression

Our data collection methodology follows extensive research into human judgements of linguistic well-formedness [314]. Such work has shown that non-expert participants offer consistent judgements of natural and unnatural sounding sentences with high test-retest reliability [192], across different data collection techniques [14, 315]. We apply this research to sentence compression, with confidence that our results reflect genuine human perceptions of shortened sentences because:

1. We carefully screen out many workers who offer judgements that violate well-known properties of English syntax, such as workers who approve deletion of objects from obligatory transitive verbs.

2. We observe that workers approve and disapprove of classes of sentences that English speakers would categorize as "grammatical" and "ungrammatical," respectively. For instance, workers rarely endorse deletion of nominal subjects, and often endorse deletion of temporal modifiers.

3. Annotator agreement in our dataset is similar to agreement in prior, comprehensive studies of acceptability judgments (§4.4.4).

---

[4]Extracting nested subclauses from source sentences is not possible with prune-only methods, because the root node of the compression must be the same as the root node of the original sentence. We plan to address this in future work.

The Appendix (§4.8) details our screening procedures, and discusses classes of accepted and rejected compressions in our dataset.

4.4.1 Methodology: measuring well-formedness

Our study adopts a standard distinction between acceptability and grammaticality [61, 297]. Grammaticality is a binary and theoretical notion, used to characterize whether a sentence is or is not generable under a grammatical model. Acceptability is a measurement an individual's perception of the well-formedness of a sentence. Empirical studies have shown acceptability to be a gradient-based phenomenon, affected by a range of factors including plausibility, syntactic well-formedness, and frequency [314]. Based on this work, we expect that workers will have graded (not binary) perceptions of the well-formedness of compressions shown in our task.

Although acceptability is gradient-based, we nevertheless *measure* worker perceptions by collecting binary judgements of well-formedness. Earlier studies [14, 315, 192] have shown that such binary measurements of acceptability correlate strongly with explicitly gradient collection techniques, such as Likert scales (Figure 4.1). We chose to collect binary judgements instead of graded judgments because (1) binary judgments avoid ambiguity in how participants interpret a gradient scale (2) binary judgements allowed us to write clear screener questions to block unreliable crowdworkers from the task and (3) binary judgements allowed us to apply binary logistic modeling to directly predict observable worker behavior.

4.4.2 Data collection prompt

We show crowdworkers on Figure Eight[5] a naturally-occurring sentence, along with a proposed compression of that same sentence, generated by executing a *single* prune operation on the sentence's dependency tree. We then ask a binary question:

[5] `https://www.figure-eight.com/`.

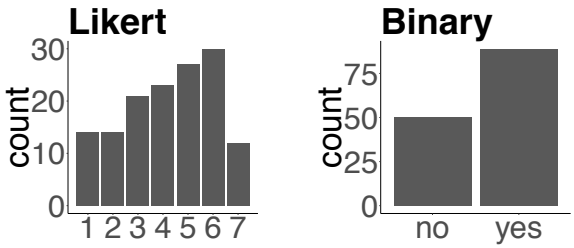


Figure 4.1: Binary judgements and graded (Likert) judgements from Sprouse and Almeida [315] for the slightly-awkward sentence, "They suspected and we believed Peter would visit the hospital". Bader and Häussler [14] describe correlations between such measurement techniques.

can the longer sentence be shortened to the shorter sentence? Our prompt is shown in Figure 4.2. We instruct workers to say yes if the shorter sentence "sounds good" or "sounds like something a person would say," following verbiage for the acceptability task [314].

Because we designed our task to follow typical acceptability prompts, we expect that workers completing the task evaluated the well-formedness of each compressed sentence, and then answered *yes* and if they deemed it acceptable.

We instruct workers to say yes if a compression sounds good, even if it changes the meaning of a sentence. While practitioners performing lexical corpus analysis will need to identify shortenings which are both syntactically well-formed *and* which do not modify the meaning of a sentence, this work focuses strictly on identifying well-formed compressions. In the future, we plan to apply active research in semantics (§4.2) to identify disqualifying changes in meaning.

4.4.3 Dataset details

We generate 10,128 sentence–compression pairs from a freely-distributable corpus of web news [354]. Each source sentence s is chosen at random, and each compression c is produced by a single, randomly-chosen prune operation on s. Our data thus reflects the natural distribution of dependency types in the corpus.

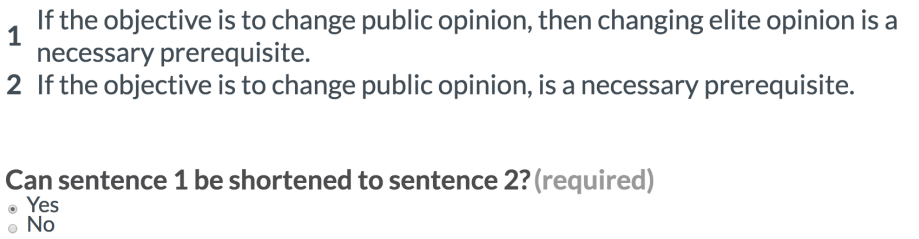


Figure 4.2: Prompt to collect human judgements of acceptability for sentence compression. Workers are instructed to answer yes if the shorter sentence "sounds good" or "sounds like something a person would say."

N judgements (train)	6010 (4522 sents.)
N judgements (test)	640 (486 sents.)
class balance	64.2%/35.8% (no/yes)
overall compression rate	μ=0.867 σ=0.174

Table 4.2: Corpus statistics.

We present each (s, c) pair to 3 or more workers,[6] then conservatively exclude many judgements from workers who are revealed to be inattentive or careless (see Appendix 4.8), in order to be certain that worker disagreements in our dataset reflect genuine perceptions of well-formedness. We then divide filtered data into a training and test set by (s, c) pair, so that our model does not use a train-time judgement about (s, c) from worker k to predict a test-time judgement about (s, c) from worker k'. Table 4.2 presents dataset statistics.[7] Our organization does not require institutional approval for crowdsourcing.

4.4.4 Measuring inter-annotator agreement

There are at least two sources of inter-annotator disagreement which could arise in our data. First, in cases when a compression is neither entirely garbled nor perfectly

[6]FigureEight will sometimes solicit additional judgements automatically.

[7]Note that we use a character-based Filippova et al. [109] rather than token-based [245] definition of compression rate.

well formed, previous empirical studies [315, 192] suggest that annotators will likely disagree. (See Figure 4.1 and §4.4.1). Second, we suspect that different individuals set different thresholds on how acceptable a sentence must be before they give a "yes" response: a compression that one person rates as acceptable might be rated by the next as unacceptable, even if they have the same impression of the compression's acceptability. Such between-annotator variability might represent a form of response bias, which is common in psychological experiments [211, 212]. We attempt to control for such bias by including a worker ID feature in our model (§4.5.1).

To evaluate the extent of such disagreement, and to compare with other work, we measure inter-annotator agreement using Fleiss' kappa [113], computing a $\kappa = .294$ on the entire filtered dataset.[8] We observe a similar rate of agreement ($\kappa = .323$) in a comprehensive study of acceptability judgements [315].[9] Our κ is lower than typical in standard annotation paradigms in NLP, which often attempt to assign instances to hard classes, rather than measure graded phenomena.

4.5 Intrinsic task: Modeling single operation compressions

We train a binary logistic regression model, $p(Y_k = 1|\boldsymbol{W}, \boldsymbol{x})$ to predict if worker k will endorse a deletion transforming s to c, where $\boldsymbol{x} = \phi(s, c, k)$ is a feature vector described in §4.5.1. We then show how the intrinsic task of modeling and controlling for individual worker behavior can be used for the extrinsic task (§4.6) of predicting the ACCEPTABILITY of multi-operation compressions, without reference to worker information.

---

[8] See Appendix (§4.8) for details of Fleiss κ for crowdsourcing.

[9] We compute this number using publicly released data from the YN study.

4.5.1 Model features

The major features in our model are: (i) language model features, (ii) dependency type features, (iii) worker ID features, (iv) features reflecting properties of the edit from s to c and (v) interaction features. We discuss each below.

Language model features. Our model builds upon earlier work examining the relationship between language modeling and acceptability judgements. In a prior study, Lau et al. [195] define several functions which normalize predictions from a language model by token length and by word choice; then test which functions best align with human acceptability judgements. We use their Norm LP function in our model, defined as:

$$\text{Norm LP}(\xi) \triangleq -\frac{\log\, p_m(\xi)}{\log\, p_u(\xi)} \tag{4.1}$$

where ξ is a sentence, $p_m(\xi)$ is the probability of ξ given by a language model and $p_u(\xi)$ is the unigram probability of the words in ξ.

We use Norm LP as a part of two features in our approach. One real-valued feature records the probability of a compression computed by Norm LP(c). Another binary feature computes Norm LP(s) - Norm LP(c) > 0. The test set performance of these language model (LM) features is shown in Table 4.3. The appendix further describes our implementation of Norm LP.

Dependency type features. We use the dependency type governing the subtree pruned from s to predict the acceptability of c. This is because workers are more likely to endorse deletion of certain dependency types. For instance, workers will often endorse deletion of temporal modifiers, and often reject deletion of nominal subjects.

Worker ID features. We also include a feature indicating which of the workers in our study submitted a particular judgement for a given (s, c) pair. We include this

feature because we observe that different workers have greater or lesser tolerance for more and less awkward compressions.[10]

Including the worker ID feature allows our model to partially account for an individual worker's judgement based on their overall endorsement threshold, and partially account for a worker's judgement based on the linguistic properties of the edit. The feature thus controls for variability in each worker's baseline propensity to answer yes. Because real applications will not have access to worker-specific information, we do not use the worker ID feature in evaluating our model and dataset for use in practical compression systems (§4.6). All workers in the test set submit judgements in the training set.

Edit property features. We also include several features which register properties of an edit, such as features which indicate if an operation removes tokens from the start of the sentence, removes tokens from the end of a sentence, or removes tokens which follow a punctuation mark. We include a feature that indicates if a given operation breaks a collocation (e.g. "He hit a ~~home~~ run"). The appendix details our collocation-detection technique.

Interaction features. Finally, we include seventeen interaction features formed by crossing exit property features with particular dependency types. For instance, we include a feature which records if a prune of a $\xrightarrow{\text{conj}}$ removes a token following a punctuation mark.

4.5.2 Model evaluation

We compare our model of individual worker judgements to simpler approaches which use fewer features (Table 4.3), including an approach which uses only language model information [195] to predict acceptability. We compute the test set accuracy of

[10]More formally, we define a given worker's deletion endorsement rate as the number of times a worker answers yes, divided by their total judgements. We observe a roughly normal distribution $(\mu = .402\ , \sigma = .216)$ of worker deletion endorsement rates across the dataset.

	Hard classification (t=0.5)		Ranking
Model	Accuracy	Fleiss κ	ROC AUC (p)
CoLA	0.622	-0.210	0.590 ($< .001$)
language model (LM)	0.623	-0.232	0.583 ($< .001$)
+ dependencies	0.664	0.124	0.646 ($< .001$)
+ worker ID	0.695	0.232	0.746 ($< .001$)
full $\triangleq p(Y_k = 1 \mid \boldsymbol{W}, \boldsymbol{x})$	**0.742**	**0.400**	**0.807**
- dependencies	0.731	0.368	0.797 (0.073)
- worker ID	0.667	0.170	0.691 ($< .001$)
worker – worker agreement	0.636*	0.270	—
Sprouse and Almeida [315]	—	0.323	—

Table 4.3: Test set accuracy, Fleiss' κ and ROC AUC scores for six models, trained on the single-prune dataset (§4.4), as well as scores for a model trained on the CoLA dataset [341]. The simplest model uses only language modeling (LM) features. We add dependency type (+ dependencies) and worker ID (+ worker IDs) information to this simple model. We also remove dependency information (- dependencies) and worker information (- worker ID) from the full model. The full model achieves the highest test set AUC; p values beside each smaller AUC score show the probability that the full model's gains over the smaller AUC score occurs by chance. We also compute κ for each model by calculating the observed and pairwise agreement rates [113] for judgements submitted by the crowdworker and "judgements" submitted by the model. Models which can account for worker effects achieve higher accuracies than the observed agreement rate among workers (0.636*), leading to higher κ than for worker–worker pairs.

each approach in predicting binary judgements from individual workers, which allows for comparison with agreement rates between workers. However, because acceptability is a gradient-based phenomenon (§4.4), we also evaluate without an explicit decision threshold via the area under the receiver operating characteristic curve (ROC AUC), which measures the extent to which an approach ranks good deletions over bad deletions. ROC AUC thus measures how well predicted probabilities of binary positive judgements correlate with human perceptions of well-formedness. Other work which solicits gradient-based judgements instead of binary judgements evaluates with Pearson correlation [195]; ROC AUC is a close variant of the Kendall ranking corre-

lation [252]. Our full model also achieves a higher AUC than approaches that remove features from the model. We use bootstrap sampling [34] to test the significance of AUC gains (Table 4.3); p values reflect the probability that the difference in AUC between the full model and the simpler model occurs by chance.

The probability that two workers, chosen at random, will agree that a given s in the test set may be shortened to a given c in the test set is 63.6%. We hypothesize that the full model's accuracy of 74.2% is higher than the observed agreement rate between workers because the full model is better able to predict if worker k will endorse an individual deletion.

We also compare our full model to a baseline neural acceptability predictor trained on CoLA [341], a corpus of grammatical and ungrammatical sentences drawn from syntax textbooks. Using a pretrained model, we predict the probability that each source sentence and each compression is well formed, denoted CoLA(s) and CoLA(c). We use these predictions to define four features: CoLA(c), log CoLA(c), CoLA(s) - CoLA(c), and log CoLA(s) - log CoLA(c). We show the performance of this model in Table 4.3. CoLA's performance for extractive compression results warrants future examination: large corpora designed for neural methods sometimes contain limitations which are not initially understood [58, 173, 42].

4.6 Extrinsic task: Modeling multi-operation compressions

In this chapter, we argue that a single sentence may be compressed in many ways; a user performing lexical corpus analysis may wish to see any lexical item Q in context, regardless of whether it occurs in the "gold standard" compression. Thus, in the remainder of this study, we examine how our clause-based sentence compression framework (§4.3), supervised with human acceptability judgements (§4.5), may be used to provide ACCEPTABILITY scores which align with human perceptions of well-

formedness. Such scores could be used as a component of many different practical sentence compression systems, including a method described in §4.6.3.

4.6.1 Defining multi-operation ACCEPTABILITY scores

We consider any function which maps a compression to some real number reflecting its well-formedness to be an ACCEPTABILITY score. In §4.5, we present a model, $p(Y_k = 1|\boldsymbol{W}, \boldsymbol{x})$, which attempts to predict if worker k will judge a single-operation compression to be well-formed. If we execute a chain of M such operations, and assume that each operation's effect on acceptability is independent, we can model the probability that M prune operations will result in an acceptable compression with $\prod_i^M p(Y_k = 1|\boldsymbol{W}, \boldsymbol{x_i})$, which is equal to the chance that a person will endorse each of the M deletions. We test this model with a function that expresses the probability that all operations are acceptable:

$$\mathrm{A}(c) \triangleq \sum_{i=1}^{M} \log \; p(Y = 1|\boldsymbol{W}, \boldsymbol{x_i}) \tag{4.2}$$

where each $\boldsymbol{x_i}$ are features reflecting the nature of the prune operations which shortens c_i to c_{i+1} in the chain of operations, and where $p(Y = 1|\boldsymbol{W}, \boldsymbol{x_i})$ is the predicted probability of deletion endorsement under our full model. Because no worker observes the deletion, we do not use the worker ID feature in predicting deletion endorsement, and we write $p(Y = 1)$ instead of $p(Y_k = 1)$.

The sum of log probabilities in $\mathrm{A}(c)$ reflects the fact that any operation on a well-formed sentence carries inherent risk: modifying a sentence's dependency tree may result in a compression which is not acceptable. The more operations executed the greater the chance of generating a garbled compression. We use this intuition to define a simpler alternative, $\mathrm{A}_M(c) \triangleq -M$, where M is the number of prune operations used to create the compression. We also examine a function $\mathrm{A}_{\min}(c) \triangleq$

$\min\{\log p(Y = 1|\boldsymbol{W}, \boldsymbol{x_i}) \mid i \in 1..M\}$, which represents our observation that a single operation with a low chance of endorsement will often create a garbled compression. We compare these functions to $\mathrm{A}_{LM}(c)$, which is equal to the probability of the compression c under a language model, normalized by sentence length. (We use the Norm LP formula defined in §4.5. Language model predictions have been shown to correlate with the well-formedness of a sentence [195, 179].) Finally, we test a function $\mathrm{A}_{CoLA}(c)$, which is equal to the predicted probability of well-formedness of the compression from a pretrained acceptability predictor (§4.5).

4.6.2 Evaluating multi-operation ACCEPTABILITY scores

To evaluate each ACCEPTABILITY function, we collect a small dataset of multi-prune compressions.[11] We draw 1000 initial candidate sentences from a standard compression corpus [107], and then remove sentences which are shorter than 100 characters to create a sample of 958 sentences. We then compress each of the sentences in the sample by executing prune operations in succession until the character length of the remaining sentence is less than B, a randomly sampled integer between 50 and 100. This creates an evaluation dataset with a (roughly) uniform distribution, by character length.

To generate each compression, we use a sampling method which allows us to explore a wide range of well-formed and garbled shortenings, without generating too many obviously terrible compressions.[12] Concretely, we sample each prune operation in each chain in proportion to $p(Y = 1)$, a model's prediction (§4.5) that the edit will be judged acceptable. This means that we delete vertex v and its descendants with probability $\frac{1}{Z}p(y = 1|W, s, c_v)$, where c_v is the compression formed by pruning the

[11] Rather than single-prune compressions (§4.4).

[12] Very many exponential possible compressions of a single sentence will be garbled or non-sensical. Pruning even a single subtree at random from an acceptable sentence (§4.5) destroys acceptability more than 60% of the time.

subtree rooted at v and $Z = \sum_{v \in \mathcal{V}} p(y = 1|W, s, c_v)$ is the sum of the endorsement probability of all possible compressions.[13]

We show each sentence in the evaluation dataset to 3 annotators, using our acceptability prompt (Figure 4.2). This creates final evaluation set consisting of 2,388 judgements of 940 multi-prune compressions, after we implement the judgement filtering process described in Appendix 4.8. We compute κ=0.099 for the evaluation dataset.

For all defined ACCEPTABILITY functions, we measure AUC against binary worker deletion endorsements (yes or no judgements) in the evaluation dataset to determine the quality of the ranking produced by each ACCEPTABILITY function (§4.5.2). The function $\mathrm{A}(c)$, which integrates information from a language model, as well as information about the grammatical details of the process which creates c from s, achieves the highest AUC on the evaluation set, best correlating with human judgements of well-formedness.

Function	Description	ROC AUC
$\mathrm{A}_{CoLA}(c)$	CoLA pretrained	.510
$\mathrm{A}_{LM}(c)$	Language model	.557
$\mathrm{A}_{M}(c)$	Number of operations $\times$ -1	.580
$\mathrm{A}_{\min}(c)$	Least acceptable operation	.581
$\mathrm{A}(c)$	All operations acceptable	.591

Table 4.4: ROC AUC for several ACCEPTABILITY functions for the multi-operation compression task. The $\mathrm{A}(c)$ model achieves a gain of .034 in AUC over the $\mathrm{A}_{LM}(c)$ model ($p = 0.005$).

[13] In the behavioral sciences, this method of choosing actions is sometimes called *probability matching* [337].

4.6.3 Exploring many compressions of one sentence

This work argues that there is no single best way to compress a sentence. We demonstrate this idea by examining some of the exponential possible compressions of the sentence shown below. (This same sentence is also shown in Table 4.1.)

s = Pakistan launched a search for its missing ambassador to Afghanistan on Tuesday, a day after he disappeared in a Taliban area.
c_g = Pakistan launched a search for its missing ambassador
c_1 = Pakistan launched a search for its missing ambassador to Afghanistan on Tuesday
c_2 = Pakistan launched search Tuesday

Table 4.5: A sentence s and a "gold" compression (c_g) from a standard corpus [107], along with two alternate compressions (c_1 and c_2).

We generate an initial list of 1000 possible compressions of this 126-character sentence via the procedure defined in §4.6.2, and we score the ACCEPTABILITY of each compression by A(c). In this instance, we define BREVITY(c) to be the maximum character length of a compression and QUERY(c) to be a binary function returning 1 only if the compression includes the lexical query term, Q="Afghanistan". Compressions which do not include Q="Afghanistan" would not be suitable for a user seeking to examine mentions of "Afghanistan in context.[14]

Following deduplication steps described in the Appendix 4.8, we generate a final list of 554 different possible compressions of the sentence. We plot each of the 554 compressions in Figure 4.3, which shows many possible shortenings with high $A(c)$ scores. The "gold standard" compression is just one arbitrary shortening of a sentence.

[14]Practical compression systems would also need to check for changes in meaning resulting from deletion (§4.2), but we leave this step for future work.

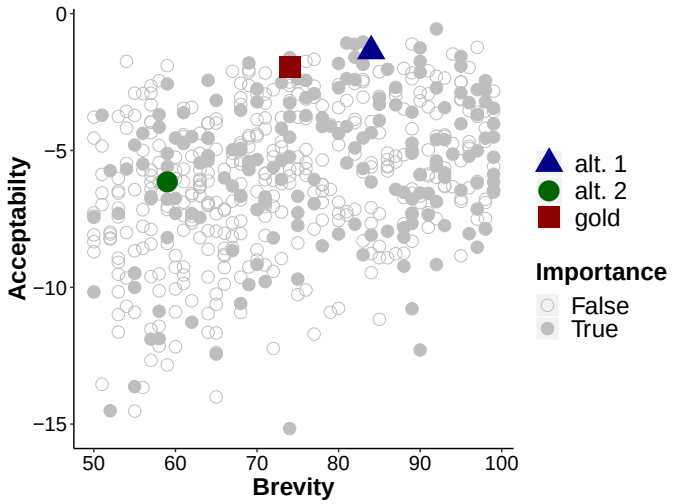


Figure 4.3: 554 possible compressions of a single sentence (Table 4.5), displayed by $A(c)$ score, BREVITY constraint and lexical QUERY criterion. The "gold standard" compression, c_g, is shown with a large red square, along with alternate c_1 (large triangle), and alternate c_2 (large circle).

4.7 Conclusion and future work

Our effort suggests a number of obvious steps for future work. To begin, our study is strictly concerned with grammatical and not semantic acceptability. In the future, we hope to apply active research in semantics (§4.2) to identify meaning-changing compressions. We also plan to add support for additional operations (e.g. paraphrasing), develop improved models of multi-prune compression, and adapt our ACCEPTABILITY scores to offer differentiable add-ons to neural network loss functions.

4.8 Appendix

4.8.1 Crowdsourcing details

We paid workers 5 cents per judgement and only opened our task to US-based workers with a level-2 designation on Figure Eight.

Following standard practice on the crowdsourcing platform, we used test questions to screen out careless workers [312]. All workers began our job with a *quiz mode* of 10 screener questions, and then saw one screener question in every subsequent page of 10 judgements. Workers who failed more than 80% of test questions were screened out from the task. Screener questions were indistinguishable from our regular collection prompt.

We wrote test screener questions based on established understanding of English syntax, to avoid biasing results with our own subjective judgements of acceptability. For example, linguists have extensively examined which English verbs require objects and which verbs do not require objects via corpus-based, elicitation-based and eye-tracking methods [117, 318]. We used this work to write screener questions which check that workers answer no for operations that prune direct objects of obligatory transitive verbs. Similarly, we wrote screener questions which check that workers answer no to deletions which split a verb and a known obligatory particle in a multi-word expression [18], or remove determiners before singular count nouns [166, p. 354]. For the multi-prune dataset (§4.6.2), we added test questions which confirmed that workers approved of well-formed, gold standard compressions from a standard corpus [107].

We also include screener questions which check if a worker is paying attention, along with several poll questions which ask workers if they grew up speaking English.[15] We ignore judgements from known non-native speakers and known inattentive workers in downstream analysis.[16] We defined rules for filtering the dataset before examining the test set to ensure that filtering decisions did not influence test-set results. We

[15]Workers are instructed there is no right answer to questions about language background, so there is no incentive to answer dishonestly.

[16]We also exclude 1663 suspected fraudulent judgements from 17 IP address associated with multiple worker IDs.

release screener questions and task instructions along with crowdsourced data for this work.

4.8.2 Per-dependency deletion endorsements

Breaking out worker responses by dependency type provides additional validation for our data collection approach. We observe that workers are unlikely to endorse deletion of dependency types which create compressions that English speakers would deem "ungrammatical," and likely to endorse deletions which speakers would deem "grammatical."

For example, in UD, the $\xrightarrow{\text{mwe}}$ relation is most commonly used to link two or more function words that obligatorily occur together (e.g. *because of, due to, as well as*). Since deleting a $\xrightarrow{\text{mwe}}$ amounts to suppressing a critical closed-class item, it is not surprising that, overall, workers only assented to deleting $\xrightarrow{\text{mwe}}$ in 9.5% of cases. Similarly, low deletion endorsement for the $\xrightarrow{\text{cop}}$ relation (15.6%) is consistent with the grammatical rules of mainstream varieties of American English, which generally require an overt copula in copular constructions.[17]

On the other hand, we found that optional [193] pre-conjunction operators like *both* or *either* were almost always considered removable (80.0% deletion endorsement). Workers also endorsed the deletion of temporal adverbs such as *tomorrow* or *the day after next* 78.9% of the time, which is sensible as temporal adverbs are typically considered adjuncts.

Since these response patterns generally align with well-established grammatical generalizations [166], they serve to validate our data collection approach.

---

[17]Not all dialects require overt copulas [129].

4.8.3 Experimental details

We report additional details regarding several of the experiments in the paper, presented in the order in which experiments appear.

Fleiss κ. Fleiss' original metric [113] assumes that each judged item will be judged by exactly the same number of raters. However, our data filtering procedures create a dataset with a variable number of raters per sentence-compression pair. (This is common in crowdsourcing.) We thus calculate the observed agreement rate for an individual item (P_i, in Fleiss' notation) by computing the pairwise agreement rate from among all raters for that item. We ignore cases where only one rater judged a given (s, c) pair, which occurs for 73.2% of pairs.

Tuning and implementation. We implement our model $\triangleq p(Y = 1|\boldsymbol{W}, \boldsymbol{x})$ with scikit-learn [272] using L2 regularization. We tune the inverse regularization constant to $c = 0.1$ to optimize ROC AUC in 5-fold cross validation over the training set, after testing $c \in \{10^j \mid j \in \{-3, -2, -1, 0, 1, 2\}\}$. We do not include a bias term. All other settings are set to default values.

NormLP. Following Lau et al. [195], in this work, we use the Norm LP function to normalize output from a language model to predict grammaticality. Our Norm LP function uses predictions from a 3-gram language model trained on English Gigaword [266] and implemented with KenLM [148]. Lau et al. [195] report identical performance for the Norm LP function using 3-gram and 4-gram models.

Lau et al. [195] found that another function, SLOR, performed as well as Norm LP in predicting human judgements. We found that Norm LP achieved higher AUC than SLOR in 5-fold cross-validation experiments with the training set.[18]

Collocations. Our model includes a binary feature, indicating if an edit breaks a collocation. We identify collocations by computing the offsets (signed token distances)

[18]Kann et al. [179] also examine SLOR for automatic fluency evaluation.

between words [215, ch. 5.2] in English Gigaword [266]. If the variance in token distance between two words is less than 2 and the mean token distance between the words is less than 1.5 we deem the words a collocation. We identify 647 total edits (across train and test sets) which break a collocation; only 11 of such edits are for $\xrightarrow{\text{mwe}}$ relations. Examples include: "forget ~~about it~~"', "kind ~~of~~" and "as ~~well~~".

CoLA. All reported results for the CoLA model use the Real/Fake + LM (ELMo) baseline from Warstadt et al. [341].[19] Across our entire dataset, the mean predicted acceptability of source sentences from the CoLA model is 0.867 (σ=0.264) and the predicted acceptability of compressions is 0.740 (σ = 0.363). We hypothesize that compression scores have a greater variance and a lower mean because only some compressions are well-formed.

Deduplication of possible compressions. In this work, we describe a method for generating multiple compressions from a single sentence. In our generation procedure, it is possible to randomly select the exact same sequence of operations multiple times. During these experiments, we remove any such duplicates from the initial list.

Additionally, in our compression framework, the sequence of operations which produces a given shortening is not unique.[20] In cases where different sequences of operations return the same compression, we select the compression with the highest A(c) score, which represents the best available path to the shortening.

[19] `https://github.com/nyu-mll/CoLA-baselines`

[20] For instance, it is possible to prune a leaf vertex with one operation, and then prune its parent vertex with a second operation; or just remove both vertexes at once via a single prune of the parent. (Each sequence returns the same compression.)

CHAPTER 5

TEXT SIMPLIFICATION (VERTEX ADDITION)

This chapter was originally published as *Query-focused Sentence Compression in Linear Time* [140].

Synopsis

During query-focused lexical corpus analysis (§1.2.1), users review mentions of lexical items in context. However, reading over all mentions of a lexical item in a corpus can be burdensome. Sentences which mention a lexical item may be long and complex, and there may be hundreds or thousands of mentions within a corpus. Therefore, this chapter introduces a new transition-based sentence simplification technique to help users review simplified mentions of a query Q in context.

Our VERTEX ADDITION method takes an English sentence as input, and grows a subgraph of the dependency parse of the sentence to return a shortened output sentence which both contains lexical query terms and adheres to a sentence-level context budget. This theoretically efficient approach achieves a corresponding 11x empirical speedup over a baseline integer linear programming method, while better reconstructing known good shortenings in a benchmark corpus. Such speedups help in user-facing applications such as ROOKIE or CLIOQUERY (Chapters 6 and 7), because users are measurably hindered by interface lags. Additionally, VERTEX ADDITION does not require access to specialized ILP software or an expensive GPU, which is important for practitioners in fields like journalism and history.

5.1 Introduction

In natural language processing, the extractive sentence compression task seeks to create short, readable, single-sentence summaries which retain the most "important" information from source sentences. Extractive sentence compression methods can assist with query-focused lexical corpus analysis (e.g. §7), by simplifying text to help users quickly read over query mentions; provided that shortened sentences include a user's lexical query Q and fit in the screen space available in a user interface. For instance, ROOKIE (§6), CLIOQUERY (§7) and the systems shown in Figure 1.5d and Figure 5.1 each show sentences mentioning Q in particular UIs.

There are no known methods to efficiently shorten sentences so that they always include Q and adhere to a character-level length budget. While techniques based on integer linear programming (ILP) can trivially accommodate such length and lexical restrictions [67, 107, 339], these approaches rely on slow third-party solvers to optimize an NP-hard integer linear programming objective, causing unwelcome user wait time [205]. An alternative LSTM tagging approach [109] does not allow practitioners to specify length or lexical constraints, and requires an expensive graphics processing unit (GPU) to achieve low runtime latency (access to GPUs is a barrier in fields like social science and journalism). These deficits prevent application of existing compression techniques in user interfaces for lexical corpus analysis, where length, lexical and latency requirements are paramount.

Therefore, in this chapter, we examine the English-language compression problem under such length and lexical requirements. In our constrained compression setting, a source sentence S is shortened to a compression C which (1) must include all tokens in a set of lexical query terms Q and (2) must be no longer than a maximum budgeted character length, $b \in \mathbb{Z}^+$. Formally, constrained compression maps $(S, Q, b) \rightarrow C$, such that C respects Q and b. We describe this task as query-focused compression because Q places a hard requirement on words from S which must be included in

C. We then present a new stateful method for query-focused compression, which is theoretically and empirically faster than ILP-based techniques, and more accurately reconstructs gold standard shortenings in a benchmark corpus.

Figure 5.1: An interface for lexical corpus analysis (boxed, top) returns a snippet consisting of three compressions which must contain a users' query Q (bold) and must not exceed $b = 75$ characters in length. The third compression C was derived from source sentence S (italics, bottom).

5.2 Related work

Extractive compression shortens a sentence by removing tokens, typically for text summarization [189, 67, 109, 339].[1] To our knowledge, this work is the first to consider extractive compression under hard length and lexical constraints.

We compare our VERTEX ADDITION approach to ILP-based compression methods [67, 107, 339], which shorten sentences using an integer linear programming objective. ILP methods can easily accommodate lexical and budget restrictions via additional optimization constraints, but require worst-case exponential computation.[2]

Finally, compression methods based on LSTM taggers [109] cannot currently enforce lexical or length requirements. Future work might address this limitation by applying and modifying constrained generation techniques [182, 278, 122].

---

[1]Some methods compress via generation instead of deletion [291, 214]. Our extractive method is intended for practical, interpretable and trustworthy search systems [64]. Users might not trust abstractive summaries [352], particularly in cases of factual error [191].

[2]ILPs are exponential in $|V|$ when selecting tokens [67] and exponential in $|E|$ when selecting edges [109].

Approach	Complexity	Constrained
ILP	exponential	yes
LSTM tagger	linear	no
VERTEX ADDITION	**linear**	**yes**

Table 5.1: Our VERTEX ADDITION technique (§5.3) constructs constrained compressions in linear time. Existing methods (§5.2) have higher computational complexity (ILP) or do not respect hard constraints (LSTM tagger).

5.3 Compression via VERTEX ADDITION

We present a new transition-based method for shortening sentences under lexical and length constraints, inspired by similar approaches in transition-based parsing [258]. We describe our technique as VERTEX ADDITION because it constructs a shortening by *growing* a (possibly disconnected) subgraph in the dependency parse of a sentence, one vertex at a time. This approach can construct constrained compressions with a linear algorithm, leading to 11x lower latency than ILP techniques (§5.4). To our knowledge, our method is also the first to construct compressions by *adding* vertexes rather than *pruning* subtrees in a parse [189, 9, 106], as in §4. Our VERTEX ADDITION assumes a Boolean relevance model; S must contain the user's lexical query Q.

5.3.1 Formal description

VERTEX ADDITION builds a compression by maintaining a state (C_i, P_i) where $C_i \subseteq S$ is a set of added candidate vertexes, $P_i \subseteq S$ is a priority queue of vertexes, and i indexes a timestep during compression. Figure 5.2 shows a step-by-step example.

During initialization, we set $C_0 \leftarrow Q$ and $P_0 \leftarrow S \setminus Q$. Then, at each timestep, we pop some candidate $v_i = h(P_i)$ from the head of P_i and evaluate v_i for inclusion in C_i. (Neighbors of C_i in P_i get higher priority than non-neighbors; we break ties in left-to-right order, by sentence position). If we accept v_i, then $C_{i+1} \leftarrow C_i \cup v_i$; if not, $C_{i+1} \leftarrow C_i$. We discuss acceptance decisions in detail in §5.4.2.2. We continue

adding vertexes to C until either P_i is empty or C_i is b characters long. We linearize C by left-to-right vertex position in S, common for compression in English [107]. Algorithm 5.3.1 shows the steps of VERTEX ADDITION. Figure 5.2 shows a single example.

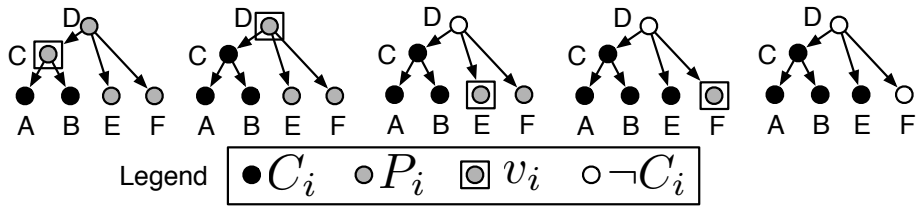


Figure 5.2: A dependency parse of a sentence S, shown across five timesteps of VERTEX ADDITION (from left to right). Each node in the parse is a vertex in S. Our stateful method produces the final compression {A,C,B,E} (rightmost). Each candidate v_i at each timestep is shown with a surrounding square box; rejected candidates $\neg C_i$ are unshaded.

Algorithm 1: The VERTEX ADDITION algorithm. The function ℓ linearizes an unordered set of vertexes to a shortened sentence. The notation $|P|$ indicates the number of tokens in the priority queue.

input: $s = (V, E)$, $Q \subseteq V$, $b \in \mathbb{R}^+$
$C \leftarrow Q; P \leftarrow V \setminus Q;$
while $\ell(C) < b$ *and* $|P| > 0$ **do**
  $v \leftarrow \text{pop}(P);$
  if $p(y = 1) > .5$ *and* $\ell(C \cup \{v\}) \leq b$ **then**
    $C \leftarrow C \cup \{v\}$
  end
end
return $\ell(C)$

VERTEX ADDITION is linear in the token length of S because we pop and evaluate some vertex from P_i at each timestep (and set $P_0 \leftarrow S \setminus Q$). Additionally, because (1) we never accept v_i if the length of $C_i \cup v_i$ is more than b, and (2) we set $C_0 \leftarrow Q$, our method always respects Q and b.

5.4 Evaluation

We observe the latency, readability and token-level F1 score of VERTEX ADDITION, using a standard dataset for sentence compression [107]. We compare our method to an ILP baseline (§5.2) because ILP methods are the only known technique for constrained compression. All methods have similar compression ratios (shown in Appendix §5.7), a well-known evaluation requirement [245]. We evaluate the significance of differences between our CPU-based VERTEX ADDITION and the ILP with bootstrap sampling [34]. All differences are significant ($p < .01$).

5.4.1 Constrained compression experiment

In order to evaluate different approaches to constrained compression, we require a dataset of sentences, constraints and known-good shortenings, which respect the constraints. This means we need tuples (S, Q, b, C_g), where C_g is a known-good compression of S which respects Q and b (§5.1).

To support large-scale automatic evaluation, we reinterpret a standard compression corpus [107] as a collection of input sentences and constrained compressions. The original dataset contains pairs of sentences S and compressions C_g, generated using news headlines. For our experiment, we set b equal to the character length of the gold compression C_g. We then sample a small number of nouns[3] from C_g to form a query set of lexical items Q, approximating both the observed number of tokens and observed parts of speech in real-world search [170, 26]. Sampled Q include reasonable lexical queries like "police, Syracuse", "NHS" and "Hughes, manager, QPR".

By sampling queries and defining budgets in this manner, we create 198,570 training tuples and 9949 test tuples, each of the form (S, Q, b, C_g). Filippova and Altun [107] define the train/test split. We re-tokenize, parse and tag with CoreNLP v3.8.0 [217]. We reserve 25,000 training tuples as a validation set.

[3] 1 to 3 nouns; cardinality chosen uniformly at random.

5.4.2 Models

5.4.2.1 ILP

We compare our system to a baseline ILP method, presented in Filippova and Altun [107]. This approach represents each edge in a syntax tree with a vector of real-valued features, then learns feature weights using a structured perceptron trained on a corpus of (S, C_g) pairs.[4] Learned weights are used to compute a global compression objective, subject to structural constraints which ensure C is a valid tree. This baseline can easily perform constrained compression: at test time, we add optimization constraints specifying that C must include Q, and not exceed length b.

To our knowledge, a public implementation of this method does not exist. We reimplement from scratch using Gurobi Optimization [133], achieving a test-time, token-level F1 score of 0.76 on the unconstrained compression task, lower than the result (F1 = 84.3) reported by the original authors. There are some important differences between our reimplementation and the original approach which might explain the discrepancy (described in detail in the Appendix §5.7). Since VERTEX ADDITION requires Q and b, we can only compare it to the ILP on the *constrained* (rather than traditional, unconstrained) compression task.

5.4.2.2 Vertex Addition

Vertex addition accepts or rejects some candidate vertex v_i at each timestep i. We learn such acceptance decisions $y_i \in \{0, 1\}$ using a corpus of tuples (S, Q, b, C_g), described in §5.4.1. Given such a tuple, we can always execute an oracle path shortening S to C_g by first initializing VERTEX ADDITION and then, at each timestep: (1) choosing $v_i = h(P_i)$ and (2) adding v_i to C_i iff $v_i \in C_g$. We set $y_i = 1$ if $v_i \in C_g$; we set $y_i = 0$ if $v_i \notin C_g$. We then use decisions from oracle paths to train a model of inclusion

[4] Another ILP [339] sets weights using a LSTM, achieving similar in-domain performance. This method requires a multi-stage computational process (i.e. run a LSTM *then* an ILP) that is poorly-suited to query-focused settings, where low latency is crucial.

decisions, $p(y_i = 1|v_i, C_i, P_i, S)$. At test time, we accept v_i if $p(y_i > .5)$. We implement $p(y_i|v_i, C_i, P_i, S)$ in two ways, VERTEX ADDITION$_{NN}$ and VERTEX ADDITION$_{LR}$, described below.

Model One. Our VERTEX ADDITION$_{NN}$ model broadly follows neural approaches to transition-based parsing (e.g. Chen and Manning [57]): we predict y_i using a LSTM classifier with a standard max-pooling architecture [69], implemented with a common neural network framework [120]. Our classifier maintains four vocabulary embeddings matrixes, corresponding to the four disjoint subsets $C_i \cup \neg C_i \cup P_i \cup \{v_i\} = V$. Each LSTM input vector x_t comes from the appropriate embedding for $v_t \in V$, depending on the state of the compression system at timestep i. The appendix details network tuning and optimization.

Model Two. Our VERTEX ADDITION$_{LR}$ model uses binary logistic regression,[5] with 3 classes of features.

Edge features describe the properties of the edge (u, v_i) between $v_i \in P_i$ and $u \in C_i$. We use the edge-based feature function from Filippova and Altun [107], described in detail in the Appendix §5.7. This allows us to compare the performance of a vertex addition method based on local decisions with an ILP method that optimizes a global objective (§5.4.4), using the same feature set.

Stateful features represent the relationship between v_i and the compression C_i at timestep i. Stateful features include information such as the position of v_i in the sentence, relative to the right-most and left-most vertex in C_i, as well as history-based information such as the fraction of the character budget used so far. Such features allow the model to reason about which sort of v_i should be added, given Q, S and C_i.

[5] We implement with Python 3 using scikit-learn [272]. We tune the inverse regularization constant to $c = 10$ via grid search over powers of ten, to optimize validation set F1.

Interaction features are formed by crossing all stateful features with the type of the dependency edge governing v_i, as well as with indicators identifying if u governs v_i, if v_i governs u or if there is no edge (u, v_i) in the parse.

5.4.3 Metrics: F1, Latency and SLOR

We measure the token-level F1 score of each compression method against gold compressions in the test set. F1 is the standard automatic evaluation metric for extractive compression [109, 188, 339].

In addition to measuring F1, researchers often evaluate compression systems with human *importance* and *readability* judgements [189, 109]. In our setting Q determines the "important" information from S, so importance evaluations are inappropriate. To check readability, we use the automated readability metric SLOR [194], which is known to correlate with human judgements [179].

We check the theoretical gains from VERTEX ADDITION (Table 5.1) by measuring empirical latency. For each compression method, we sample and compress $N = 300,000$ sentences, and record the runtime (in milliseconds per sentence). Because we observe that runtimes are distributed log-normally (Figure 5.3), we summarize each sample using the geometric mean. ILP and VERTEX ADDITION$_{LR}$ share edge feature extraction code to support to fair comparison. We test VERTEX ADDITION$_{NN}$ using a CPU (Table 5.2), to test performance for users without access to specialized hardware. The Appendix §5.7 further details latency and SLOR experiments.

5.4.4 Comparisons: Ablated & Random

For comparison, we also implement an ABLATED vertex addition method, which learns inclusion decisions using only edge features from Filippova and Altun [107]. ABLATED has a lower F1 score than ILP, which uses the same edge-level information to optimize a global objective: adding stateful and interaction features (i.e. VERTEX ADDITION$_{LR}$) improves F1 score. Nonetheless, strong performance from ABLATED

hints that edge-level information alone (e.g. dependency type) can mostly guide acceptance decisions.

We also evaluate a RANDOM baseline, which accepts each v_i randomly in proportion to $p(y_i = 1)$ across training data. RANDOM is a strong F1 baseline because (1) $C_0 = Q \in C_g$ and (2) F1 correlates with compression rate [245], and b is set to the length of C_g.

Approach	F1	SLOR	*Latency
RANDOM (lower bound)	0.653	0.377	0.5
ABLATED (edge only)	0.827	0.669	3.7
VERTEX ADDITION$_{NN}$	0.873	0.728	2929.1 (CPU)
ILP	0.852	0.756	44.0
VERTEX ADDITION$_{LR}$	0.881	0.745	4.1

Table 5.2: Test results for constrained compression. *Latency is the geometric mean of observed runtimes (in milliseconds per sentence). VERTEX ADDITION$_{LR}$ achieves the highest F1, and also runs 10.73 times faster than the ILP. Differences between all scores for VERTEX ADDITION$_{LR}$ and ILP are significant ($p < .01$).

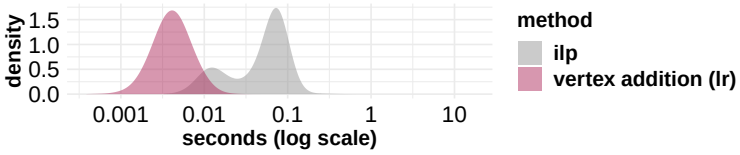


Figure 5.3: Density plot of log transformed latencies for VERTEX ADDITION$_{LR}$ (left) and ILP (right). Theoretical gains (Table 5.1) create real wall clock speedups. The ILP shows greater runtime variance, possibly reflecting varying approaches from Gurobi Optimization [133].

5.5 Future work: VERTEX ADDITION in practice

This work presents a new method for fast, query-focused, sentence compression, motivated by lexical corpus analysis. While our approach shows promise in simulated experiments, we expect that further work will be required before the method can be employed for practical, user-facing search.

To begin, both our technique and our evaluation ignore the conventions of search user interfaces, which typically display missing words using ellipses. This convention is important, because it allows snippet systems to transparently show users which words have been removed from a sentence. However, we observe that some well-formed compressions are difficult to read when displayed in this format. For instance the sentence "Aristide quickly fled Haiti in September 1991" can be shortened to the well-formed compression "Aristide fled in 1991." But this compression does not read fluidly when using ellipses ("Aristide...fled...in...1991"). Human experiments aimed at enumerating the desirable and undesirable properties of compressions displayed in ellipse format (e.g. compressions should minimize number of ellipses?) could help guide user-focused snippet algorithms in future work.

Our method also assumes access to a reliable, dependency parse, and ignores any latency penalties incurred from parsing. In practical settings, both assumptions are unreasonable. Like other NLP tools, dependency parsers often perform poorly on out of domain text [22], and users looking to quickly investigate a new corpus might not wish to wait for a parser. Faster approaches based on low-latency part-of-speech tagging, or more cautious approaches based on syntactic uncertainty [181], each offer exciting possibilities for additional research.

Our approach also assumes that a user already knows a reasonable b and reasonable Q for a given sentence S.[6] However, in some cases, there is no well-formed shortening of which respects the requirements. For instance, if Q="Kennedy" and b=15 there is no reasonable shortening for the toy sentence "Kennedy kept running", because the compressions "Kennedy kept" and "Kennedy running" are not well-formed. We look forward to investigating which (Q, S, b) triples will never return well-formed compressions in later work.

[6]Recall that we simulate b and Q based on the well-formed shortening C_g, see §5.4.1.

Finally, some shortened sentences will modify the meaning of a sentence, but we ignore this important complication. In the future, we hope to apply ongoing research into textual entailment [42, 271, 227] to develop semantically-informed approaches to the task.

5.6 Conclusion

We introduce a query-focused VERTEX ADDITION method for shortening sentences containing one or more query words Q. Our method has much lower theoretical complexity (and empirical runtimes) than baseline techniques, while achieving similar performance in reconstructing known good sentence shortenings. VERTEX ADDITION thus takes steps towards our goal of developing computationally-efficient methods for lexical corpus analysis (§1.5).

5.7 Appendix

5.7.1 Neural network tuning and optimization

We learn network parameters for VERTEX ADDITION$_{NN}$ by minimizing cross-entropy loss against oracle decisions y_i. We optimize with ADAGRAD [93]. We learn input embeddings after initializing randomly. The hyperparameters of our network and training procedure are: the learning rate, the dimensionality of input embeddings, the weight decay parameter, the batch size, and the hidden state size of the LSTM. We tune via random search [35], selecting parameters which achieve highest accuracy in predicting oracle decisions for the validation set. We train for 15 epochs, and we use parameters from the best-performing epoch (by validation accuracy) at test time.

Parameter	Value
Batch size	135
Embedding dim.	315
Hidden dim.	158
Learning rate	0.025
Weight decay	1.88×10^{-9}

Table 5.3: Hyperparameters for VERTEX ADDITION$_{NN}$

5.7.2 Reimplementation of Filippova and Altun

In this work, we reimplement the method of Filippova and Altun [107], who in turn implement a method partially described in Filippova and Strube [108]. There are inevitable discrepancies between our implementation and the methods described in these two prior papers.

1. Where the original authors train on only 100,000 sentences, we learn weights with the full training set to compare fairly with VERTEX ADDITION (each model trains on the full training set).

2. We use Gurobi Optimization [133] (v8) to solve the integer linear program. Filippova and Strube [108] report using LPsolve.[7]

3. We implement with the common Universal Dependencies (UD, v1) framework [259]. Prior work [108] implements with older dependency formalisms [47, 84].

4. In Table 1 of their original paper, Filippova and Altun [107] provide an overview of the syntactic, structural, semantic and lexical features in their model. We implement every feature described in the table. We do not implement features which are not described in the paper.

[7] `http://sourceforge.net/projects/lpsolve`

5. Filippova and Altun [107] augment edge labels in the dependency parse of S as a preprocessing step. We reimplement this step using off-the-shelf augmented modifiers and augmented conjuncts available with the enhanced dependencies representation in CoreNLP [296].

6. Filippova and Altun [107] preprocess dependency parses by adding an edge between the root node and all verbs in a sentence.[8] We found that replicating this transform literally (i.e. only adding edges from the original root to all tokens tagged as verbs) made it impossible for the ILP to recreate some gold compressions. (We suspect that this is due to differences in output from part-of-speech taggers). We thus add an edge between the root node and *all* tokens in a sentence during preprocessing, allowing the ILP to always return the gold compression.

We assess convergence of the ILP by examining validation F1 score on the traditional sentence compression task. We terminate training after six epochs, when F1 score stabilizes changes by fewer than 10^{-3} points.

5.7.3 Implementation of SLOR

We use the SLOR function to measure the readability of the shortened sentences produced by each compression system. SLOR normalizes the probability of a token sequence assigned from a language model by adjusting for both the probability of the individual unigrams in the sentence and for the sentence length.[9]

Following Lau et al. [194], we define the function as:

---

[8]This step ensures that subclauses can be removed from parse trees, and then merged together to create a compression from different clauses of a sentence.

[9]Longer sentences are always less probable than shorter sentences; rarer words make a sequence less probable.

$$\text{SLOR} = \frac{\log P_m(\xi) - \log P_u(\xi)}{|\xi|} \tag{5.1}$$

where ξ is a sequence of words, $P_u(\xi)$ is the unigram probability of this sequence of words and $P_m(\xi)$ is the probability of the sequence, assigned by a language model. $|\xi|$ is the length (in tokens) of the sentence.

To define language model probabilities, we use a 3-gram model trained on the training set of the Filippova and Altun [107] corpus. We implement with KenLM [148]. Because compression often results in shortenings where the first token is not capitalized (e.g. a compression which begins with the third token in S) we ignore case when calculating language model probabilities.

5.7.4 Latency evaluation

To measure latency, for each technique, we sample 100,000 sentences with replacement from the test set. We observe the mean time to compress each sentence using Python's built-in *timeit* module. In order to minimize effects from unanticipated confounds in measuring latency, we repeat this experiment three separate times (with a one hour delay between experiments). Thus in total we collect 300,000 observations for each compression technique. We observe that runtimes are log normal, and thus report each latency as the geometric mean of 300,000 observations. We use an Intel Xeon processor with a clock rate of 2.80GHz.

5.7.5 Compression ratios

When comparing sentence compression systems, it is important to ensure that all approaches use the same rate of compression [245]. Following Filippova et al. [109], we define the compression ratio as the character length of the compression divided by the character length of the sentence. We present test set compression ratios for all methods in Table 5.4. Because ratios are similar, our comparison is appropriate.

RANDOM	0.405
ILP	0.408
ABLATED	0.387
VERTEX ADDITION$_{LR}$	0.403
VERTEX ADDITION$_{NN}$	0.405
C_g Train	0.384
C_g Test	0.413

Table 5.4: Mean test time compression ratios for all techniques. We also show mean ratios for gold compressions C_g across the train and test sets.

PART III: HOW TO DESIGN LEXICAL SYSTEMS FOR SPECIFIC USER GROUPS

CHAPTER 6

ROOKIE

This chapter is adapted from *Rookie: A unique approach for exploring news archives* [138].

Synopsis

In this section, we describe the development, implementation and evaluation of ROOKIE, a tool for lexical corpus analysis designed for journalists, readers and editors learning about new topics from news archives. ROOKIE facilitates exploratory lexical corpus analysis by drawing the user's attention to unusually frequent noun phrases from a corpus vocabulary (applying methods from Chapter 2), and also facilitates query-focused lexical corpus analysis by helping users quickly investigate such noun phrases in context. (Exploratory and query-focused lexical analysis are defined in §1.2.1)

Our effort designing and testing ROOKIE offers a case study in building user-facing lexical systems. It also shows one way in which current search interfaces are imperfect tools for making sense of broad narratives across individual archived documents; we observe that ROOKIE users complete an archive-based sensemaking task 37% faster than other users assigned to complete the same task using a traditional search user interface. This finding inspires our search for alternative text user interfaces, which may also assist other users in other domains.

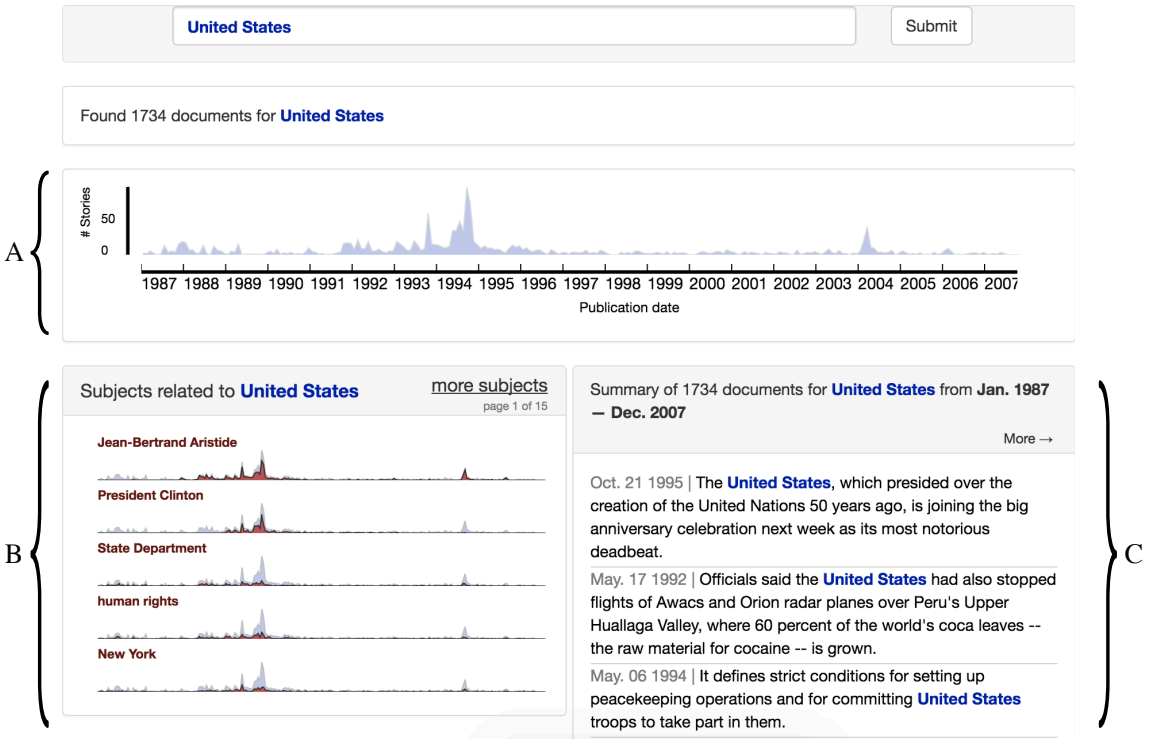


Figure 6.1: The ROOKIE interface running on a corpus of *New York Times* articles about Haiti. The user has queried for "United States." The interface features three linked visualization and summarization views: (A) an interactive timeline, (B) a *Subjects Summary* showing automatically-generated related subjects, and (C) a *Snippet Summary* showing sentence summaries. The temporal spikes indicate major events such as a 1994 U.S. intervention in Haiti and a 2004 military coup. Related subjects include specific actors in some of these events (Jean-Bertrand Aristide, President Clinton) as well as long-running topics (human rights). Users can click and drag along the timeline to investigate specific time periods.

6.1 Introduction

News archives offer a rich historical record. But if a reader or journalist wants to learn about a new topic with a traditional search engine, they must enter a query and begin reading or skimming old articles one-by-one, slowly piecing together the intricate web of people, organizations, events, places and concepts that make up "the news."

To help such users identify broader stories across documents, we propose ROOKIE, which began as an attempt to build a useful tool for journalists. With ROOKIE, a

user's query generates an interactive timeline, a list of important subjects drawn from the corpus lexicon, and a summary of a user-selected lexical item, all displayed together as a collection of interactive linked views (Figure 6.1). Quantitative user testing shows that such features help ROOKIE users correctly complete a historical sensemaking task 37% faster than users assigned to complete the same task with a traditional search user interface. Additionally, qualitative user testing suggests how ROOKIE might help users fluidly investigate complex news stories as they evolve across time.

More broadly, throughout this thesis, we argue and strive for a number of design goals in user-facing lexical systems (Section 1.5). Many of these goals emerged from our work on ROOKIE; because we developed ROOKIE for real-world journalists, we were forced to cope with limitations in the speed, accuracy and interpretability of current natural language processing techniques. We describe these limitations in NLP methods, and our efforts to design around them, in Section 6.4.

6.2 The ROOKIE system

The ROOKIE interface always reflects the **user selection state**, a triple (Q, Q', T) where:

- Q is a free text query string (e.g. "Bashar al-Assad")
- Q' is a related subject string (e.g. "30 years") or is `null`
- T is a timespan (e.g. Mar. 2000–Sep. 2000); by default, this is set to the span of publication dates in the corpus.

Users first interact with ROOKIE by entering a query, Q into a search query bar using a web browser. For example, in the Figure below, a user seeking to understand the roots of the Syrian civil war has entered Q = "Bashar al-Assad". In response, ROOKIE renders an interactive time series visualization showing the frequency of

query-matching documents from the corpus (§6.2.4), a list of subjects in the matching documents called *Subjects Summary* (§6.2.2) and a textual summary of those documents called *Snippet Summary* (§6.2.3).

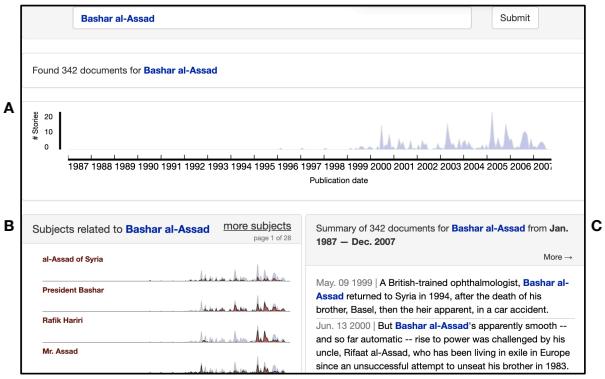


Figure 6.2: A user searches for Q ="Bashar al-Assad". ROOKIE shows: (**A**) a time series visualization, (**B**) *Subjects Summary* (a lexical view), and (**C**) *Snippet Summary* (a text view). In this figure, Q' is null.

After entering Q, the user might notice that "Bashar al-Assad" is mainly mentioned from early 2000 onwards. To investigate, they might adjust the time series slider to select T = Mar. 2000–Sep. 2000 (Figure 6.3).

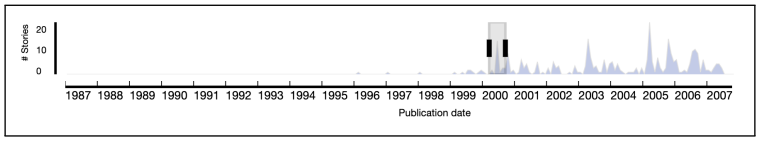


Figure 6.3: The user zooms to T = Mar. 2000–Sep. 2000.

When the user adjusts T to Mar. 2000–Sep. 2000, *Snippet Summary* and *Subjects Summary* change to reflect the new timespan (Figure 6.4). *Subjects Summary* now shows subjects like "President Assad", "TRANSITION IN SYRIA",[1] and "Hafez al-

[1]In this example, the corpus is a collection of *New York Times* world news articles from 1987 to 2007 that contain the string "Syria". All of the country-specific examples in this study are subsets of the same *New York Times* (NYT) LDC corpus [292]. Capital letters are from NYT section headers.

Assad" which are important to Q during T. (Bashar al-Assad succeeded Hafez al-Assad in the year 2000.)

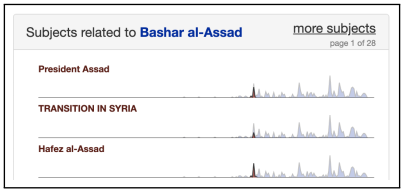


Figure 6.4: *Subjects Summary* shows important subjects for Q ="Bashar al-Assad" during T = Mar. 2000–Sep. 2000.

At this point, the user might explore further by investigating the related subject, Q' ="Hafez al-Assad". When they click to select, *Snippet Summary* attempts to summarize the relationship between Q ="Bashar al-Assad" and Q' ="Hafez al-Assad" during T = Mar. 2000–Sep. 2000 (see Figure 6.5) If the user wants to understand any sentence from *Snippet Summary* in context, they can click to open the underlying document in a modal dialog.

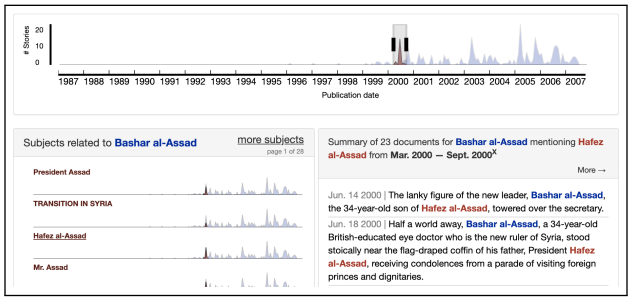


Figure 6.5: ROOKIE now adds mentions of Q' ="Hafez al-Assad" to the time series graph. *Snippet Summary* updates to reflect the updated state, (Q ="Bashar al-Assad", Q' ="Hafez al-Assad", T = Mar. 2000–Sep. 2000).

Q' and Q are assigned red and blue colors throughout the interface, allowing users to quickly scan for information. Bolding Q and Q' gives additional clarity, and helps ensure that ROOKIE still works for colorblind users.

This example demonstrates how ROOKIE's visualization and summarization techniques work together to offer linked views of the underlying corpus. Linked views

(a.k.a. multiple coordinated views) interfaces are common tools for structured information [49, 332, 262]: each view displays the same selected data in a different dimension. For instance, a linked view system might show a geographic map of a city in one view, and update a histogram of housing values in a second view when a user clicks a zip code. In ROOKIE's case, linked views display different levels of resolution. The time series visualization offers a **temporal view** of query-responsive documents, *Subjects Summary* displays a medium-level **lexical view** of important subjects within the documents, and *Snippet Summary* displays a low-level **text view** of parts of the underlying documents. The documents themselves (shown when the user clicks extracted sentences) offer the most detailed level of zoom. Thus ROOKIE supports the commonly advised visualization pathway: "overview first, zoom and filter, and details on demand" [308].

Note that we use the term **summarization** to mean selecting a short textual span, or a sequence of short textual spans, to represent a body of text. By this definition, both *Subjects Summary* and *Snippet Summary* are a form of summarization, as each offers a textual representation of the corpus, at two different levels of resolution (phrases and sentences). In the NLP literature, "summarization" usually means generating a sentence or paragraph length summary [82].

ROOKIE is a web application implemented in Python.[2]

6.2.1 Linked views in the ROOKIE system

ROOKIE's **user selection state** (Q, Q', T) picks out a set of documents $\mathbb{D}_{(Q,Q',T)}$, which were published within T, match the query Q in Whoosh and contain Q' (if Q' is not `null`). The selection state also specifies a set of sentences $\mathbb{S}$, used to construct

[2]We use the Flask (`http://flask.pocoo.org/`) framework with a Postgres (`https://www.postgresql.org/`) database and a React front end (`https://facebook.github.io/react`). We used the open-source search engine Whoosh (`https://whoosh.readthedocs.io`), which is broadly similar to Lucene, to find documents matching Q.

a summary (§6.2.3). These documents and sentences are then shown to the user in the linked views, described individually in the following sections.

6.2.2 Lexical view: *Subjects Summary*

ROOKIE uses natural language processing methods to find and recommend a list of subjects related to the query Q, during time T. These subjects are presented as a concise list of terms, offering a *lexical* view of the $\mathbb{D}_{(Q,T)}$ selection (Figure 6.1, bottom left).

ROOKIE's subject-finding algorithm works in two stages. At index time, ROOKIE makes a single pass over the corpus to find and record all phrases which match certain part-of-speech patterns. Specifically, ROOKIE uses the NPFST method (§2) to extract phrases, which ROOKIE stores in a document–phrase index.[3] Then, at query time, ROOKIE uses this index to rank phrases which occur in documents responsive to Q, returning top-ranked phrases as subjects for display in the UI.

Each time a user changes Q or T, ROOKIE identifies all phrases which occur in the matching documents $\mathbb{D}_{(Q,T)}$. ROOKIE then assigns each phrase a subject relevance score. Relevance scores for each subject s are calculated with $\text{QF-IDF}_s = q_s * \frac{1}{df_s}$ where the first term, q_s ("query frequency"), is a count of how many times term s occurs in $\mathbb{D}_{(Q,T)}$ and the second term, $\frac{1}{df_s}$ ("inverse document frequency"), is the inverse of the number of documents which contain s across the corpus. Highly relevant phrases occur frequently in query-matching documents, $\mathbb{D}_{(Q,T)}$, but less frequently overall, similar to pointwise mutual information [262]. ROOKIE places such high-ranking phrases at the top of *Subjects Summary*.

Note that NP extraction often produces split or repeating phrases (§2) such as "King Abdullah", "Abdullah II" and "King Abdullah II". ROOKIE uses several simple

[3]We only index subjects that occur at least five times in the corpus for use in subject list generation, though document retrieval for Q utilizes a standard full text index.

hand-written string-matching rules based on character-level Levenshtein distance and token-level Jaccard similarity to avoid displaying duplicate terms.

6.2.3 Text view: *Snippet Summary*

ROOKIE's time series visualizations offer an immediate question: what does Q have to do with Q' during T? For example, in Figure 6.1, the user might wish to learn: what does "United States" have to do with the phrase "human rights" in articles about Haiti from the early 90s? ROOKIE attempts to answer using extractive summarization, picking sentences from $\mathbb{D}_{(Q,Q',T)}$ that can explain the relationship. Unlike in traditional NLP, in ROOKIE the goal is not just to summarize some topic expressed by Q (as in traditional query-focused summarization [82]), but to describe what Q' has to do with Q during T. (ROOKIE does not use relationship summarization3, but this might be applied in future versions of the system.) ROOKIE also requires that any summary can be produced quickly enough to support interactive search (see §6.4).

Thus, in building ROOKIE, we found it useful to require that the client be able to generate a summary in less than half a second without server communication.[4] This principle allowed us to achieve fluid, exploratory interactions. The details of *Snippet Summary* are discussed below.

6.2.3.1 Summary implementation: server side

After ROOKIE sends a user query, Q to the server, each query-responsive document in $\mathbb{D}_{(Q,Q',T)}$ is permitted to send exactly one sentence to the client. This sentence is selected via a two-tiered priority queue of sentences (from a given document). Each sentence's tier 1 score in the queue records if the sentence contains both Q and Q' (top priority), Q or Q' (medium priority) or neither Q nor Q' (low priority). The sentence's tier 2 score is simply the sentence's sequential number in the document.

[4]Half a second is a rough rule of thumb for acceptable latency in interactive systems, informed by work from both Nielsen [255] and Liu and Heer [205].

(Sentences that come earlier in the document get higher priority). ROOKIE pops the first sentence in each document's queue to sent to the client, along with its publication date, sentence number and tier 1 score. We use $\mathbb{S}$ to denote the set of sentences passed to the client.

6.2.3.2 Summary implementation: client side

ROOKIE seeks to help explain what *happened* during a particular timespan, T. So where traditional summarization seeks topical diversity [82], ROOKIE aims for temporal diversity. It achieves this diversity by sampling sentences with probability in proportion to the count of each monthly bin. For instance, if $\mathbb{S}$ contains 1000 sentences and 100 of them come from March of 1993, then there ought to be a one in ten chance that a sentence from March 1993 is included in the summary.

In choosing sentences for the summary, ROOKIE will first pick randomly from among sentences containing Q and Q', then pick randomly in proportion to publication date from among sentences that contain Q or Q' and, finally, pick randomly from sentences containing from neither Q nor Q'.

ROOKIE draws sentences, one-by-one, until each sentence from $\mathbb{S}$ is selected and placed into a list. ROOKIE allows users to page through this list (Figure 6.1), starting from highest-ranked and moving towards lowest-ranked.

6.2.4 Temporal view: Interactive time series

ROOKIE's time series visualization is a standard line graph showing both $\mathbb{D}_{(Q)}$ and $\mathbb{D}_{(Q,Q')}$ across the time variable. The y-axis represents counts of documents and the x-axis represents time. For instance, in Figure 6.1, the blue line shows counts of documents which match $Q =$ "United States". A small copy of the time series graph for each subject is shown in the subject list (Figure 6.1). These small graphs (sometimes called "sparklines" [327]) give cues about a subject's importance at difference times in a corpus, even if the subject is not selected.

The time series graph allows the user to specify a desired time range T. Users can select particular areas of the time series graph by clicking and dragging a timebox [159] to create specialized summaries of certain time periods. If the user holds down their mouse, clicks the grey rectangle, and slides the mouse across the timeline, the user state changes to reflect the new T. *Subjects Summary* and *Snippet Summary* show the evolving relationship through time.

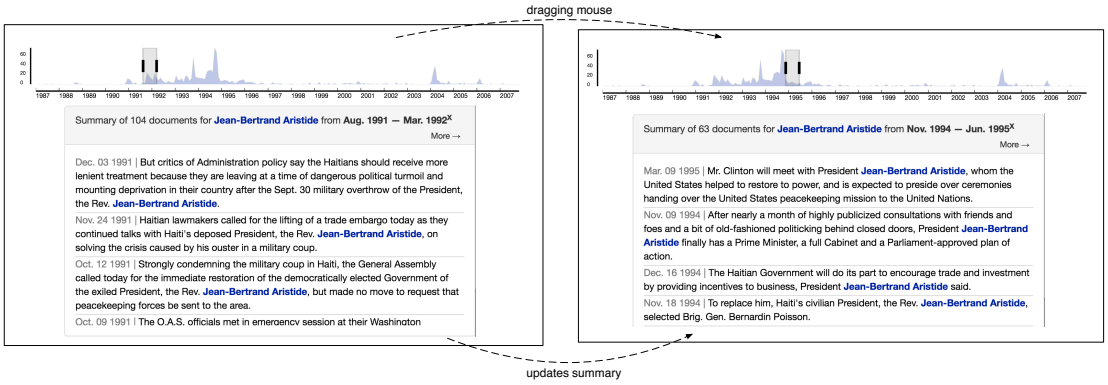


Figure 6.6: ROOKIE's summaries and time series offer linked views of the corpus. The *Snippet Summary* panel updates in less than half a second, so users can drag a cursor across the timeline to read about unfolding events. In this example, as the user drags the rectangle to the right, they read about the 1991 coup that removed Bertrand Aristide from power, and then his return several years later with American support.

6.3 Evaluation

We evaluate ROOKIE using several established practices for evaluating exploratory search [342, 343] including (1) surveys and questionnaires to measure user experience and (2) quantitative measurements of human performance in completing a search task.[5]

[5]All studies were approved by IRB.

Figure 6.7: A baseline interface "IR" interface for exploring news archives, similar to the search functionality found on many news websites. Users enter a (Q,T) query pair and the system returns a list of document–snippet pairs. We compare ROOKIE to this baseline interface.

For each evaluation, we compared ROOKIE to a traditional search engine, the baseline tool for answering any question from a collection of documents.

Traditional information retrieval (IR) systems return a ranked list of document–snippet pairs in response to a user's textual query. Users read or skim these snippets and documents until they better understand some aspect of the corpus, possibly re-querying for new documents during the search. We implemented the IR baseline using Whoosh. Because ROOKIE allows limiting documents by date, we also added a frequently-downloaded datepicker widget so that IR users can limit results to date ranges. [6]

[6]Like other traditional search engines, Whoosh creates small snippets which highlight portions of each query-responsive document, boldfacing matching unigrams from the query. We tuned Whoosh snippets by adjusting the *top* and *surround* parameters to help the IR system fairly compete with ROOKIE, which displays whole sentences. *Top* controls how many "..."-delimited fragments Whoosh returns for each document result, while *surround* is the maximum number of characters around each highlighted text fragment. By default, Whoosh sets *surround*=20, but we found that this made for choppy, confusing snippets, so we adjusted it to 50. We then did a grid search over possible values of the *top* parameter—seeking the value that minimized the average absolute difference in the number of characters shown for each snippet between ROOKIE versus Whoosh, arriving at *top*=2. All other Whoosh parameters were set to default values. We used react-datepicker for the datepicker widget `https://www.npmjs.com/package/react-datepicker`

We compared to a traditional search engine for a number of reasons: users are familiar with Google (but not with alternatives), there are few robust implementations of text analytics research systems available and previous work rarely compares to traditional search, which we believe is a robust and powerful baseline.

6.3.1 In-person group evaluation

While building ROOKIE, we solicited ongoing feedback from working journalists. Once we were confident in the final design, we conducted a larger and more formal user test with 15 undergraduate journalism students. Undergraduate journalism students are a good choice for a user group, as ROOKIE is built for reporters and readers learning about new topics. (An additional qualitative evaluation with professional journalists would have improved our study).

During the study, we loaded ROOKIE with a corpus of 5496 articles from the *New York Times* from 1987 to 2007 which mentioned the country "Syria". After a short tutorial demo, we presented the users with a exploratory search prompt: "Imagine you just got your first job as a fact checker and production assistant on a world news desk. Your organization frequently covers the civil war in Syria. Use ROOKIE to better understand the roots of the Syrian civil war so that you can begin contributing at your new job".

We gave users twenty minutes to try ROOKIE using this prompt; then presented a questionnaire about their subjective experience. We did not to tell users about the design intentions behind ROOKIE. We synthesize answers to each question below.

Q1: Did you enjoy using Rookie? What was good about it? Or bad about it?

Users overwhelmingly reported that they "really enjoyed using this tool" and found it useful "extremely useful in doing research". One user said: "It made me feel like I

could find things that may be buried on a more generic search engine". Another said: "It makes a fluid way to search through a lot of information quickly".

Q2 "How do you think something like [Rookie] could help journalists?"

Many users reported that ROOKIE could be helpful for journalists or other researchers starting to learn about a new topic, which was our intention in designing ROOKIE. One wrote: "As journalists, it's important to have a large-view grasp of a story before writing about it. The system could be helpful in providing both a snapshot and an ability to then dive deeper into your story".

Q3 "When would Rookie be better than using a traditional search engine? When would it be worse?"

At the end of the user study, students tried researching the same topic with an IR iterface loaded with the same corpus. We asked which tool would be better and when. Many mentioned that ROOKIE would be superior if you were starting out researching a new topic, but that a traditional search engine would be superior if you already had a clear search need. As one student wrote in praising ROOKIE: "If you aren't ... familiar about the history of the topic you probably want to build some context first".

6.3.2 Task completion evaluation

6.3.2.1 Historical sensemaking task

Gary Marchionini [219] distinguishes between simple fact-finding tasks and exploratory search, which involves activities like comprehension, interpretation and synthesis. These activities are difficult to measure, but a simple and direct way to test how well an exploratory system supports them is to record how long it takes a person

to accomplish a sensemaking task which requires these behaviors. This method is ometimes called measuring "task time" [342].[7]

We thus measured how long it takes users to correctly answer the same complex, non-factoid research question when using ROOKIE and the IR system. The question asks a question about the complex historical relationship between the United States and the Haitian political figure Jean-Bertrand Aristide. We asked users to pick the correct answer from among these four: (1) The United States has been a longtime opponent of the Haitian President Jean-Bertrand Aristide. (2) The United States has been a longtime ally of the Haitian President Jean-Bertrand Aristide. (3) The United States was initially an ally of Bertrand Aristide – but then stopped supporting him. (4) The U.S. government was initially an opponent of Bertrand Aristide, but then started supporting him.

The third answer is broadly correct, within the timespan of the corpus: the Clinton administration used U.S. troops to restore Bertrand Aristide's democratically-elected government following a coup in the mid 1990s. Then, ten years later, the Bush administration did not support Aristide during a later coup.

Users were asked to answer this question by searching for information *New York Times* articles which mention "Haiti." Within this corpus, articles exist describing events in this historical relationship, but there does not appear to be a complete narrative summary of this history in a single document. Users have to sort through, comprehend, and synthesize many pieces of information across multiple articles until they know the correct answer. The task took up to 21 minutes to complete (36 seconds minimum, 1261 maximum) and was fairly difficult: only 52% IR users and 54% ROOKIE users answered correctly.

---

[7]White and Roth [342, Chp. 5] note that ideally exploratory search evaluations would also measure depth of learning and understanding; however, task time is a good place to start.

ROOKIE's evaluation simulates a practical task that a journalist might undertake in learning about a new subject: either to write an "explainer" piece[8] or to research the historical context for current events.

6.3.2.2 Experiment design

We employed a between-subjects design with U.S. users from Amazon Mechanical Turk, placing fifty workers into a ROOKIE group and fifty workers into an IR group and comparing their task completion times and other behaviors. Turkers had a maximum of 30 minutes to complete the task, much more than the roughly 15 minutes required.[9] We limited our study to U.S.-based users.

For each group, the study began with a few short screening questions, checking to make sure that workers had their volume turned on and were using a laptop or desktop (this version of ROOKIE is designed for these modalities). ROOKIE users also practiced interpreting a timeseries graph by explaining why mentions of Afghanistan might have spiked in the *New York Times* in 2001 and 2002. Each group then watched a short video explaining their interface and task.

During iterative prototyping (§6.4), we observed that it takes a few minutes to learn to use ROOKIE. Thus the final preliminary phase for the ROOKIE group was a practice session to learn the ROOKIE interface on a different corpus of articles mentioning "Cuba". During this tutorial, users practiced manipulating T to select T = 1994–1995 and then answered a question about the U.S and Fidel Castro during this time period. After this session was complete, users then were given the main task with the Haiti corpus. This helped ensure the main task was measuring how long it took users to find answers using ROOKIE, as opposed to learning how to use it. IR users did not practice using their interface.

---

[8] e.g. `http://www.vox.com/2015/9/14/9319293/syrian-refugees-civil-war`

[9] Amazon suggests giving workers a generous maximum time limit so that they do not feel rushed.

Users then attempted to answer the question using ROOKIE or IR. In each case, users saw the question and answers on a panel on their screen as they completed their work. To better constrain the task, we presented each group of users with interfaces already loaded with a useful pre-filled query, rather than relying on users to think of such queries themselves. We set ROOKIE to **user state** (Q ="Betrand Aristide", Q' ="United States" and T =1987-2007) which selects 830 documents—and we loaded IR with the query "Betrand Aristide United States" which selects 841 documents. For the IR system, we disabled the search button during the task—users were not allowed to change the query, but could use datepickers to zoom in on certain dates. For ROOKIE, users could not change Q or Q', but could vary T. Thus, this experiment measures well ROOKIE's linked temporal browsing and *Snippet Summary* help users learn the answer to a question, it does not measure other aspects of the full ROOKIE UI.

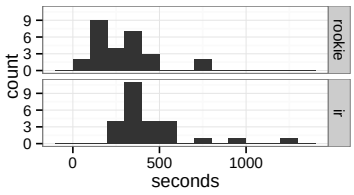
Users were instructed to research until they were reasonably confident in their answer, then submit a response. They were also required to copy and paste two sentences from *Snippet Summary* to support their multiple choice selection. By requiring evidence, we avoided junk responses and signaled to participants that answers will be scrutinized, following best practices [187] for user interface studies conducted via MTurk. In the analysis below, we analyze time to completion in cases where workers answered the question correctly, in order to ensure we are measuring "good faith" [187] attempts to complete the task.

6.3.2.3 Results and analysis

Limiting our results to the 26 workers who found the correct answer with the IR system and 27 workers who found the correct result with the ROOKIE system, we find ROOKIE users complete the task faster. We show results below, in seconds. While there is considerable variation within each group, ROOKIE users, on average, com-

pleted the task 166 seconds faster (using 37% less time); the difference is statistically significant ($p = 0.003$, t-test or $p = 0.001$, non-parametric Mann-Whitney test).

	Mean	Std. Dev.
Rookie	285.1	(169.8)
IR	451.0	(217.5)



We also observe that the fastest IR user completed the task in 210 seconds, but 12 ROOKIE users finished faster than that. Interestingly, the choice of system did not affect accuracy of the results; roughly half of both groups got the question right, and among users who submitted incorrect answers, completion times were similar (232 (149) vs. 460 (277)).

Examining system logs for ROOKIE and IR, we find that 25 successful (i.e. answered the question correctly) IR users opened individual news stories for inspection (viewing 6.4 stories on average) while only 7 successful ROOKIE users opened individual stories for inspection (viewing 2.0 stories on average). This suggests that IR users solved the task by reading documents, but ROOKIE users solved the task by reading summaries. Interestingly, the most frequently requested document by Rookie users, which describes how Aristide fled Haiti in 2004 ahead of U.S. troops, was not requested by a single IR user. The headline and Whoosh snippet for that document only describe the United States, not its relationship to Aristide. In contrast, for that document, ROOKIE's *Snippet Summary* shows a sentence that describes the U.S. and Aristide. ROOKIE users were able to find relevant information within this document.

6.4 Discussion

Natural language processing researchers have developed many techniques for extracting entities, relations, events, topics or summaries from news text.[10] Such work, in the words of Jonathan Stray [320], sometimes "discusses journalism as a potential application area without ever consulting or testing with journalists". ROOKIE is one of a handful of projects, including Vox Civitas [88] and Overview [43] which seek apply work from the NLP research community using feedback and input from actual readers, writers and editors.

During development, we consulted with three reporters, two news editors, a journalism professor and a new media applications developer. We spent nearly 18 months testing features and modifying the design before we began formal user testing. Several crucial themes emerged from this process. We strongly suspect that these lessons would apply to others seeking to join journalism and NLP.

6.4.1 Practical systems should handle NLP failures with grace

Early versions of ROOKIE attempted to produce authoritative summaries of queried documents. One early version of *Snippet Summary* attempted to combine multiple phrases and sentences from across documents without line breaks (similar to Zagat[11] reviews). Another early version showed only the top N topics in *Subjects Summary*, or the top N sentences in *Snippet Summary*, without pagination. These designs were well-intentioned: we hoped to cut through "information overload" [304] and present users with only the most important data.

However, users reported confusion and frustration with our first attempts. They did not understand ROOKIE's long and sometimes incoherent summaries—and they were annoyed when ROOKIE only showed the top 5 most important sentences or

[10] Eisenstein [100] offers a broad introduction to NLP.

[11] e.g. http://www.zagat.com/r/franklin-barbecue-austin

phrases for display. In the first case, we had trouble accounting for discourse effects in summaries, so sentences strung together into a long summary did not necessarily make sense as a whole. In the later case, we attempted to create an authoritative, fixed-width summary (either with a list of 5 terms or a list of 5 sentences), without allowing users to expand the summary as needed using pagination (as in the current design).[12]

Because NLP systems are not able to produce perfectly well-formed summaries which include the most salient information in text, we argue that it is important to design summarization interfaces which are robust to NLP errors. In ROOKIE, we did this by (1) splitting each extracted sentence into a standalone snippet (protecting against ungrammatical or semantically nonsensical multi-sentence summaries) and (2) allowing for pagination (protecting against poor computational judgments about importance). Informal feedback improved. This insight about summarization generalizes to other sorts of software that uses NLP judgements for computational journalism, such as newsroom software which uses NLP to find entities, extract events or resolve coreference. Designers and developers must build systems which can handle inevitable failures [10].

6.4.2 Text visualization should allow drill down to actual words

Early versions of ROOKIE's time series graph simply showed the frequencies of stories through time, without the rich interactions described above. However, in testing ROOKIE, we found that users assumed that they could manipulate the time series graph to drill down for more detail, and were confused when they could not do so. One editor explained: "It is useful to see how many stories appear in a given month. But that is not clickable. How are you helping the user by providing information that they can't act on"?

[12]We also heard similar feedback on early versions of the CLIOQUERY interface (§7).

We thus modified the time series graph to allow users to move smoothly from visualization to underlying text. Again, informal feedback improved. We think that this insight might also be applied to other text visualizations like metro maps [303], entity–relation graphs [317] or t-SNE plots of word distances [333]. In particular, popular time series frequency visualizations such as the Google N-Grams Viewer[13] or *New York Times* Chronicle[14], which show the frequency of a word or phrase through time, could be improved with some form of textual drilldown like a KWIC view or list of underlying documents. As Görg *et al.* [126] explain following years of development of one text analytics system, "interactive visualization of connections between entities and documents alone cannot replace the reading of reports". Such experiences with ROOKIE informed our work on the Document Viewer component in CLIOQUERY(§7).

6.4.3 NPs, not entities (or topics)

Many text previous systems have sought to help users explore and make sense of documents [242, 317, 169, 88], including software specifically designed for news archives [98, 348], software specifically designed for reporters [43], and software focused on evolving topics though time [92, 79, 80, 203].

All such systems extract some interesting aspects of text, and present them to users for visualization and navigation. Some find and display entities and relationships.[15] Others find and display learned word clusters or "topics" [92, 79, 80, 203], sometimes arranged in a hierarchy. A third approach relies on manually-created tags [98, 348].

Each of these established method has limitations. State of the art NER systems [115, 97] incorrectly tag or fail to recognize entities. Learned topics can miss categories

---

[13] `https://books.google.com/ngrams/graph?content=mobile`

[14] `http://chronicle.nytlabs.com/?keyword=mobile`

[15] e.g. `https://www.media.mit.edu/projects/news-graph/overview/` or `https://neo4j.com/blog/analyzing-panama-papers-neo4j/`

defined by domain experts, or generate topics that do not make sense [65]. Human annotation is expensive and often infeasible.

ROOKIE thus takes a very different approach: find noun phrases (NPs) and let the user quickly browse them, drawing attention to co-occurrence relationships and phrases in context. This rapid browsing replaces topical term clustering or relation identification in other systems for exploring news text. NPs have many advantages over entities and topics. Unlike topics, NPs can be expressed concisely and understood quickly, without reading and interpreting lists of (possibly nonsensical) words from a topic model. Moreover, where inference for a topic model incurs latency (a major disadvantage in a user-facing system), simple lists of important NPs can be generated very quickly in response to user queries. This is discussed further in §6.4.4. Similarly, unlike NER and relations, NPs can be extracted with very high accuracy, which is vital to user-facing systems, where nonsensical output from NER systems may confuse users unfamiliar with NLP. Additionally, where NER systems require a predefined ontology, NPs work without configuration on many corpora (§2), offering the fine-grained specificity of entity–relation systems without specialized annotations or a predefined knowledge base.

Note that unlike the NER systems that are usually more popular in computational journalism [320, p. 4], NP extraction is less tied to annotation decisions in labeled data and imposes fewer assumptions about the semantic types needed for an application. In ROOKIE, NPs included valuable concepts like "eye doctor" or "Assad family", which are important to understand queries like Q = "Bashar al-Assad", but which do not refer to the sort of concrete entities which typically serve as the basis for conventional NER systems.

The particular justification and theory for ROOKIE's *Subjects Summary* extraction method is discussed in earlier chapters (§2), following an active area of NLP research in automatically identifying "important" phrases in corpora, sometimes called

keyphrases [62], multiword expressions, or facets [319]. We consider *Subjects Summary* an application of faceted search; see Hearst [151] for more discussion of this user interface technique.

6.4.4 Speed, correctness and interpretability are not optional

In designing ROOKIE, we required that each UI component could be generated quickly, interpreted easily by ordinary users, and would never make a mistake in presenting some semantic representation of the underlying text. These requirements proved useful.

Speed. Heer and Shneiderman [153] have pointed out that "To be most effective, visual analytics tools must support the fluent and flexible use of visualizations at rates resonant with the pace of human thought." We followed this advice in building ROOKIE. In particular, NLP has developed many techniques for summarization which require several seconds of compute time [229], which is much too slow for use in a UI. During the design process, we implemented one such method [83] using a multithreaded implementation in C which employed CVBO [13], a variational method for rapid inference for topic models. Our implementation still proved too slow for interactive use. The trouble is that if summarization code runs on the server, then each time a user adjusts Q, Q' or T, ROOKIE must (a) make a call across a network to fetch a new summary (b) wait for the server to generate the summary and (c) wait for the reply. Such latency costs are unacceptable in user-facing applications. Fast, approximate summarization techniques [229] which run client-side in the browser are an exciting possibility for future research.

Correctness and interpretability. ROOKIE's time series is considerably simpler than other text visualizations proposed for news archives, many of which show evolving themes across time [92, 79, 80, 203, 147]. This was a deliberate design choice. ROOKIE's line charts showing a single lexical item certainly cannot represent clusters

of related vocabulary words or "topics", nor can they show the relationships between such topics. However, accurately summarizing topical relationships and their evolution is an AI-hard research challenge. Because ROOKIE was designed for real world use, we chose a visualization technique that could not confuse or mislead users by extracting and displaying nonsense clusters or missing important topics in the text. We show that Rookie's simple line charts improve user understanding, and we welcome such demonstrations for other more complex approaches.

6.5 Conclusion and future work

ROOKIE began as an effort to create a useful tool for news reporters. Because we set out to develop a user-facing software system, in building ROOKIE, we were forced to address and design around some of the shortcomings of modern NLP techniques. While we eventually were able to apply language engineering methods to build a tool which measurably helps end users quickly complete historical sensemaking tasks, our experiences presenting NLP methods to real users helped us define broader design goals, outlined in Section 1.5. Many of these design goals also proved important in the CLIOQUERY system, described in Chapter 7.

CHAPTER 7

CLIOQUERY

Synopsis

In this chapter, we describe the design, development and implementation of ClioQuery, a system for lexical corpus analysis built to help historians answer research questions from newspaper archives. To create ClioQuery, we first investigated the search needs of historians and archivists, and then used our understanding of such needs to inform the design of a custom interface which helps historians gather and analyze the comprehensive set of all mentions of a lexical query in a corpus.

Our work on ClioQuery demonstrates how text simplification methods from Chapter 3 and Chapter 4 can be applied in a user-facing system. During a field study evaluation and qualitative interview evaluation, we found that experts could use ClioQuery's text simplification features to answer substantive questions based on their own research. In particular, some explained how system features designed to support rapid query-focused lexical corpus analysis offered an advantage over traditional keyword document search tools. Such results suggest a role for text simplification in future lexical systems.

7.1 Introduction

Newspaper archives are fundamental resources for historians, librarians and social scientists [55, 8], because they offer a detailed primary source record of how social processes evolve across time [276]. For instance, social researchers have used news archives to examine vital questions such as why the United States abolished slavery

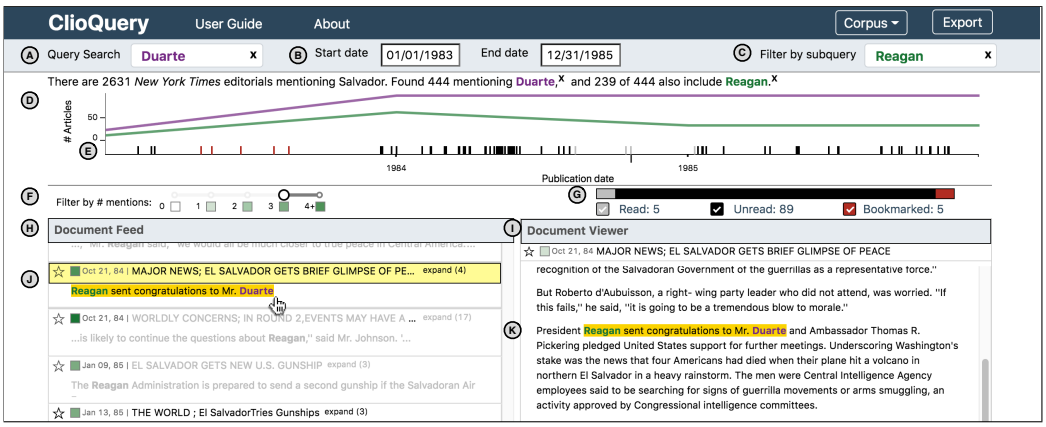


Figure 7.1: ClioQuery, an interactive text analytics system for helping historians investigate queries in news archives. Features (letters A to K) include: (D) a Time Series View showing the frequency of a user's query through time, (H) a linked Document Feed showing a skimmable query-oriented summary of every mention of the query in the corpus, and (I) a linked Document Viewer showing a selected news story, with text from the query-oriented summary highlighted in yellow. Section 7.4 describes the full system, explaining each feature.

[114] and how different jurisdictions slowed the spread of the 1918 flu [223]. While historians are known to use archives in different ways (e.g. sequential browsing [8]), prior work reports that historians often look for "specific keywords" [8, p. 2] in newspaper corpora. For instance, scholars in history and social science journals describe tracking down and reviewing occurrences of words like "William Benbow" [282], "Frances Maule" [313], "watermelon" [37], "Japanese beetles" [307], "refugee" [275], "Loving" [161], and "race suicide" [213] in news archives to help answer questions about society. In this work, we describe such search terms (e.g. "William Benbow") as **queries** and we describe each exact occurrence of a query in an archive as a query **mention**. (Section 7.9 discusses possible improvements to exact string matching.) We then describe the task of locating query mentions as **mention gathering**, and the closely-related

task of reviewing and drawing conclusions from mentions amid surrounding text as **mention analysis**.[1]

Mention gathering and mention analysis have not been carefully explored in existing work on interactive text analytics. Instead, many prior systems often focus on offering overviews of corpus contents (Section 7.2). For instance, many systems including Termite [63], TIARA [203], Overview [43], and ConVisIT [162] show users groups of thematically-related words using **word cluster displays** to suggest broad topics across documents. Others show users **textual summaries** [233, 21], or **time series graphs** of word frequencies [147, 286] to help users survey large corpora. Such overview features may help users form questions or define queries when exploring new texts. However, they are not designed to help users who already know what to search for, and wish to review specific query terms in underlying documents. This need is common among historians [8, p. 2], and has also been reported among journalists [320] and intelligence analysts [126].

Yet while prior text analytics systems are not specifically designed for mention gathering and analysis, it is certainly possible to use some but not all existing systems for this purpose. For example, although tools like Termite [63] or Overview [43] lack features to gather and analyze query mentions, in principle, a historian could use Jigsaw's list view [317] to identify documents containing some named entity, and then click, open, and scroll through such documents one-by-one in the Jigsaw document view to find and review mentions of the entity.

In practice however, social researchers like Shinozuka [307] and others [282, 313, 161, 37, 275, 213] often use traditional **keyword document search engines** such as ProQuest [279] to gather and analyze mentions in newspaper archives and other corpora. (In this work, we use *corpus* and *archive* interchangeably; we assume the corpus

[1]Using terminology from prior work [277], it is possible to interpret mention gathering as a kind of information foraging and mention analysis as a kind of sensemaking.

is an archive.) Keyword document search engines return relevance-ranked document lists in response to a free-text query. Because such systems are widely used in historical practice [8, 282, 55, 313], we propose that keyword document search systems are baseline tools for mention gathering and analysis (see Section 7.2.2.1).

Regardless of whether a historian uses a specialized interactive text analysis tool like Jigsaw, or a baseline search engine like ProQuest, performing mention gathering and analysis by opening and reviewing individual documents has two downsides. First, because almost all words are very rare (a well-known property of text [357]), any given query will very likely appear only a small number of times within a document. Therefore, finding and analyzing query mentions by reading whole documents requires examining many passages that will not directly mention the query. While search within document features (e.g. `control + F` in Chrome [70]) can certainly help, gathering and analyzing mentions still requires opening each article in its own window or tab,[2] locating mentions within the article, reading passages which mention the query, and integrating information from such passages with existing knowledge, before moving on to the next document in the corpus. This means that the user must context switch across stories as they perform a multi-step process to gather and analyze query mentions, mentally or otherwise keeping track of information from one document as they jump to the next (see Figure 7.2). Navigating between documents is thought to impose cognitive costs in keyword document search tools [105, 351], and context switching across views is thought to impose cognitive costs in visual analytics systems [340].

[2]Showing a single document in a single window or tab is a common interface pattern. It is employed, for instance, in the Overview [43] and Jigsaw [317] document viewers, and in traditional search user interfaces, which often link to individual documents from a main search engine results page [77, Section 6.3].

User study	Num. participants	Total hours
Needfinding study to guide system design (Sec. 7.3)	5	4.5
Expert interview study (Sec. 7.5)	5	5
Field study (Sec. 7.7)	2	5

Table 7.1: This work presents three separate user studies with historians and archivists. Section 7.3 describes institutional approval. Tables 7.4, 7.6 and 7.7 describe the backgrounds of participants in greater detail.

Noting the importance of historical investigation and the limitations of existing approaches, we propose the ClioQuery text analytics system[3] to help historians in their practice of mention gathering and analysis. We designed ClioQuery in collaboration with historians and archivists, using iterative prototyping [121] and user-centered design techniques [290], while drawing from prior study of the information-seeking behavior of historians.

Because our needfinding investigations revealed the importance of comprehensive review in historical research (Section 7.3.2.2), ClioQuery includes a skimmable summary showing every single mention of a user's query term across a corpus. Such summaries are designed to reduce reading burden and context switching, by condensing all query mentions into a single interface view, using techniques from natural language processing (NLP). Additionally, because we found that historians require transparency and contextual information in archive search, ClioQuery's summaries are presented alongside a linked full-text document view. Such linking is designed to quickly and transparently show summary text within the context of full-length source documents. Moreover, because temporal analysis is crucial to historians, ClioQuery also includes an interactive overview visualization, designed to help users review the frequency of query mentions through time. Together, through these and other features (Section 7.4), our system offers a query-oriented approach to text ana-

[3]Clio [345] is a common prefix (e.g. ClioVis [48]), implying a connection with history.

lytics, which allows historians to quickly and easily gather and analyze query mentions across an archive.

In total, our work offers the following:

- **A synthesis of extensive prior research in text analytics** (Section 7.2). In reviewing prior work on interactive analysis of text across time, we found that many efforts from the NLP, HCI and Visualization communities focus on offering overviews of corpus contents. Such overviews might help users formulate queries, but are not designed for mention gathering and analysis, when the user already knows what to search for.

- **An investigation into user needs and requirements** (Section 7.3). To build our tool, we translated prior research on historians' information-seeking behavior into concrete guidelines for system design. We also validated and contextualized prior work by conducting five needfinding interviews with historians and archivists, and by gathering feedback on early prototypes. This process revealed a need for transparency, trustworthiness, context and comprehensiveness in archival tools, which might inform future work on historical search [311, 310] and text summarization [82, 248].

- **The ClioQuery system** (Section 7.4). ClioQuery is an open-source text analytics system designed for historians, which tests the idea of using text summarization to reduce reading burden and context switching during mention gathering and analysis. The system also uses linked views, in-text highlighting and a time series plot to help experts quickly, comprehensively and transparently find and review query mentions across an archive.

- **An evaluation of specific ClioQuery features** (Sections 7.5 and 7.6). To test the utility and usability of ClioQuery, we conducted an expert interview study with five social researchers, who used the system to answer a historical

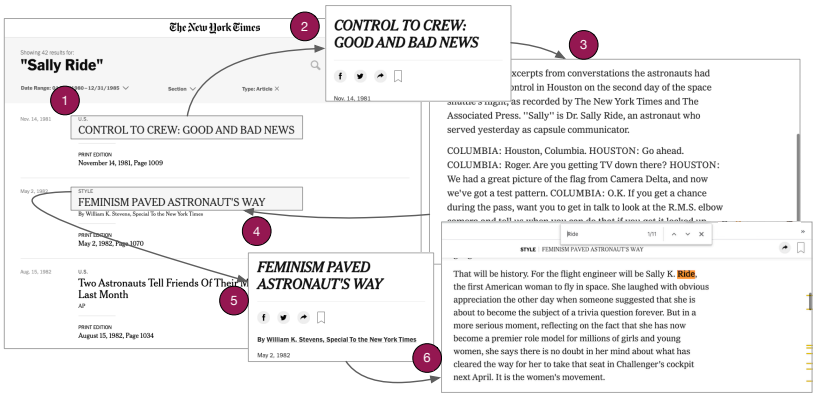
question from news archives. After methodically coding qualitative feedback, we learned that many experts found ClioQuery's skimmable summaries useful, because they condensed documents to facilitate quick review of query mentions. We also learned that linking summary text with underlying source documents using in-text highlighting was essential, because it offered necessary context for interpreting summary output.

- **An evaluation of ClioQuery in the wild** (Section 7.7). To test ClioQuery in a realistic setting, we deployed the system in a field study with two historians, who used ClioQuery to answer questions from their own research. In comparing experiences with ClioQuery to prior experiences with keyword document search systems, one historian explained how ClioQuery reduced their reading burden, and another explained how text summarization features facilitated rapid mention gathering and analysis.

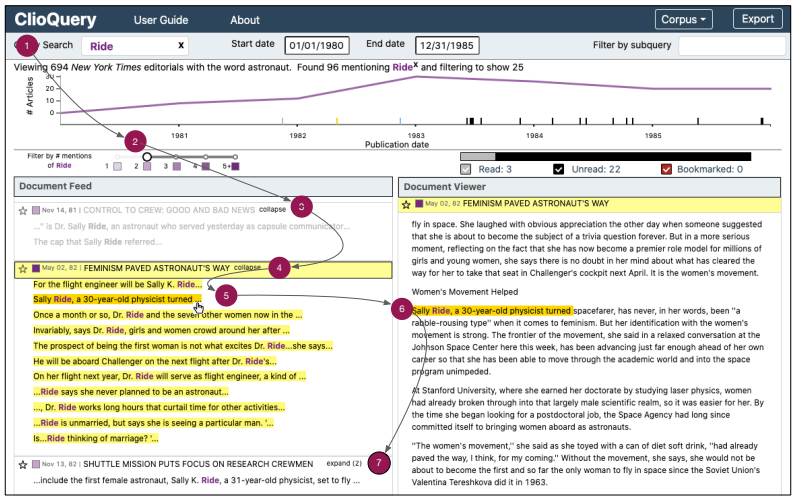
We conclude by discussing our findings (Section 7.8), reviewing limitations and future work (Section 7.9), and describing possible applications of features and ideas from ClioQuery in other query-oriented settings, beyond historical research (Section 7.10).

7.2 Related work

Historians sometimes gather and analyze mentions of specific query words in archives (Section 7.1). However, much prior work from the HCI, Visualization and NLP communities focuses on helping users gain high-level overviews of large bodies of text. We review this overview-oriented literature in Section 7.2.1. In Section 7.2.2, we also review another literature on search-based systems, which focus on retrieving text from a corpus in response to a user query. This search-based approach seems better suited to mention gathering and analysis, as search-based systems can help



(a) A user investigates Sally Ride by performing mention gathering and analysis using the *New York Times* web archive [251], a baseline keyword document search interface (Section 7.2.2.1). They first (1) click the top headline on the search engine results page (left) in order to (2) open a document in a new tab (shown on the right) and then (3) scroll down to locate mentions of "Ride" in the linked news story. The user reads and analyzes these mentions and then (4) context switches to the second document by clicking the second headline on the results page. This (5) opens a new story in a new tab. For this second document, they (6) use a search in document feature [70] (i.e. `Control+F`) to help locate mentions of "Ride" within the story.



(b) A user (1) searches for "Ride" using ClioQuery and (2) sets the filter-by-count slider to limit results to stories with at least two mentions of "Ride". The user then clicks the expand/collapse button on two news stories (3 and 4) to review all mentions of Ride from each story in the Document Feed. They then (5) click one shortened sentence mentioning "Ride" (6) to read it within the context of the full original document in the linked Document Viewer, with help from automatic in-text highlighting. The user then prepares to (7) click expand to review additional mentions of Ride in the next story.

Figure 7.2: Reviewing mentions of U.S. astronaut Sally Ride in *The New York Times*, using ClioQuery (bottom) and a keyword document search tool (top).

historians find and review query mentions in a corpus. Much evidence (Section 7.1 and Table 7.2) also suggests that search-based systems are central to contemporary historical practice.

In presenting prior work, we emphasize common user interface design patterns [323], shared among multiple prior systems. Interface design patterns are "concrete bundles of components" [324] that help a user achieve some task. We say that *overview design patterns* help the user survey the contents of archives, and that *search design patterns* help the user query for specific selections from a body of text. Table 7.2 offers a summary of major design patterns from prior work. Some individual systems (e.g. Expedition [310]) may implement both overview and search patterns.

7.2.1 Overview design patterns

7.2.1.1 Word clustering

Because users often can not review every document in a large corpus, many prior text analytics tools such as Termite [63], TIARA [203], Overview [43], RoseRiver [80], TextFlow [79] and ConVisIT [162] try to suggest overall themes in a body of text, by showing lists of thematically-related words from underlying documents. We describe this approach as the word clustering design pattern.

Some systems which implement the word clustering pattern are based on prior work from NLP, information retrieval, and text mining, focused on identifying and representing patterns of co-occurring words using methods such as topic models [38] and word embeddings [236].[4] Researchers in HCI and Visualization extend this work by considering how to present such patterns in a graphical interface; some systems show changes in cluster patterns across time [203, 80, 79] (e.g. Figure 7.3a), others

[4]The system Themail [335] clusters words by time, instead of by co-occurrence statistics. Because this system shows lists of related words (related by time period), we say the system implements word clustering. Similarly, VisGets shows clusters (of document tags) defined by a user's selection in the interface [98], which we consider to be a form of clustering.

do not show time-based topics [63, 43]. Because automatic clusters may not match a user's mental model of a corpus, some work investigates human-in-the-loop techniques, which allow users to modify and guide word and document clusters through a GUI [165, 260, 306, 162].

Word clustering has a clear role in historical research. In query-oriented settings, clustering methods may help the user formulate queries they had not considered [330]. Moreover, specialized and computationally-oriented digital humanists [240] and historians [128] have used word clusters from topic models for historical literary analysis. Nevertheless, successful application of topic modeling requires specialized knowledge and extensive interpretive effort [28, 294], making this method less accessible to a broader audience of historians. Additionally, many historians approach archives looking for mentions of what Allen and Sieczkiewicz describe as "specific keywords" [8] rather than looking to explore word cluster overviews from a topic model API. Because we design for historians investigating known query terms (Section 7.1), we do not employ the word clustering pattern in the ClioQuery interface.

7.2.1.2 Textual summary

Rather than showing lists of related words to offer a corpus overview, a large body of work on text summarization from NLP [82] instead attempts to create short paragraphs which convey the most "important" information in a corpus, by selecting a collection of sentences or sentence fragments from input documents to form an output summary. (This is sometimes described as extractive summarization [82], because the output text is extracted from input text.) User-facing systems such as Newsblaster [233] and NSTM [21] apply this research by showing such textual summaries in a graphical interface. We say that such tools implement the textual summary design pattern (Figure 7.3b).

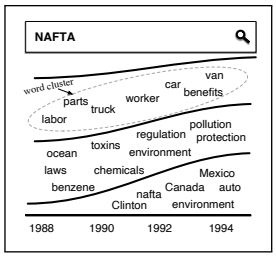
Like word clusters, traditional text summaries do not seem to help with mention gathering and analysis. A user can't turn to a traditional text summary to find and review query mentions, because "important" sentences selected for inclusion in summary output may or may not contain a given query word. Moreover, traditional approaches typically do not explain *how* "important" information is chosen, which may be important in the history domain (Figure 7.5 and Section 7.8.2).

However, two ideas from the traditional text summarization literature may help historians perform mention gathering and analysis. First, work in query-focused summarization tries to identify the most salient information in a corpus, based on a user's query [248]. Historians might use such query-focused summaries to review keywords in text. Query-focused summaries which define *all* query mentions as important enough to warrant inclusion in summary output may be especially helpful (see Section 7.3.2.2). Second, work on sentence compression [189, 108, 109] tries to shorten individual sentences by removing words, usually for the purpose of including more (shortened) sentences in a fixed-length summary. These methods, or closely-related sentence fusion techniques [27], might be used to shorten passages containing query terms, to help users quickly review many mentions of a query in context. We apply these two ideas from text summarization in ClioQuery (see Section 7.4.3 and 7.4.7).

7.2.1.3 Time series plot

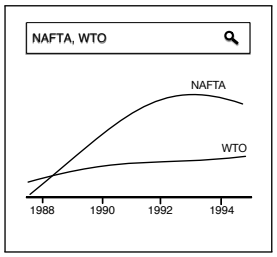
Instead of showing text to summarize corpus contents, time series plots present the frequency of documents or query-mentions across time to offer a visual (rather than textual) corpus overview. This pattern is often implemented in text analysis tools [286, 220, 88, 89] and keyword search systems [310, 225, 253]. We divide time series visualizations into two groups. One group [235, 311, 225] shows the frequency of a single query term across time (e.g. Figure 7.3c), often using a line chart. Another group shows the frequency of multiple terms (e.g. highest-count words) using a stacked

Overview patterns
(Sec. 7.2.1)



(a) Word clustering (Sec. 7.2.1.1)
Examples: [80, 43, 63, 260, 306, 162]

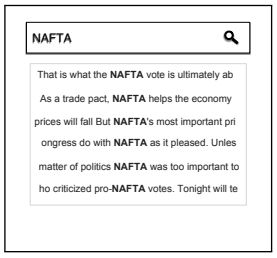
(b) Textual summary (Sec. 7.2.1.2)
Examples: [233, 232, 21]



(c) Time series plot (Sec. 7.2.1.3)
Examples: [147, 235, 286, 30]

Search patterns
(Sec. 7.2.2)

(d) Keyword search (Sec. 7.2.2.1)
Examples: [251, 150, 160, 253, 73, 286]



(e) Multi-doc. snippet (Sec. 7.2.2.2)
Examples: [208, 262, 286, 335]

Figure 7.3: We define five major user interface design patterns from prior work devoted to helping users understand news archives and other corpora (Section 7.2). Three of the design patterns focus on helping users gain an overview of archive contents (top two rows). Two of the design patterns focus on helping the user to search for specific documents or passages from a corpus (bottom row). Following Tidwell [323], this figure presents prototypical wireframes of each design pattern, created by the authors of this work. Each example above shows a system presenting results from 129 documents matching the query "NAFTA" on a corpus of *New York Times* editorials published between 1988 and 1994.

area chart [147, 30], and may not require a user-supplied query. While time series plots alone can not be used for mention gathering and analysis because they do not show underlying text from a corpus, such visualizations can hint at important events or changes across documents (e.g. Michel *et al.* [235]). We implement this design pattern in ClioQuery (Section 7.4.2).

7.2.2 Search design patterns

7.2.2.1 Keyword document search (baseline)

Traditional keyword document search tools return relevance-ranked lists of documents on a search engine results page (SERP), in response to a free-text query [216]. Because historians often use such tools in practice (Section 7.1), we consider these systems to be baselines for mention gathering and analysis.[5]

Although keyword document search tools are widely used (Table 7.2), these systems have clear downsides for finding and reviewing query mentions. First, keyword document search systems impose unnecessary burdens from reading and context switching. This is described in detail in Section 7.1. Additionally, keyword document search systems rank documents according to a computational model of relevance, which may be undesirable for historians as relevance-ranking introduces opaque algorithmic influence over qualitative conclusions (by guiding users towards particular documents). Section 7.3 describes the importance of neutral and comprehensive review in historical research.

Ranking aside, keyword document search tools may also shape user perceptions of the contents of the individual documents in an archive, through displaying single-document summaries (also called query-biased snippets [325]) on the search engine

[5]One strand of humanities scholarship critically investigates how widespread adoption of keyword document search tools might be distorting traditional humanistic research [282, 313, 330].

	Rel. ranking	Filter by Date	Known users
Chronicling America	✓	✓	$I3, I4, I5$,$P4, H1$
Newspapers.com	✓	✓	$I1$,$P1, P4, H2$
New York Times Search	✓	✓	$I3, I4, I5$,$P1$,$P5$,$H1, H2$
ProQuest	✓	✓	$I1$ to $I5$, $P1, P3, P4, P5, H1, H2$

Table 7.2: Example baseline keyword document search systems, featuring relevance-ranked search engine results pages and filtering by date; such features are common in many news archive interfaces [99]. Tables 7.4, 7.6, and 7.7 further describe known users. Above, we use I1 to I5 to indicate all interviewees.

results page.[6] For example, Figure 7.3d displays three sample single-document summaries, showing what a computer deems to be the most important information from three different search results. Such single-document summaries may be inappropriate for historical research, as some historians may be skeptical of opaque models which select "important" information for their review (search engines try to include keywords in snippets, but do not try to explain summaries [77, Section 6.3.1]). Figure 7.5 describes our own experiences attempting to apply similar document summarization techniques for historians without success.

7.2.2.2 Multi-document snippet

Where keyword document search systems return links to single documents in response to a user query, other systems return collections of smaller units like paragraphs, sentences or character spans, which are often drawn from multiple documents (see Figure 7.3e). We observe two different implementations of this multi-document snippet design pattern in interactive text analytics.

First, multi-document snippet features can be used in word clustering systems to help the user investigate mentions of particular clustered words in context. For

[6]Google sometimes shows complex results snippets on the SERP, using proprietary techniques. Brin and Page briefly mention the need for such "Result Summarization" in their original paper [46, Section 6.1].

example, TIARA [203] allows analysts to review individual words from a cluster in underlying text. However, because TIARA is designed for showing broad themes rather than for reviewing query mentions, it does not comprehensively show all mentions of a given word in its multi-document snippet. Instead, TIARA chooses some selection of mentions for display, optimizing for diversity [203, Section 6]. Such curation may introduce unwanted algorithmic bias (Section 7.3.2.4), because the system chooses some but not all query mentions for display.

Additionally, other text analysis systems which are not necessarily focused on clustering sometimes include keyword-in-context (KWIC) views [286, 262], showing each mention of a query word (or a selection of such mentions) on its own line of text, amid immediately surrounding tokens or characters (e.g. Figure 7.3e). While this form of multi-document snippet can be used for mention gathering and analysis, KWIC views have some limitations for historical research. First, in many cases, historians need to investigate particular query mentions within the context of full documents (Section 7.3.2.3). While KWIC views may include links to underlying sources, jumping from KWIC views to documents requires context switching into new windows or tabs to gather and analyze evidence. We explain why this is undesirable in Section 7.1. Second, KWIC views always show some number of pixels, characters or words immediately surrounding each query mention. This may result in awkward-sounding or choppy snippets that do not include the most salient information in source sentences; evidence suggests that users dislike awkward-sounding snippets [66]. Finally, KWIC views do not offer a way to keep track of which mentions have been reviewed during analysis, which may be important in historical research (Sections 7.3.2.2 and 7.4.5). Noting these shortcomings, it is possible to interpret certain ClioQuery features (Section 7.4.3 and 7.4.4) as a particular form of KWIC view, addressing some of these limitations.

7.3 Current practices, user needs and design requirements

In Section 7.1, we document a common practice of mention gathering and analysis, in which historians find and then review mentions of specific keywords in newspaper archives. We then offer evidence which suggests that historians often perform this work using keyword document search systems, which require the user to undertake unnecessary reading and context switching. Noting these limitations, we study the needs of historians (Section 7.3.1) in order to define requirements for a text analytics system (Section 7.3.2), designed for mention gathering and analysis.

7.3.1 Observing and analyzing user needs

We identified user needs by collecting and analyzing two different sources of data, described below.

7.3.1.1 Observing needs from existing literature

First, we studied user needs by reviewing a large literature from history, library science and information science devoted to the systematic study of the digital and non-digital information-seeking behavior of historians (Table 7.3). This work is largely unknown in computer science disciplines like NLP, IR, VIS and HCI. In our study, we seek to translate its descriptive findings—focused on how historians find information—into actionable design requirements for user-facing software. To identify this literature, we followed citations starting from Allen and Sieczkiewicz's paper "How Historians use Historical Newspapers" [8], which we first found via a search on Google Scholar. In total, we reviewed and took notes on six prior studies describing surveys and interviews with 1002 historians (Table 7.3).

7.3.1.2 Observing needs from interviews and feedback on prototypes

We additionally supplemented, contextualized and validated existing studies by conducting five of our own one-on-one needfinding interviews (Table 7.4) on Zoom

Author(s)	Venue	Study type	Participants
Allen and Sieczkiewicz [8]	*Proc. ASIS&T* (Info. Science)	Interview	8
Case [51]	*The Library Quarterly*	Interview	20
Duff and Johnson [95]	*The Library Quarterly*	Interview	10
Chassanoff [55]	*The American Archivist*	Survey	86
Dalton and Charnigo [81]	*College & Research Libraries*	Survey	278
Duff, Craig, and Cherry [94]	*The Public Historian*	Survey	600

Table 7.3: A selection from prior work in library science and information science, focused on the information-seeking behavior of historians. These papers describe studies of $N = 1002$ historians (in total).

video chat over a period of three months.[7] All but one interview was 60 minutes long. (We met with I4 for 30 minutes, due to limited availability.) User interviews proceeded in two phases. During *Phase A*, in the initial exploratory stage of our work, one researcher from our group interviewed I2, I4 and I5, who we recruited through convenience sampling [124]. The interviewer asked open-ended, exploratory questions about needs and practices, and solicited feedback on early prototypes. The researcher also took detailed notes. Later, when we better understood how historians find information in archives, we began *Phase B*. During this phase, the same researcher conducted two one-on-one, video-recorded, semi-structured interviews with I3 and I1, who also provided feedback on later prototypes. We recruited I3 and I1 via email outreach.[8] The researcher again took detailed notes. We include the interview script in supplemental material. In total, each of the five interviewees across *Phase A* and *Phase B* reviewed a different iterative prototype. In the interest of space, we only

[7] All three of our user studies—needfinding interviews, expert interviews, and field study (Sections 7.3.1, 7.5, and 7.7 respectively)—were approved as exempt from review by our institution's human subjects IRB office. All participants received a $50 Amazon gift card for their time.

[8] We emailed five PhD students in history at a nearby university. Each student expressed interest in media, archives or science in describing their work on their department's web page. We also emailed all members of the editorial board at a history journal. We do not list the name of the university or journal to ensure interviewees remain anonymous.

present feedback on what we consider to be the two most important prototypes, shown in Figure 7.4 and Figure 7.5.

7.3.1.3 Analyzing observations of user needs

Following data collection, one researcher qualitatively analyzed and organized notes and transcripts to articulate four overall user needs, and translate these needs into four corresponding design requirements (described in Section 7.3.2). In general, we found that feedback from needfinding interviews and feedback on early prototypes was very consistent with findings from prior work. Nevertheless, our own needfinding interviews helped to contextualize and translate prior descriptive findings on historians' information-seeking behaviors into actionable guidelines for system design.

ID	Research experience	Library experience	University role	Gender	Field
I1	5-10	1-5	PhD Candidate	Male	History
I2	0	20-30	Librarian	Female	Lib. Science
I3	10-20	0	Junior Faculty	Female	Am. Studies
I4	10-20	10-20	Archivist	Male	Lib. Science
I5	10-20	10-20	Librarian	Non-binary	History

Table 7.4: Interviewees in our needfinding study. We list research experience and library experience as a range of years. We abbreviate American Studies as Am. Studies and Library Science as Lib. Science.

7.3.2 Needfinding results and design requirements

Following data collection and data analysis, we defined four design requirements (R1-R4), based on four user needs. We describe each requirement below.

7.3.2.1 R1: A system should show an overview of change over time

Prior study of the information-seeking behavior of historians emphasizes the theoretical importance of *"the dimension of time"* [51] in historical research, and also emphasizes historians' practical need to perform *"searching and narrowing by date"* [8]. In our needfinding interviews, historians and archivists also stressed the theoretical

and practical importance of time-based investigation. *"Time is always a historian's first move*," I3 explained. *"It's about change over time as the fundamental thing."* I5 noted: *"Historians are often trying to find articles within a specific date range and about a specific topic ... research often starts with a keyword and a date range and a source or list of sources*." Because historical research involves studying change across time, I2 explained how time series plots showing the frequency of query words by time period (see Figure 7.3c) are often useful for gaining a temporal overview of a corpus. *"Bar charts [or line charts] by time are really helpful*," I2 explained, *"because news has these peaks where a topic becomes important and then dies down."* Such charts *"help people trace an idea or series of ideas or terminology over time*." Observing the centrality of temporal analysis in historical research, we assert a design requirement (R1): a historical search system should show a navigable overview of change over time. Showing a temporal "overview first" [153, 308] adapts known best practices for interactive data analysis to the history domain.

7.3.2.2 R2: A system should help users comprehensively review mentions

Prior work often emphasizes the importance of gathering comprehensive evidence during historical research. *"Comprehensiveness is clearly the highest priority in searching a database,"* one study concludes [81], explaining that 70% of 278 survey respondents would prefer to spend time filtering out irrelevant material than run the risk that relevant material *"might fall through the cracks"* in a limited search. Nevertheless, some historians in prior work acknowledge that truly comprehensive search is an impossible goal. *"I never think I'm going to be able to read every record in the archives,"* one reports [95]. *"I'm always creating priority orders of what I think is going to be most useful."*

Our interviewees similarly emphasized the importance of comprehensiveness in gathering and evaluating historical evidence. *"The most important thing for histor-*

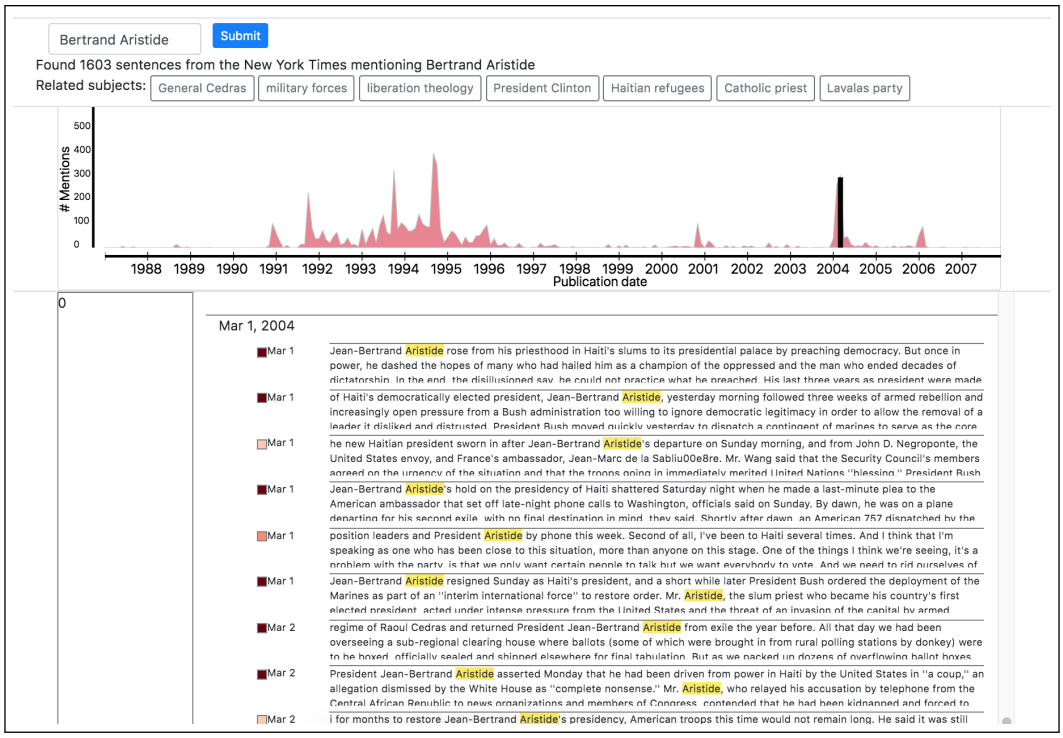


Figure 7.4: An early prototype of CLIOQUERY, displaying and highlighting every single mention of the query term "Aristide" in *New York Times* articles mentioning "Haiti". "Are we showing too much information in this interface?" one researcher from our group asked I4, when presenting the prototype. "This is literally every mention of your query term." *"No this is good,"* I4 explained, *"because of what I was calling the type II error concern [i.e. the fear of missing relevant material]. When I see something that is trying to decide or curate for me that is a worry. That is a red flag."* However, I4 went on to explain how the interface needed to provide more context and transparency surrounding highlighted snippets. *"With this design you have to click or read each snippet to see if it is relevant,"* he said. "*The snippets are valuable and good but very small and you have to look at the contents of the article. Sometimes you can eliminate that by just quickly scanning the article title ... there needs to be a way to provide the information in a more transparent way.*" (The search bar shown at the top is non-functioning mockup; the "0" on the left hand side is a placeholder.)

ical researchers is to be confident that they are being exhaustive" said I4. *"I want to know I can be confident I have been able to access everything relevant. Did my search cast a wide enough net?"* I4 also praised an early prototype (Figure 7.4) for displaying a very large number of potentially relevant passages. *"The biggest fear is Type II error,"* he explained. *"In doing searches, am I missing something that is crucial but I*

don't know because I never looked"? Similarly, I5 explained that it is important to *"be as completist as possible"* in historical research. *"The thing about historians....they want to be as comprehensive as possible with their topics*." Citing the importance of comprehensiveness, I5 expressed deep skepticism about an early prototype which omitted some information to form a summary (Figure 7.5). However, like some interviewees in prior work, I2 pointed out that truly comprehensive investigation may not be possible. *"Ultimately,"* she noted, *"there is a limit in terms of time and money for any given project"*. Because historians often search archives for mentions of a query (Section 7.1), we translate the need for comprehensive archival search into a second design requirement (R2): a system should help users comprehensively review all query mentions in a corpus.

7.3.2.3 R3: A system should present as much context as possible

Prior work sometimes emphasizes the importance of context in historical study, and newspaper archive research. *"Building context is the* sine qua non [indispensable condition] *of historical research,"* Duff and Johnson write [95]. *"Without it historians are unable to understand or interpret the events or activities they are examining."* These authors explain that that historians (1) need to understand the broader historical context for any specific event, (2) need to understand the broader context surrounding creation of any given archive record (i.e. in the words of one historian, *"who is writing it and why"* [95]) and (3) need to understand the context surrounding the creation of any given archive record. *"You can't have the specific facts without the context,"* one historian similarly explains, in a separate study [8]. *"Where an article is in the paper, and what surrounds it, matters."* Similarly, historian Adrian Bingham [36] cautions that *"newspapers were materials that we were bought, read and passed around."* Because keyword search returns news stories *"plucked out of their context"*,

he advises that these tools *"should be used in conjunction with, rather than replacing, the careful study of whole newspapers"*.

During our needfinding interviews, historians and archivists also repeatedly emphasized the importance of contextual information in archive news search. The job of a historian is to *"put facts in context"* I5 said. A historian will need to *"contextualize"* facts from a periodical by examining its publishers and audience. Similarly, I4 noted that *"as an archivist I do research to give context to collections."* Finally, I2 stressed the importance of contextualizing evidence in archive search software. *"Who does the New York Times have writing this?"* I2 asked, while examining an early CLIO-QUERY prototype (very similar to Figure 7.5). *"Where does each sentence occur in the document? What section of the newspaper? You need to show more context"*. Observing the importance of context in historical research, we assert a design requirement (R3): a system should present as much context as possible for any given record in an archive. In this work, we adopt historians' use of the term context to refer to both the text surrounding any given sentence or paragraph in an article, and to refer to the structured (e.g. publication date, headline) and unstructured (e.g. related articles during time period, other articles on newspaper page) contextual metadata for any given news story.

7.3.2.4 R4: A system should be as neutral as possible

Prior studies of the information-seeking behavior of historians underscore the need for trustworthy tools that transparently present digital archival materials in a neutral manner. For instance, in one study [94], historians report that each digital source must be *"accurate, undistorted and complete*," explaining that each source is *"not edited in some way."* Similarly, in another study [55], a historian explains that direct *"access to the original image of the primary source rather than to a transcribed version"* is important *"especially when there is no description of what rules they used to transcribe*

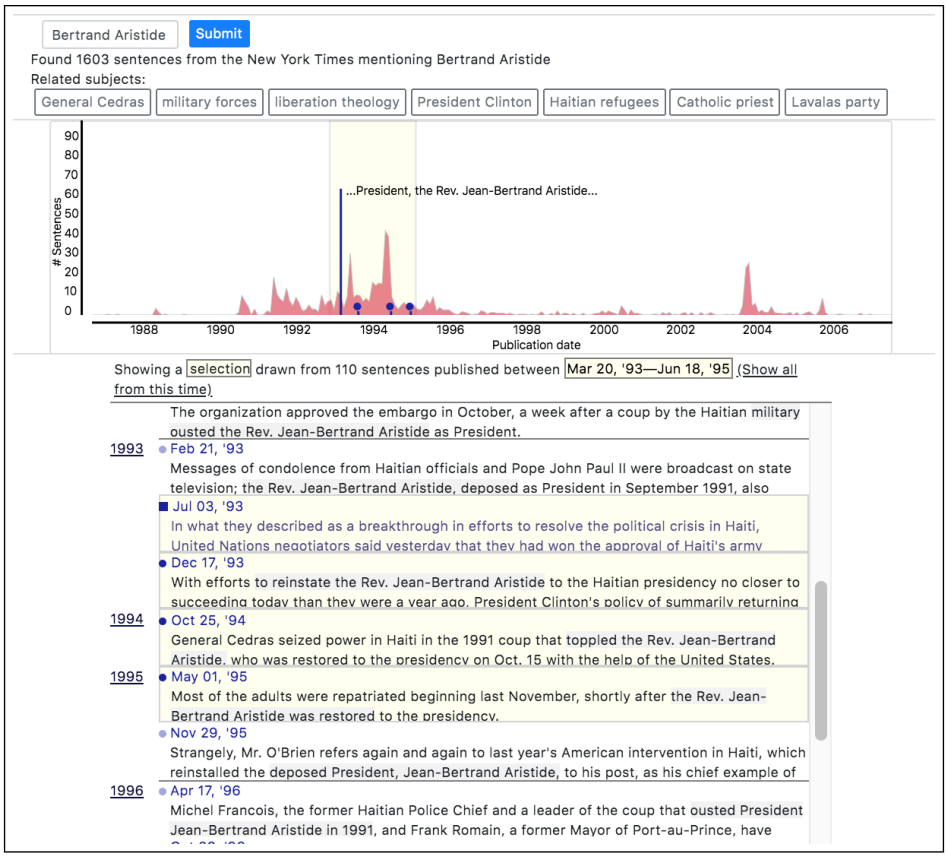


Figure 7.5: An early prototype of ClioQuery, which used traditional, optimization-based text summarization methods from natural language processing [229] to try and select the most salient information from a given time period for display (see Section 7.2.1.2). This prototype selects four most "important" articles (shaded in light yellow in the feed below), to summarize the hundreds of articles mentioning the query "Bertrand Aristide" in *The New York Times* from May 20, '93 to June 18, '95 (shown shaded in light yellow in the time series above). I5 strongly disliked this approach, prompting a shift towards interfaces emphasizing transparency and trustworthiness. *"I need to know what is included and why,"* I5 explained. *"I need to know why it is showing this limited view."* I5 continued, *"I am wary of algorithms that choose for me what the important facts are. I am a PhD historian. Leaving stuff out. We are taught to be critical of that."* Ultimately, I5 noted, *"History is written by the victors. What actually matters is what people choose to put in the timeline."* We theorize that I5 could not trust the prototype because it seemed to lack the capacity to select important facts or the integrity to adhere to historical research principles; prior work in HCI (e.g. SMILY [50]) assumes that in order to earn user trust, a system must both have the capacity to help the user and the integrity to adhere to principles which are important in a given domain.

documents". This historian reports that they do not trust and can not interpret electronic transcription, and thus must rely on direct observation of digitized images to draw conclusions.

In our interviews, historians and archivists similarly described the importance of transparently presenting digitized archives in a neutral manner. *"When I see something that is trying to decide or curate for me that is a worry. That is a red flag*," I4 explained (see Figure 7.4). Similarly, I2 added, *"I think the system should be as transparent as possible. I need to distinguish between what some primary source is saying versus what the computer thinks a primary source is saying."* I5 also cited the importance of transparency and trust in expressing deep skepticism about an early prototype (see Figure 7.5).

Yet, even as some interviewees stressed the importance of unbiased, transparent and trustworthy presentation of archive evidence, I3 reported that, in practice, historical researchers do trust ranked results from keyword document search systems. She explained that many historians might not realize that black-box document rankings from a keyword document search tool will affect conclusions from archival research. Nevertheless, because many other historians frequently expressed commitments to direct and neutral observation of archive evidence, in designing new archive tools, we assert a design requirement (R4): search software should show archive evidence in a maximally transparent and trustworthy manner.

7.4 System

We designed and implemented ClioQuery via iterative prototyping [127], while studying the needs and practices of historical researchers (R1-R4, see Section 7.3). We hypothesize that ClioQuery's novel combination of query-focused text summarization, linked views, in-text highlighting, and history tracking can help experts

quickly, comprehensively and transparently gather and analyze the comprehensive set of all query mentions in a news archive, to help answer historical questions.

7.4.1 High-level system description

The ClioQuery web interface presents results from a Boolean search [216, Chapter 1], which returns the unranked set of documents mentioning a query term in a corpus. (We say a document or passage mentions Q if it contains an exact string match for the query term Q. The exact string match is called a query mention. Section 7.9 discusses possible extensions to exact string matching.) When a user enters a single-word query Q into the search bar at the top of the interface (Figure 7.1A), ClioQuery identifies all documents mentioning Q and presents the documents using three linked views [49]. First, ClioQuery includes a **Time Series View**, showing a graphical overview of the count of documents mentioning the query by year (Figure 7.1D). Second, ClioQuery includes a **Document Feed** view, presenting all query mentions from across all documents in a single scrollable window (Figure 7.1H). Finally, ClioQuery includes a **Document Viewer**, which shows the full text of a single document from the corpus, with individual query mentions from the document highlighted in context (Figure 7.1I). ClioQuery also includes a **filtering system** to help users narrow the set of query mentions shown in the interface (Figure 7.1B, C and F), and a **history tracking system** to automatically monitor and display reading history during comprehensive search (Figure 7.1G). All features in the interface also follow a coordinated **color coding** scheme. For instance, the user's query word is always displayed using the purple query color ■ in the Document Feed and Document Viewer, and the Time Series View also uses a purple line to represent query frequency (Figure 7.1D). We consider the color-coded bolding of query terms to be one form of **automatic in-text highlighting** [141] throughout the ClioQuery interface. Automatic in-text highlighting draws user attention to some word, phrase

or passage in text by automatically setting the text's foreground color, background color or decoration (e.g. bolding). The Appendix describes our process for selecting a colorblind safe and print-friendly palette. It also provides additional engineering details about our implementation of ClioQuery.

7.4.2 Overview first: a Time Series View (R1)

Because change across time is central to historical research (R1), ClioQuery presents a navigable Time Series View (Figure 7.1D) showing query frequency by year across a corpus. Displaying such an "overview first" [308, 153] is a known best practice in interactive data analysis. In ClioQuery's Time Series View, the x-axis represents time and the y-axis represents the count of documents mentioning a user's query during a given year (Section 7.2.1.3 describes similar plots in other systems). If a user also enters a subquery (Section 7.4.6), ClioQuery's Time Series View also shows the annual count of documents mentioning both the query and subquery. In Figure 7.1D, ClioQuery displays one line showing the count of documents mentioning the query term in the purple query color, and another line showing the count of documents mentioning the subquery term (as well as the query term) in the green subquery color ■. ClioQuery's time series plot also shows a single rug point (small vertical line) for each document mentioning the query, just beneath the temporal x-axis (Figure 7.1E). Such rug points allow the user to easily preview and navigate to individual news stories (we describe these interactions in detail in the Appendix).

7.4.3 A Document Feed for comprehensive search (R2)

During needfinding, we found that experts often emphasized the importance of gathering comprehensive evidence (Section 7.3.2.2), and also often search for specific query terms in news archives (Section 7.1). We thus designed ClioQuery's Document Feed to help such users easily gather and analyze the comprehensive set of every single mention of a query term in a collection of news stories (R2). We assume the

user is working with a small corpus (or small set of documents from a larger corpus), where such comprehensive review is possible. (This assumption is appropriate for our use case; for instance, Black reviews roughly 500 documents to analyze the racial history of "watermelon" [37] and MacNamara reviews 605 documents to analyze "race suicide" [213].)

After a user issues a query Q, ClioQuery populates the Document Feed to show a comprehensive, skimmable, summary of the query across the corpus (Figure 7.1J). To create the summary, ClioQuery shortens each document mentioning Q and then displays shortened documents in chronological order. To shorten each document, ClioQuery first selects all sentences mentioning Q within the document, and then shortens each sentence by removing words (see Section 7.4.7.1), without removing the query. ClioQuery also automatically highlights the query word in the purple query color within each shortened sentence. By shortening, highlighting and presenting all sentences mentioning Q in this manner, ClioQuery creates a summary of all mentions of a query across a corpus, in a single view, in a visually consistent format designed for skimming (Figure 7.6).

Note that by default, ClioQuery displays a single shortened sentence from each document beneath the document's headline (Section 7.4.7.4 describes how the sentence is chosen). To see all sentences mentioning Q within a document, the user can click an "expand" button. The user can also click a star to bookmark a document in the red bookmark color ■.

ClioQuery's Document Feed is designed to directly address two of the limitations of baseline keyword document search systems, described in Section 7.1. First, by summarizing documents mentioning Q, ClioQuery is able to fit more query mentions in limited screen space, reducing the need for context switching across individual windows or tabs. For instance, in Figure 7.1, the Document Feed saves the user from having to open 239 separate documents during comprehensive review. Second,

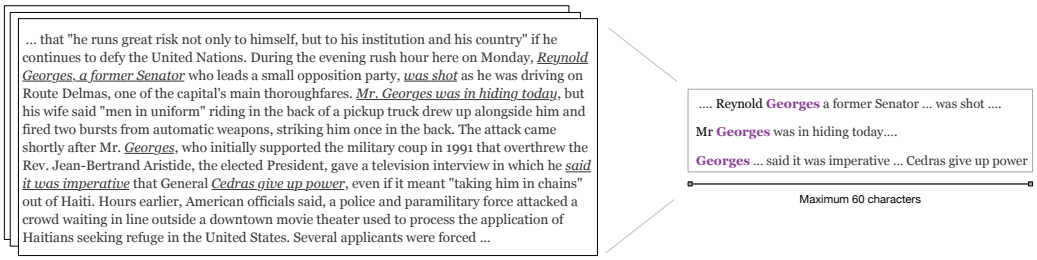


Figure 7.6: ClioQuery's Document Feed uses text simplification methods from natural language processing in conjunction with color-coded, automatic, in-text highlighting, in order to summarize query mentions in a visually consistent format designed for skimming. Here one portion of one document (top of stack, left) containing three mentions of the query term Q="Georges" has been shortened into a summary of "Georges" (right). To create this summary, ClioQuery extracts each of the three sentences mentioning "Georges" and simplifies them to render shorter sentences that are fewer than 60 characters long. In this figure, for illustration, spans of tokens from one source document included in the summary are shown with italicized and underlined text. In typical use of ClioQuery there are often hundreds or thousands of documents containing Q (shown as document stack, above left).

Text to read	Number of tokens
All documents with Reagan and Duarte	222,544 tokens
All mentions of (i.e. sentences with) Reagan or Duarte	49,382 tokens
All shortened mentions of Reagan or Duarte	28,859 tokens

Table 7.5: This table shows (from top to bottom) the count of tokens in 239 documents containing containing the words Reagan and Duarte (Figure 7.1), the count of tokens in all sentences mentioning Reagan or Duarte within such documents, and the count of tokens after ClioQuery applies sentence simplification. Sentence simplification removes 87.0% of tokens from all documents, and removes 41.6% of tokens from all sentences. Counts are from the Salvador corpus (see Appendix).

by selecting sentences mentioning the query from documents, and removing tokens from those sentences, ClioQuery reduces the user's reading burden. For instance, in Figure 7.1, the user has queried for documents mentioning Reagan and Duarte (in this example, Reagan is a subquery; subqueries are described in detail in Section 7.4.6). By selecting and simplifying sentences, ClioQuery removes 87.0% of the tokens in all documents mentioning these two words (Table 7.5). We include a detailed description of ClioQuery's text simplification techniques in Section 7.4.7.

7.4.4 A linked Document Viewer for necessary context (R3, R4)

Because historians need to evaluate evidence in context without black-box algorithmic influence (R3, R4), we anticipated that ClioQuery users would need to quickly review shortened sentences from the Document Feed within the context of underlying news articles. Therefore ClioQuery's Document Feed is closely linked with a corresponding **Document Viewer**, which shows the complete text of a single selected document from the corpus (Figure 7.1I). After a user clicks a shortened sentence in the Document Feed, the Document Viewer updates to show the entire document mentioning the sentence. ClioQuery also automatically scrolls the document so that the (just clicked) simplified sentence is visible on screen.

ClioQuery also makes it easy for users to locate simplified sentences, by using automatic in-text highlighting to further link the Document Feed and Document Viewer. All simplified sentences in the Document Feed are shown with yellow background highlighting □ in the Document Viewer. Additionally, if a user hovers over a sentence in the Document Feed or Document Viewer, the mention is highlighted in dark yellow hover color □ in each component (shown in Figure 7.1J and 7.1K). We hypothesize that linking between shortened text and full documents helps build user trust for summaries, because it helps experts transparently see and understand how shortened mentions are drawn from underlying text. This feature is inspired by CommunityClick [171].

7.4.5 Color-coded history tracking for systematic review (R2)

Some historical researchers emphasize the importance of comprehensively examining all available evidence during research (R2). To support historians in this work, ClioQuery keeps track of which documents the analyst clicks in the Document Feed and opens in the Document Viewer. ClioQuery also keeps track of bookmarked news stories (Figure 7.1J), and displays a simple stacked horizontal bar chart

(Figure 7.1G) showing the proportions and counts of read, unread and bookmarked documents. The bar chart uses the read ■, unread ■, and bookmarked ■ color scheme employed across the color-coordinated interface. (ClioQuery considers all documents to be either read but not bookmarked, unread or bookmarked. We do not allow intersection between these sets.) For instance, Figure 7.1G shows 5 read, 89 unread and 5 bookmarked documents. The user can click check marks (Figure 7.1G) to show or hide documents in each category.

ClioQuery's Document Feed and Time Series View use the same color scheme to help users quickly identify opened and unopened documents. Stories that a user has already clicked appear with grey read text in the Document Feed, and their corresponding rug points are shown in grey in the Time Series View. For instance, in Figure 7.1, the user has read the story published on Jan. 9, 1985. The story is greyed out in the Document Feed, and its corresponding rug point is shown in grey beneath the time series plot. Similarly, there are five red rug points in Figure 7.1E, because the user has bookmarked five documents.

Note that ClioQuery's history tracking is query-dependent; tracking resets each time a user issues a new query, unlike the history tracking mechanism in some prior work [169, Section 6]. Such query-dependent tracking is appropriate for ClioQuery, because the system is designed to help historians review all mentions of some specific keyword in a corpus. We hypothesize that this feature offers experts assurance they have comprehensively reviewed all mentions of a given query, and leave exploration of other forms of history tracking for future work.

7.4.6 Filter instead of rank, to avoid confounds (R4)

Some prior text analysis systems designed for historians (e.g. Expedition [310]) attempt to answer keyword queries by ranking documents to direct users towards most-relevant news articles. Because such ranked retrieval might introduce unwanted

algorithmic influence over the expert search process (R4), ClioQuery instead responds to queries with Boolean search, which returns the unranked set of all documents mentioning a query term. (The Document Feed shows such documents in chronological order.) ClioQuery then allows users to narrow down unranked search results with a filtering system, consisting of three filter controls.

The **filter-by-date** control selects documents by time period. After users select a start date and end date from date pickers at the top of the interface (Figure 7.1B), ClioQuery updates to show only those documents mentioning the query published during the selected interval. (Historians are often interested in specific time periods; see Section 7.3.) In Figure 7.1B, the user has filtered to documents published in 1983–1985.

The **filter-by-subquery** control allows users to select documents that mention some additional word, called a subquery. For instance, after a user queries for the Salvadoran leader "Duarte" they might wish to further narrow results to understand the relationship between "Duarte" and his ally U.S. President Ronald Reagan. To investigate this question, the user can enter the subquery "Reagan" to select all documents mentioning the word "Duarte" which also mention the query word "Reagan" (Figure 7.1C). We included this feature because complex Boolean queries are often popular with experts [216, Section 1.4]. More complex Boolean expressions are possible in future work.

The **filter-by-count** control filters results based on the the number of times a query term is mentioned in a document. When a user adjusts the filter-by-count slider to some value $K \in \{1, 2, 3, 4, 5\}$, all components of the interface update to show only those documents with K or more mentions. In cases where a user has set a subquery, the filter-by-count control allows the user to select documents with K or more mentions of the subquery. For instance, in Figure 7.1F, the user filters out documents which contain fewer than three mentions of the subquery "Reagan".

ClioQuery also helps users quickly see the count of query terms within documents using square-shaped, query-colored **count markers**, shown beside each document headline. Count markers use brightness to encode the count of a query term within a document. For instance, count markers for documents with more mentions of a query term have a darker purple color than count markers for documents with fewer mentions. If a user enters a subquery, count markers show the count of the subquery within each document, using shades of the subquery color (as in Figure 7.1F and 7.1H). This feature is inspired by TileBars [150].

7.4.7 Sentence simplification to help summarize a query

7.4.7.1 Overview of sentence simplification in ClioQuery

ClioQuery's Document Feed displays a query-focused summary of a user's query and subquery, by first extracting and then simplifying sentences mentioning query (or subquery) words. To simplify sentences, we turn to sentence compression techniques from the text summarization literature in NLP (introduced in Section 7.2.1.2). These methods try to summarize naturally occurring input sentences by removing words, to create shorter and well-formed output sentences, which contain the most salient information from the input. (A well-formed sentence is one that sounds natural, rather than garbled or choppy [298].) In particular, we turn to a specific class of sentence compression methods, which can ensure that simplified sentences both (A) fit within limited screen space in a user interface and (B) mention the user's query term or subquery term. Such methods are appropriate for ClioQuery because each line in the Document Feed has a fixed width, and must include some mention of the user's query or subquery.

More concretely, we use a *query-focused clause deletion* [143, 140] method to shorten sentences in cases when a user has entered a query (Section 7.4.7.2), and also use *relationship span extraction* [139] in cases when a user has entered both a

query and subquery (Section 7.4.7.3). We also employ a final fallback approach, *character windowing*, when it is not possible to shorten a sentence using other techniques (Section 7.4.7.2). In the next sections, we describe each sentence shortening method in greater detail, and then conclude by describing how ClioQuery handles cases when more than one technique might be used to shorten a given sentence (Section 7.4.7.4).

7.4.7.2 Query-focused clause deletion, and character windowing

ClioQuery's Document Feed requires shortened sentences that mention Q and fit within available screen space. We assume that such shortenings should also be well-formed and contain the most salient information from longer source sentences. Prior research in IR suggests that users prefer well-formed snippets [66], and prior work in sentence compression [189, 107, 109] strives for both well-formedness and salience. We also assume that methods for constructing shortenings must run with low latency, which is known to be important in user-facing analytics systems [205]. Different sentence shortening techniques might optimize for and manage tradeoffs between such requirements. But in this work we turn to a simple *query-focused clause deletion* method to meet such criteria, allowing us to focus on how to apply text summarization methods in user interfaces for historical research.

Query-focused clause deletion exploits the fact that natural language sentences are sequences of words, which exhibit hierarchical and nested grammatical structure [32]. For instance, the sequence "She swims in the pool" can be divided into interrelated word groups, with specific grammatical relationships; the words "in the pool" form a prepositional phrase that modifies the verb "swims." To represent such linguistic structure, clause deletion employs a dependency parse tree [259] grammatical formalism. A dependency parse is a directed tree graph with one vertex for each word in

the sentence, along with a latent root vertex.[9] Each subtree in the parse corresponds to a constituent subsequence in the sentence. The sentence simplification literature sometimes describes such subtrees as *clauses* [108]. Figure 7.7a shows an example dependency parse.

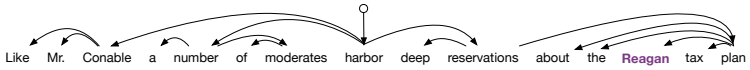
Sentence simplification via clause deletion shortens sentences by iteratively deleting clauses from a dependency parse.[10] Figure 7.7 shows how one sentence is shortened by iteratively deleting two clauses. Unlike sentence compression techniques which consider individual tokens for removal (e.g. Filippova *et al.* [109]), deleting clauses naturally identifies and removes groups of related words. For example, a single deletion could remove the prepositional phrase "after the election," or a much longer word group with more modifiers and embedded clauses: "after the previous election last year, which went poorly." Shortening sentences via clause deletion also makes it easy to ensure that output sentences must include Q; clauses that contain query mentions are not allowed to be removed during deletion.[11]

To try and create well-formed output sentences, ClioQuery turns to prior work on clause deletion [143, Section 6], which has found that in general removing more clauses from an input sentence makes it less likely that the resulting output sentence will be well-formed. Thus, to shorten an input sentence, ClioQuery's clause deletion first identifies those candidate output shortenings that can be constructed by removing at most K clauses from the input (without removing Q), and are also short enough to fit in one line of text within the Document Feed. Because in practice it is

[9]We use the UD (v1) dependency formalism [259]; other related formalisms allow for non-tree parses [296]. Eisenstein [100, Chapter 11] offers a broad introduction to dependencies. We perform dependency parsing using Stanford CoreNLP [217, 57].

[10]Tokens from the remaining tree are then printed in left-to-right order, based on their position in the original sentence.

[11]It is also possible to enforce such query constraints using integer linear programming (ILP). However, ILP-based sentence compression techniques (e.g. Clarke and Lapata [67]) are NP-hard and have been shown to be orders of magnitude slower than other iterative approaches to query-focused sentence compression [140].



(a) An untyped dependency parse tree of an input sentence mentioning Q="Reagan."

(b) To simplify the sentence, ClioQuery first removes the clause (subtree) "Like Mr. Conable."

(c) In this step, ClioQuery removes another clause that does not contain Q.

Figure 7.7: Sentence simplification via query-focused clause deletion [140, 143]. ClioQuery removes two subtrees from a dependency parse across two steps to simplify the input sentence (7.7a) into the output sentence (7.7c).

often possible to dramatically shorten an English news sentence by removing only one or two large clauses (for example, a lengthy relative clause, such as "Reagan met with the envoy ~~who was sent by the ...~~"), ClioQuery only considers shortenings which can be constructed by removing $0 < K \leq 2$ deletions.[12]

To try and ensure that output shortenings include the most salient information from input sentences, ClioQuery then returns the candidate output shortening with the highest tf-idf score [216]. Tf-idf scores are often used in extractive sentence compression [67, 108] and text summarization [82] to identify salient information for inclusion in summary output; the metric identifies words which occur with unusual frequency (relative to the overall corpus), which is an important signal of salience

[12]In addition to encouraging well-formed output, this strict limit ensures low latency for the user. For a sentence M words long, the worst case for performance is a tree where all words are leaf vertexes, resulting in $M + M(M-1)/2$ possible outputs of $K = 1$ or 2 deletions. But in typical trees, there are far fewer possible deletions because: (1) the query word and all its ancestors are not allowed to be deleted, (2) after the first deletion of a clause length C (i.e., the size of the deleted subtree) only $M - C$ candidates remain for the second deletion, and (3) if ClioQuery finds any candidate shortenings using K=1, it won't search for candidates using K=2, as shortenings which remove fewer clauses are more likely to be well-formed. We do not consider cases where $K = 0$, as most unshortened news sentences are too long to fit within the Document Viewer.

in summarization [249]. The Appendix includes details of how we compute td-idf in ClioQuery to identify words which occur frequently in documents mentioning a query.

In some cases, there is no way to shorten a sentence by removing one or two clauses while ensuring that that output sentence mentions Q and will fit in the Document Feed. In these circumstances, ClioQuery resorts to shortening the sentence by extracting the span of N characters to the left and right of Q in the sentence, where we maximize N under the constraint that the resulting character span will both fit in Document Feed and respect word boundaries. We use this *character windowing* method only as a last resort because it may cut off syntactic constituents (e.g. show only a portion of a prepositional phrase), which may create awkward-sounding output. Section 7.4.7.4 describes how often ClioQuery uses this fallback, during an example run of ClioQuery.

In the future, it might be possible to shorten more sentences with query-focused clause deletion by considering candidate output shortenings that are created using more than $K = 2$ deletions. (Prior work on query-focused clause deletion does not yet offer an efficient solution for considering such candidates [143].) Because the number of candidates grows with K, developing algorithms which efficiently search over possible outputs or learn greedy deletion policies based on data (e.g. with reinforcement learning) might offer useful starting points.

7.4.7.3 Relationship span extraction

ClioQuery users who search for a query term Q can also filter query results by a subquery. When a user enters both a query and a subquery term, we assume that they are broadly interested in how these two terms are related in the corpus. For instance, a user might query for the Salvadoran leader Q="Duarte" and apply

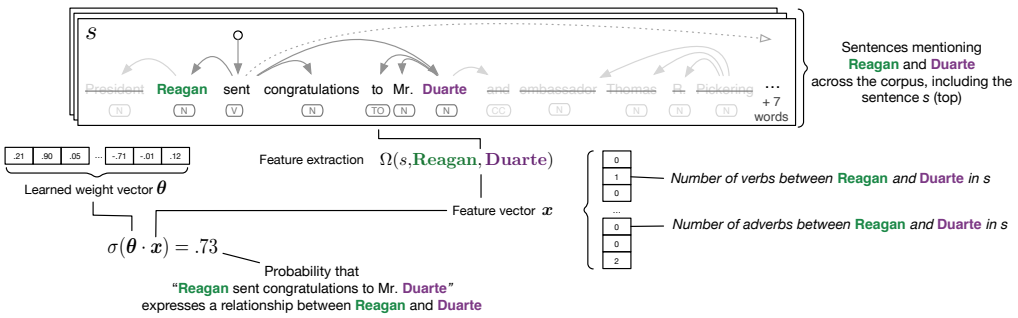


Figure 7.8: Sentence simplification via relationship span extraction [139]. This method predicts the probability that the span of tokens between the query and subquery words in a given sentence s will sound natural as a shorter and standalone sentence. (If the predicted probability is high, the sentence can likely be shortened to the span of tokens.) This figure shows the predicted probability that the token span between the query **Duarte** and the subquery **Reagan** will sound natural as a shortened sentence, when extracted from the longer sentence shown in Figure 7.1 (letter K). Seven words from s are not shown in the diagram above.

a subquery for the then U.S. President "Reagan," in order to understand Duarte's relationship to Reagan (Figure 7.1).

To meet this information need, ClioQuery attempts to simplify long and complex sentences mentioning both the query and subquery terms into short sentences which concisely describe the relationship between the query and subquery. We describe the process of shortening sentences in this manner as *relationship span extraction* [139], because each shortened sentence is a token span (i.e. sequence of tokens) extracted from a longer sentence. For instance, in Figure 7.1, we extract the span "Reagan sent congratulations to Mr. Duarte" from the longer sentence "~~President~~ Reagan sent congratulations to Mr. Duarte ~~and Ambassador Thomas R. Pickering pledged United States support for further meetings~~."

Relationship span extraction employs logistic regression to determine which sentences can be shortened to express relationships. It extracts a vector $\boldsymbol{x}$ of linguistic features from two input query words and input sentence s (e.g. is there a verb token between query words in s?) and then passes the dot product of $\boldsymbol{x}$ and a learned weight vector $\boldsymbol{\theta}$ through a logistic function σ to return a predicted probability that the token

span between the query and subquery will sound natural when removed from the sentence (Figure 7.8). The method is supervised using a sentence compression corpus [107]. In ClioQuery we shorten a sentence s to a relationship span if the predicted probability that the span sounds natural is greater than a threshold $T = 0.5$.[13] We implement with Scikit-learn [272].

7.4.7.4 Choosing among possible sentence shortening methods

The previous sections describe three different sentence shortening techniques, which are applied in the ClioQuery interface. Below, we describe how ClioQuery chooses to apply the three different methods.

After a user enters a query Q, for each document mentioning Q, ClioQuery's Document Feed displays the first sentence within the document mentioning Q that can be shortened via query-focused clause deletion. If no such sentence exists, ClioQuery resorts to shortening the first sentence mentioning Q via character windowing. (Character windowing is only used as a last resort because it does not attempt to create well-formed output containing salient words from the input.)

In cases when a user has entered both a query and subquery, for each document mentioning the query or subquery, ClioQuery will attempt to display the first sentence in the document that can be shorted via relationship span extraction. This is because we assume the user is interested in the relationship between the query and subquery. If there is no sentence that can be shortened via relationship span extraction, ClioQuery will display the first sentence that can be shortened via query-focused clause deletion. If no sentence can be shortened via clause deletion, it will resort to shortening the first sentence mentioning the query or subquery via character windowing.

---

[13] Setting a lower threshold T might increase the total number of shortened sentences, at the cost of creating fewer well-formed extractions (and vice versa).

In Figure 7.1, ClioQuery uses relationship span extraction to shorten and display some sentence from 31 out of 239 documents which mention "Duarte" and "Reagan". It uses query-focused clause deletion to shorten and display some sentence from 85 documents, and it resorts to character windowing for the remaining 123 documents.

ClioQuery also allows the user to click "expand" to see all sentences mentioning the query within the document (Section 7.4.3). In this case, ClioQuery will first attempt to shorten each sentence mentioning Q via query-focused clause deletion, before resorting to shortening the sentence with character windowing. If the user has also set a subquery (in addition to Q), ClioQuery will first try to shorten each sentence mentioning the query and subquery using relationship span extraction (and then attempt clause deletion, and character windowing).

7.5 Expert interview study procedure

After analyzing user needs (Section 7.3) and designing the ClioQuery system based on such needs (Section 7.4), we conducted an interview study over Zoom video chat to test ClioQuery with historians and archivists, who used the system to investigate questions from news archives.

7.5.1 Recruitment, participants and corpora

We recruited five participants from two universities in the U.S., by emailing students, faculty and staff listed on history and library department web pages. All participants had advanced degrees (master's or PhD) in history or library science, much like the expected users of our system. We summarize the backgrounds of participants in Table 7.6. Interviewees from our needfinding study (Section 7.3) did not participate in our expert interview study, to avoid what Sedlmair *et al.* describe as a potential form of bias [302]. Each participant in the interview study had an established research or curatorial interest in some topic related to late 20th century or early

ID	Research experience	Library experience	University role	Gender	Topic
P1	6	0	PhD candidate	Male	Iraq
P2	5	0	PhD candidate	Male	Zimbabwe
P3	4	0	PhD candidate	Female	combat
P4	20	0	Instructor/researcher	Female	wages
P5	0	3	History librarian	Male	copyright

Table 7.6: Interview study participants. We report history and library experience in years.

21st century history, which we express as a single topic word (see Table 7.6). We identified this designated topic word based on each participant's publication record and professional web presence. Before each interview, we then loaded ClioQuery with a corpus of *New York Times* (NYT) editorials[14] published between 1987-2007 [292] mentioning the designated topic word.

7.5.2 Data collection

To administer the study, one researcher from our group conducted five, one-on-one, sixty-minute interviews over Zoom video chat. (See supplemental materials for a detailed script.) During each interview, the researcher asked each participant to brainstorm and then articulate a high-level research question, based on the participant's prior work (10 minutes). They then introduced the participant to ClioQuery via a tutorial (7 minutes), asked them to investigate their research question using ClioQuery (30 minutes) and concluded with a semi-structured interview (13 minutes). Throughout, the researcher observed and recorded participant reactions and invited participants to think-aloud [257] as they used the system. If a participant offered feedback on some portion of the interface during their investigation (e.g. offered detailed feedback on the Time Series View), the researcher did not ask about this topic again during the semi-structured interview.

[14]Social researchers sometimes study editorials to better understand media sources [60, 209].

7.5.3 Thematic coding

The researcher who conducted the interviews analyzed automatic Zoom transcriptions for each of the video recordings, and corrected transcription errors. The researcher then extracted 183 quotes from across five interview transcripts. Each quote consisted of a few sentences on a focused topic, along with the preceding question or comment to provide context (e.g. a quote might discuss the Document Feed). The researcher attempted to extract as many quotes as possible, excluding irrelevant quotes (e.g. tutorial instructions).

The researcher then developed a codebook of six high-level codes (described in Section 7.6), by grouping and re-grouping the 183 quotes to identify common themes, much like the codebook-based approach described in Miles, Huberman and Saldaña [237, Chp. 4].[15] After each assigning each quote to exactly one of the six codes, the researcher shared the codebook with an undergraduate coder with training and experience in qualitative coding, who was not involved with the development of CLIO-QUERY. The second coder independently assigned codes to the same quotes, using the codebook. The second coder was also invited to add new codes to the codebook if needed, but reported that no new codes were necessary. (Thus we did not modify the codebook.) We include a copy of the codebook in supplemental materials.

Following independent coding of each of the 183 quotes, the two coders met for 1 hour over Zoom video chat to discuss 41 disagreements, and attempted to reach consensus via discussion. In 21 cases, the two coders were able to reach agreement regarding the appropriate code. In 20 cases, the coders determined that disagreement reflected genuine ambiguity in qualitative data, and agreed to disagree.

McDonald *et al.* [228, Section 2.2] use the term *reliability* to describe the extent to which coders reach the same result from independent work, and use the term

[15] Miles, Huberman and Saldaña [237, Chp. 4] describe assigning codes in two phases; we assign codes in a single phase.

agreement to describe the extent to which coders reach consensus after discussion. Adopting this terminology, we measure the reliability of the two coders by computing a Cohen's $\kappa = 0.724$ (using the R `psych` 2.0 package [283]), and we measure the agreement of the two coders by computing a $\kappa = 0.855$.

7.6 Expert interview study results

In this section, we describe the six themes that emerged from qualitative coding.

7.6.1 ClioQuery helps with historical sensemaking

While using ClioQuery, each of the five participants formed a question related to their research (Table 7.6) and then collected and interpreted evidence to start to answer their question, in a process Dalton and Charnigo [81] describe as historical sensemaking. We observed that ClioQuery helped historians in this process, offering partial validation for our hypothesis that ClioQuery features can aid historians in their work (Section 7.4).

For instance, as part of his research, P1 studies *New York Times* news coverage from journalists embedded with United States military units in the Iraqi city of Fallujah during the second U.S.–Iraq war. From prior study, P1 understood that embedded U.S. journalists often published news stories reflecting the perspectives of U.S. military leaders. But while examining mentions of "Fallujah" in *New York Times* editorials using the ClioQuery interface, P1 expressed surprise when finding a more nuanced perspective from the opinion desk. *"I didn't see nearly as much of the sort of sensational depiction of Fallujah, and the militants in Fallujah [in editorials] that I expect from embedded journalists [in news stories] ..."* he reported.

Similarly, P2 used ClioQuery to find confirming evidence of shifting U.S. perspectives towards Robert Mugabe. As P2 expected, early *New York Times* editorials from the corpus praised Q="Mugabe" as a liberator, but then began to criticize *"him*

as a bad statesman, as a tyrant and a dictator." P3 was likewise able to partially answer a research question with ClioQuery. She explained that while she had *"a deep knowledge of [women in combat]. I don't have a deep knowledge of what the [NYT] editorial board has to say about it."* Using ClioQuery, she found evidence of editorials using *"the gendered trope that women are supposed to be wives and mothers*." P5 also discovered an unexpected connection with musical copyright, while researching a hypothesis surrounding literary copyright. *"The parity [with the music service] Napster that's that's really interesting ... That's not something I thought about ... I was thinking ... definitely more in literary items because that's what I deal with."*

7.6.2 ClioQuery offers overviews and context

Participants offered detailed feedback on ClioQuery features during interviews, which often matched our design goals for particular components of the interface. To begin, three participants reported that ClioQuery's **Time Series View** offered a useful overview of the entire corpus, by directing their attention to salient time periods. P1 said the Time Series View was an *"easy way of visualizing"* corpus trends, and P5 suggested that the Time Series View might be helpful *"when students are kind of in that exploratory phase ... as a way of ... coming up with research questions."* P4 offered similar feedback. *"I really like this,"* she said. *"This looks really functional and really useful. I like how there is quite a lot of information packed in."*

P1, P2 and P5 reported that the **Document Feed** was a useful feature of ClioQuery, because it helped summarize query mentions. The Document Feed *"condenses all of the essential information and sort of leaves out all the extra stuff*," said P1. Said P2, *"I found [the Document Feed] useful, especially the expand button. If I click expand I can see a rundown of the mentions right after the title without seeing the article*." P5 reported using the Document Feed to *"do some ... simple kind of topic modeling in my own head ... just to see if I could pull out any ... themes there."*

P5 added that, *"having this here [i.e. Document Feed] is really helpful to kind of see what they're talking about."* P3 and P4 discussed the Document Feed while describing the importance of context in historical research; we include their feedback on this feature in Section 7.6.4.

Several participants also reported that the **Document Viewer** helped during their research. For instance, P3 reported that automatic in-text highlighting in the feed was very helpful. *"I'm a visual person. So I'm looking for the words. I like that they're in purple and green ... the words that you've given me the pop out ... and I can see if it's a pro or con article pretty quickly just from that.*" P2 said he used the Document Viewer to *"provide detail."*

P1, P2 and P5 noted that ClioQuery's linked **Document Feed and Document Viewer** served complementary purposes. They described how the Document Feed provided a summary of the query term, while the Document Viewer provided necessary and complementary details. *"You need both [the Document Feed and Viewer],*" said P2. *"With just the Document Feed I won't be able to get the full picture of the story. And with just the Document Viewer I will not be able to trace the mentions quite comprehensively and specifically.*" P2 then added, *"as a researcher, it's important to see things in detail. If you just conclude from what you see in the Document Feed you are not going to get an objective picture of the context of the story line. But if you see the Document Feed, see the mentions, see what they imply, and then you want to understand the context of the story you are going to get to the Document Viewer.*" P1 said, *"I like having both the Document Feed and the Document Viewer, side by side. [The Document Viewer helps with] reading for more depth when I want more depth and [the Document Feed] helps with ... quick scans pretty easy."* Similarly, P5 explained, *"I see [the Document Feed and Viewer] working together really well ... I start by looking at the feed to kind of pick out the articles that would want to kind of dive into deeper and then I go into the Document Viewer."*

P4 and P5 specifically mentioned that complementary linked views from the Document Feed and Document Viewer helped with **mention gathering and analysis**, as compared to a baseline keyword document search system. *"A lot of a lot of databases that we work with, do something similar to this [i.e. the Document Feed]*," said P5, while describing a search engine results page. *"But you often then have to click on the article to go into the article to get to that reading ... here it is nice that it was just kind of next to it and you can scroll through it."* Similarly, P4 described the difficulties of context switching between documents from the Google search engine results page. *"Obviously, it's is a time saver"* she said, comparing ClioQuery to the keyword document search system. *"You can tell ... just using the editorials at one newspaper."*

Two participants relied on ClioQuery's **filtering system** to investigate their research topics. P1 investigated the *NYT* editorial board's discussion of the query term "Fallujah" using the filter-by-subquery feature (e.g. searching for "Fallujah" and "resistance" or "Fallujah" and "terrorist"). *"It's pretty interesting to me that I get three hits with the words Fallujah and resistance and only one with the word terrorist,"* he said. *"That would suggest a certain orientation from the editorial board that will be unexpected*." P2 found the filter-by-count feature very helpful. *"Oh, this is good*," he said, while testing out the slider. *"It gets us through to the most important, the most critical pieces that we want to read*."

7.6.3 Comprehensive review has high costs

During needfinding, interviewees emphasized the importance of comprehensively reviewing all available evidence. However, to our surprise, during the expert interview study, P4 explicitly disavowed an obligation to search comprehensively. *"I don't feel like I have an obligation to look at everything,"* she said. *"I have an obligation to get an overview and I think you know, with a completely unscientific measure of, oh,*

I think I've got enough now." Similarly, P1 commented that, *"I don't think anyone actually does it [search comprehensively]."* He went on *"A lot of people pretend they do it ... [but] in terms of like visiting archives ... everyone's skimming ... they already know what they're looking for and they're just trying to find it."* P2 pointed out that comprehensive manual review was desirable but ultimately had high costs. *"I am not saying we should get rid of personal scrutiny, the way you do it yourself. [But] you want to save time. If you do it [i.e. read] one-by-one it wastes too much time*." We discuss ambiguity surrounding comprehensive review in Section 7.8.

7.6.4 Context is crucial, so some are wary of summarization

Like during needfinding (Section 7.3.2.3), participants often emphasized the importance of context in historical research, using the word context both to describe metatextual information (e.g. stories printed beside an individual article on a sheet of newsprint, or information about a publisher) as well as textual information (e.g. information described throughout an article, providing context for an individual fact). For instance, P3 described extensive research to prepare for oral history interviews in order to *"get that context to be able to ask them the questions that I asked them*." P2 also reported that context is *"very important"* for historians, as it *"helps you understand why things are what they are."*

Some users' emphasis on context informed their feedback on the Document Feed. While P1, P2 and P5 found the Document Feed useful (Section 7.6.2), P3 and P4 expressed reservations, because they felt they needed more context to reach conclusions. P3 took the more extreme position. *"For me, I don't know if [the Document Feed] is necessary*," she said. *"As a history scholar, you can't take things out of context. You need to know the bigger context."* On the other hand, P4 reported that she would need more context (i.e. longer extractions from news stories) before the feature would be useful. *"The more context I can take in within as compact a time frame*

and compact a format, but sufficiently informative [the better]" she said. *"But I think these [shortened sentences in the Document Feed] might have to be longer for that to work."*

7.6.5 Some report tradeoffs between neutral review and limited time

During needfinding and prototyping, interviewees often stressed the importance of avoiding possible bias from software in historical research. But during our expert interview study, P4 reported that she relied on black-box relevance models to direct her attention while searching archives. *"I do try to use the chronological sorting [when using ProQuest],"* said P4. *"But it is ... too much to wade through. If your corpus is reasonably big then you have to have a relevance kind of algorithm in there. Otherwise, it's just going to be too frustrating."* P4 also recognized that reliance on ranking introduces confounds. *"I think it would be appropriate to make people look at all of the irrelevant stuff,"* she said. *"So they realize the algorithm is pulling the relevant stuff for you ... but you can't make the search s*** for people just to sort of make that point."*

On the other hand, P5 liked how ClioQuery used filtering to avoid potential bias. *"I think it's better that its just showing everything,"* he explained. *"I prefer having everything there to kind of whittle down ... as opposed to having certain things like cherry-picked ... I guess it's never super clear to me why certain things might be moved to the top of results ... it raises questions about how things are ordered and how they're brought to light."*

As I3 predicted (Section 7.3.2.4), P1 described relying on the search function of the *New York Times* website [251], without understanding how the site was ranking search results by relevance. *"I wasn't super aware of how they were pulling up articles for me ... They rank it in terms of views right?"* he said. He added, *"I just don't, you*

know, have the knowledge of how to navigate these ... search engines well enough." We discuss heterogenous feedback on algorithmic bias in history in Section 7.8.3.

7.6.6 Access, integrity and integration in current practices

Many participants commented on the importance of access, integrity and integration in describing their current practices with newspaper archives (see also Section 7.9). P1 reported gathering news articles on U.S.-Iraqi relations from around the web "for years" by using search engines like Google or the *New York Times* website [251], saving these articles to the Internet Archive [71] and then organizing this collection using the software program Omeka [74]. This participant pointed out that ClioQuery *"assumes you have found all the stuff you want to work with,"* which is not true for his current research. P2 said that he had to rely on physical archives of print newspapers in Zimbabwe, which required burdensome international travel. P3 said that she rarely used newspapers in her own research, because many newspaper archives are often inaccessible behind paywalls, and P4 emphasized the need for better optical character recognition technology to improve search over printed newspapers. P5 reported that he *"used Zotero a lot"* to store and organize archival sources; he liked that Zotero is open-source and integrates with Microsoft Word.

7.7 Field study

In their review of design study methodology, Sedlmair *et al.* emphasize the importance of deploying a designed solution "in the wild" to test if new software helps "real users" solve "real problems" with "real data" [302]. Thus, we deployed ClioQuery over the web in a field study for two historians, who used the tool to answer questions from their own research. Unlike in the expert interview study, during the field study, historians investigated questions over multiple meetings, and tried to reach substantive rather than preliminary conclusions. We believe that this evaluation of-

fers more realistic but also less uniform feedback than the one-hour expert interviews described in Section 7.5.

7.7.1 Procedure

We recruited two historians, $H1$ and $H2$ (Table 7.7) through convenience sampling [124]. $H1$ and $H2$ did not participate in the initial design or development of ClioQuery, to avoid what Sedlmair *et al.* [302] describe as a potential source of bias. During the field study, one member of our research team conducted three one-on-one meetings with each historian over Zoom video chat. The first meeting was 30 minutes long and the subsequent meetings were 60 to 70 minutes long, with 1 to 3 weeks between each meeting. Each meeting in the three meeting sequence had a distinct focus. During the first meeting, the researcher presented a tutorial of the software, described the field study process, and invited the historian to describe a question related to their research. After the first meeting, a member of our research team gathered the data needed to answer the historian's research question and loaded it into ClioQuery (the Appendix describes this data gathering). During the second meeting, each historian learned to use the ClioQuery software, and performed a preliminary exploration of the data. Then, during the final meeting, each historian investigated some specific query by analyzing the comprehensive set of all mentions using the Document Feed and Document Viewer. During each meeting, the researcher observed each historian and invited the historian to think aloud [257] as they used

ID	Research experience	Library experience	Academic role	Gender	Research area
H1	5	0	PhD student	Male	Media and society
H2	25	0	Tenured faculty	Female	Space exploration

Table 7.7: Historians in the field study. History and library experience are listed in years.

the system. The researcher also asked the historian to describe their findings, and explain how ClioQuery helped or did not help answer their research question.

7.7.2 ClioQuery helps experts investigate by skimming

During the field study, $H1$ and $H2$ each used ClioQuery to reach substantive historical conclusions, offering additional evidence for our hypothesis (Section 7.4) that ClioQuery can help experts answer research questions from news archives.

$H1$ used ClioQuery to verify a well-known claim from Herman and Chomsky, who argue that for-profit news organizations in the United States shape public opinion towards the interests of political and economic elites [155]. To offer evidence for this theory, in their work, Herman and Chomsky assert that *The New York Times* wrote five articles in February and March of 1984 describing the Salvadoran army as a protector of El Salvador's election. To verify this result, $H1$ searched a *New York Times* corpus (see Appendix) for the query "election" and then used the filter-by-date feature to select articles from February and March of 1984. $H1$ then used the filter-by-subquery feature to identify those query results which contained the subquery "army" and then systematically reviewed all 32 matching documents, through what $H1$ described as *"skimming highlighted parts"* in the Document Viewer. By using ClioQuery in this manner, $H1$ said that they were *"able to find what might be the five articles"* Herman and Chomsky used to partially support their conclusions. $H1$ explained, *"The tool is great for exactly this."*

$H1$ found ClioQuery's in-text highlighting helpful for their research task, drawing a comparison with a baseline keyword document search system (Section 7.2.2.1). *"I like how you have the bold highlighted and colored words in the text itself,"* they said. *"That is the advantage that this interface has over the New York Times website."* $H1$ also explained how such highlighting reduced reading burden, compared to keyword document search, which allowed him to skim relevant documents. *"What I*

need to know is the army described as a protector of the election [in an article]," he said. *"I don't need to read every word of the article to find that out. I can look at the paragraphs where they are describing the army and I see what they are saying in those paragraphs. That is pretty useful."*

*H*2 chose to use ClioQuery to study how the United States media represented female astronauts Svetlana Savitskaya and Sally Ride in the early 1980s, to gather evidence for an upcoming book about gender and space exploration. To investigate, *H*2 used ClioQuery's Document Feed and Document Viewer to review portrayal of Sally Ride in *The New York Times*. *H*2 queried for the word "Ride" and then scrolled through the Document Feed to skim over mentions of Ride in the 63 matching documents, sometimes also clicking to open individual news stories in the Document Viewer. *"I have some hypotheses that I was able to develop very quickly through the experience of using this [system]*," *H*2 reported. *"One is that Ride was presented to the American public [in The New York Times] ... first as a woman and second as a scientist"*. *H*2 asked us to continue to provide access after the study, so she could continue researching her book using the tool.

ClioQuery's Document Feed was particularly helpful for *H*2, who found that query-focused summarization offered an advantage over a baseline keyword document search system. Ride was a PhD astrophysicist turned astronaut, and *H*2 wanted to understand how the media portrayed her scientific credentials. The Document Feed helped *H*2 quickly review this information. *"[Here] she's called a flight engineer*," *H*2 said, pointing to the Document Feed. *"I can see this already [without opening the document]."* *H*2 then scrolled through the Document Feed to find shortened sentences where Sally Ride was described with her academic title (Dr. Ride), and sentences where Ride was described (or not described) as a physicist. *H*2 explained that she could identify this information *"just doing the quick scan [in the Document Feed]"*. She went on to explain how she would normally research this question with

The New York Times archive by opening and reading individual news stories using a web browser. *"The question is*," she said, *"what can I do here [with* ClioQuery*] that I can't do there [i.e. on The New York Times website]?"* $H2$ continued, *"It's exploring the left hand Document Feed here. This is awesome ... I am liking these short contextual pieces [i.e. shortened sentences]"*. We illustrate this comparison in Figure 7.2; by using ClioQuery, $H2$ was able to easily gather and analyze mentions of Ride across the corpus.

7.8 Discussion

7.8.1 New features and directions for text analysis

Much prior work in interactive text analysis focuses on helping users gain overviews of corpora that are too large to read (Section 7.2). For instance, systems and frameworks from the Visualization and HCI community such as TIARA [203], TextFlow [79], Termite [63], Overview [43], ParallelTopics [92], RoseRiver [80], ConVisIT [162] and Vis-Kt [260] from venues such as TiiS, IEEE TVCG, IEEE VAST and AVI focus on helping users explore overall themes in corpora by interacting with clusters of words and documents.

But because historians often need to gather and analyze mentions of *specific* query words in large archives, ClioQuery instead applies text summarization technologies developed within NLP venues such as ACL and EMNLP to show skimmable summaries of a query term across a corpus. The system then presents such summaries using linked views and in-text highlighting, to help users easily review summary text in underlying documents. During expert interview and field study evaluations, many historians said that they found such features helpful for archival research. They reported skimming over query mentions in the Document Feed to gain a sense of a query's use across a corpus, and then reading highlighted mentions in the Document

Viewer for more context and detail. Several specifically mentioned that these components helped with mention gathering and analysis.

This query-oriented approach suggests new directions for interactive text analytics, where ideas and features from ClioQuery might be applied in new query-oriented systems, or might be applied to extend existing overview-oriented tools. For instance, users might formulate a query using overview-oriented features such as word clusters, and then investigate this query using a ClioQuery-style summary and full-text view.

Prior research in interactive text analytics suggests the need for such an approach. For instance, reflecting on the development of Overview [320], author Jonathan Stray notes that users often "knew what they were looking for in advance," which stands "in contrast to the large research literature concerned with 'exploring' a document set". Similarly, reflecting on the development of Jigsaw [126], Görg *et al.* describe the importance of helping users review underlying documents. They write that "interactive visualization...cannot replace the reading of reports". Such conclusions from prior work seem to reinforce our findings, suggesting a role for query-focused summarization and linked full-text views in interactive text analysis.

7.8.2 User feedback on summarization has implications for NLP

ClioQuery applies particular ideas from query-focused summarization in NLP for interactive text analysis. However, building and evaluating a user-facing summarization system for historians forced us to reexamine several core assumptions from the literature on text summarization. In particular, early versions of ClioQuery applied standard optimization-based summarization methods [229] to select "important" information from a corpus, much like prior temporally-oriented language engineering systems such as HistDiv [311], TimeMine [8] and TimeExplorer [225], which each automatically identify relevant information, based on a query.

However, during needfinding and prototyping, we found that some historians and archivists strongly disliked this approach. Experts reported that they needed to understand why the computer was showing particular summaries, before they could actually draw conclusions from the output (see Figure 7.4). Based on this feedback, in later versions of ClioQuery, we stopped trying to extract "important" mentions of a query term in search results. Instead, we decided to shorten and present every single sentence mentioning a user's query in the Document Feed (see Figure 7.4), and allow users to easily examine such shortenings in context in the Document Viewer. During our expert interview and field study and evaluations, we found that this approach was more successful. We hypothesize that experts liked this format, because they could understand why ClioQuery showed query shortenings, and thus use ClioQuery output in their research.

Our experiences might have implications for NLP, where research in summarization typically focuses on generating summaries which best match "gold" references [82, 248], without worrying about how to explain how summaries are formed. In particular, much recent work on abstractive summarization in NLP [291, 156] seeks to generate summary passages that do not occur in the input text. Because such abstractive output can not be checked against underlying sources, and because such methods also currently suffer from frequent factual errors [191], much more research towards trustworthy abstraction may be required, before these approaches might be applied towards social research.

7.8.3 Supporting comprehensive and unbiased analysis

During needfinding interviews, historians and archivists often emphasized the importance of directly and comprehensively examining all evidence relevant to a given research question, without allowing black-box algorithms to influence their conclusions. We thus designed ClioQuery to help users keep track of which query men-

tions they have reviewed in a corpus, without potential bias from algorithmic ranking. Yet feedback on these aspects of ClioQuery was mixed (Section 7.6.3 and 7.6.5). While some appreciated how ClioQuery used filters instead of ranking to narrow down search results, others reported that truly forgoing algorithmic curation required the researcher to spend too much time reading irrelevant documents. For instance, some users admitted that they often no have choice but to trust computer models of relevance to find evidence in archives, because keyword search often turns up far more documents than they can possibly review.

Why did some users express deep commitment to full manual review of evidence during needfinding interviews, while other users admit that they had to trust search engines to select evidence during system evaluation? There seem to be at least two possibilities. One possibility is that historians and archivists might express commitment to comprehensive review when describing their ideal practices, but remember the limitations of this ideal when faced with a real task during system evaluation. (Some approaches to needfinding in HCI emphasize the limits of user interviews [41]). Another possibility is that there is simply some variation in our users' commitment to comprehensiveness. Some but not all historians may feel required to comprehensively review all evidence during research, possibly based on intellectual background or subfield.

Future researchers might resolve the contradiction between experts' stated commitments to comprehensive review and the realities of inevitable tradeoffs between recall and time [277, Fig. 6] with improved user interfaces. Specifically, systems might transparently show which documents are selected or hidden by an algorithm, and allow users to easily override and investigate any document ranking decisions from a machine. Research on tools for visually and interactively refining search results [300] might offer a useful starting point.

7.9 Limitations and future work

Because it was difficult and expensive to recruit and interview highly-trained experts, in this study we report results from in-depth meetings with a small sample of humanists. While such one-on-one interviews provided rich feedback, the opinions of our participants likely only approximate the true requirements of all historians and archivists. In the future, we thus plan to take steps to facilitate adoption, in order to learn more about user needs. In particular, we found that historians have to collect, organize and sometimes digitize news stories, before they are ready to gather and analyze query mentions (Section 7.6.6). We thus plan to add features for importing news stories into ClioQuery from existing tools like Zotero and the Internet Archive. Additionally, throughout this work, we assume that query mentions are defined by exact string matches. This simplifying assumption allows us to focus on user experience and interaction, but has clear limitations. For instance, authors sometimes refer to "Reagan" using the nickname "Dutch". Automatically detecting such aliases (and other deviations from exact string matching) will be important for future work.

7.10 Conclusion

In this study, we considered how to design an interactive text analytics system to help historians in their practice of gathering and analyzing mentions of a free-text query in a newspaper archive. To create the system, we first studied the needs of historians and found that analyzing change across time, undertaking comprehensive review of evidence, evaluating contextual information and conducting neutral observation are each central to the practice of historical research.

We then designed the ClioQuery system based on such findings. ClioQuery uses methods from NLP to create a skimmable summary of the comprehensive set of all mentions of a user's query term in an archive. The system also uses linked views and automatic in-text highlighting to show summary text within underlying source doc-

uments, to facilitate review of query mentions in context. Moreover, because change across time is central to historical research, ClioQuery includes a navigable time series visualization to offer an overview of temporal trends in query frequency. Finally, ClioQuery also tries to engender trust by avoiding potential sources of unwanted algorithmic bias. For instance, the system offers filtering features to narrow results in a neutral manner, instead of ranking results with black-box algorithms (which might influence research).

We tested such features in two separate user studies with historians, where we found that ClioQuery's approach to organizing and presenting query mentions could help users answer real questions from news archives. Many historians reported that ClioQuery's Document Feed facilitated rapid analysis of query mentions, and that ClioQuery's linked Document Viewer offered complementary context and detail.

Such findings suggest possible new features and directions for user-facing language systems, particularly in query-oriented settings. Where much prior work in interactive text analytics focuses on providing corpus overviews, ClioQuery instead focuses on helping users gather and analyze query mentions, through skimmable query-oriented summaries linked with a full-text view of underlying text. This same feature set might be used to extend existing overview-oriented tools (e.g. Themail [335]), or in new systems for query-oriented corpus analysis. For instance, some marketing applications suggest notable keywords from user comments in online forums [125, Section 4.1]. ClioQuery features might thus be applied to help marketers gather and analyze mentions of such keywords, or to help other users investigate other queries in other domains.

PART IV: CONCLUSION

CHAPTER 8

CONCLUSION, LIMITATIONS AND FUTURE WORK

Conclusion

This study defines and introduces lexical corpus analysis, in which people formulate and answer qualitative research questions by (1) reviewing a corpus vocabulary, and (2) examining items from that vocabulary in context. For instance, traditional concordances are one particular tool for lexical corpus analysis, because they present a unigram vocabulary and help users review items from that vocabulary amid immediately surrounding text. However, throughout this thesis, we argue that there are many other possible approaches to lexical investigation (e.g. Figure 1.5). For example, like a traditional concordance, a ConceptMap interface (Chapter 3) both defines a corpus lexicon and presents items from that lexicon in context, in order to help people make sense of text.

We propose that there are three fundamental research questions underlying such lexically-oriented corpus investigation:

R1: How can lexical systems represent a corpus vocabulary to reflect human mental and linguistic models of a domain?

R2: How might lexical systems show query terms in context, to best satisfy user search need?

R3: How can we build new lexical systems for specific user groups?

We investigate these questions in Part I through Part III of this thesis, where we consider and evaluate possible solutions in terms of metrics like computational efficiency,

usability and domain independence, which we argue are crucial to user-facing corpus investigation (Section 1.5).

Together, our study defines and introduces lexical corpus analysis, provides a collection of new NLP methods to facilitate lexical corpus analysis, and offers two investigations into specific lexical systems created to assist particular user groups. In computational evaluations, we demonstrate the ways in which our proposed NLP methods offer advantages over baseline techniques (e.g. lower latency, Chapter 5); and in human evaluations, we demonstrate the ways in which new lexical interfaces offer both quantitatively measurable (e.g. faster task completion, Chapter 6) and qualitative benefits (e.g. comprehensive presentation of evidence, Chapter 7).

Yet while we make measurable progress towards addressing R1, R2 and R3, our work is plainly incomplete. In the next sections, we describe both the limits of our efforts, and our planned steps towards addressing those limits in future research. We then conclude by reviewing broad lessons from our experiences with user-facing NLP.

8.1 Future work towards representing the lexicon

In the first chapter of this work, we argue that gaps between machine representations of a corpus vocabulary and human mental models of a lexicon hinder the effectiveness of tools for lexical corpus analysis. We then ask how systems might represent a corpus lexicon to better reflect the ways that people think, talk and write about the terms and concepts in a domain. In Part I, we offer a partial answer to this question through the NPFST method, which efficiently extracts important multiword phrases across heterogeneous corpora. Yet while NPFST offers improvements over a bag of unigrams, our effort plainly falls short of our distant goal of machines which can mirror human mental models of a corpus vocabulary.

One particular gap in our representation of a corpus lexicon stands out as most in need of immediate future attention. Throughout this thesis, we typically assume

that each lexical item or concept in a corpus is defined by a particular string. We then assume that only passages which contain the string mention the lexical item. For instance, in Chapter 3, we assume that only those sentences which contain the exact string "President Clinton" mention the once U.S. leader. While this simplifying assumption offers advantages,[1] we believe that systems which rely on exact string matching will violate user expectations in two important ways, necessitating further work.

First, exact string matching will introduce false positives, in which the user is shown passages that do not actually mention their lexical query. Some false positives will arise from entities with the same name. For instance, the query "Jamaica" may refer to the Caribbean nation "Jamaica" or the New York City neighborhood of "Jamaica" Queens. Other false positives will arise from less concrete queries such as "peace treaty," which may refer either to the general idea of a peace treaty, or to a specific peace treaty during a specific conflict. Such false positives may mislead the user (e.g. a CLIOQUERY summary might describe events in the wrong "Jamaica"), or require the user to take on the unnecessary burden of reviewing irrelevant evidence.

Exact string matching will also introduce false negatives, in which a system will not show the user passages which do in fact mention their query. For instance, the concept "Bill Clinton" may be referenced via the strings "President Clinton," "U.S. leader" or "White House" (as in "White House strategy"). But passages containing such query mentions would not be identified with exact string matching techniques. This limitation in recall is especially important in fields like history, which emphasize comprehensive review of all available evidence (Chapter 7).

---

[1]Exact string matching allows us to focus on specific research objectives (e.g. designing a text analytics system for historians, Chapter 7), ensures interpretable output (a person can understand why a computer believes a given passage references a lexical item), and is clearly sufficient to help some users in some circumstances (e.g. journalists, Chapter 7).

It it likely possible to apply existing work in NLP to partially address such shortcomings. Most obviously, future research might employ tools and methods for resolving coreference, which seek to identify token spans that refer to the same entity across one or more documents. (Eisenstein [100] offers an introduction to this well-studied area.) For example, coreference resolution methods might help detect that the spans "President Aristide" and "Haitian leader" each refer to the once Haitian President (in a particular news article), or help with the challenging task of distinguishing between occurrences of identical strings that reference different entities (e.g. occurrences of "John Smith", which refer to different people [15, Sec. 6]).

Other closely-related work may also help address false negatives. For instance, work on entity linking [305] tries to identify spans of text which mention known named entities listed in a knowledge base (KB). Future lexical systems might build on or apply this work to help identify mentions of some free-text queries.[2] For example, if a user queries for "Clinton," a lexical system might use entity linking techniques to identify spans which refer to the "U.S. leader". Work concerned with cataloging [238] and detecting [149, 288] lexical relations like hypernyms, or identifying word senses [246] might also be applied in similar circumstances. A user searching for mentions of the word "general" might wish to see passages which include the hypernym "commander"; and a user reviewing mentions of "Obama" might be interested in passages containing certain senses of the phrase "White House" (e.g. a "White House strategy").

Yet while prior study offers tools, methods and ways of thinking which could extend our work, we expect that applying this existing knowledge towards user-facing lexical systems will present new research challenges, related to broad and unsolved questions in NLP. For instance, many current methods for coreference resolution

[2]Other lexical queries (e.g. "peace treaty") will be insufficiently concrete to be listed in a KB.

rely on domain-specific supervision. Applying such supervised techniques in general-purpose corpus analysis tools (e.g. for eighteenth-century court records, Chapter 2), would present a challenging domain adaptation problem. Similarly, much work on entity linking assumes access to a complete KB. But knowledge bases will not cover all areas that a user wishes to explore; in our work on ROOKIE (Chapter 6), we found that Wikipedia did not contain entries for many prominent politicians in a local news corpus. Therefore, we expect that applying entity linking methods towards lexical corpus analysis would require adapting research focused on creating knowledge bases, possibly using distantly supervised [239] or cold start [20, Sec. 6.5] techniques, intended for settings with limited supervision. Finally, from an HCI perspective, applying any predictive method in a user-facing system will open questions surrounding how to set and meet user expectations for an intelligent interface. How to best design intelligent user-facing systems is a subject of ongoing research [10].

8.2 Future work towards showing lexical items in context

In Part II, we consider how to show lexical items in context in order to satisfy user search need. While our work offers measurable advantages over baselines (e.g. higher yield or lower latency, Chapters 3 and 5), our limited efforts still have clear shortcomings which might be addressed in future research.

Reconsidering extractive and single-sentence summarization

Most immediately, much of our work in Part II focuses on single-sentence, extractive summarization; we try to create query-focused summaries by selecting and shortening individual sentences (e.g. Chapter 7). This single-sentence and extractive orientation allows us to simplify and thus make progress on new tools for lexical corpus analysis. But the approach has limitations. In some cases, salient information may be scattered or repeated across multiple sentences from one or more documents,

or expressed with unnecessary verbosity. Therefore, in the future, we might investigate sentence fusion methods [27] which seek to synthesize salient phrases from across multiple sentences to create a new single summary sentence. We might also investigate similarly abstractive techniques for paraphrasing [118] wordy portions of source documents, to help make better use of a summary budget. However, because these approaches risk confusing or misleading the user (see Section 1.5 and Chapter 7), future research might focus on developing accurate, trustworthy and interpretable abstractive techniques.

Developing corpus resources

Our work in Part II is also hindered by a lack of corpus resources, which are often central to advances in NLP. Ideally, we would like to have a large, multi-domain dataset of queries, documents and reference summaries, constructed to reflect a user's information need.[3] Yet historically, NLP has created few annotated resources for query-focused, multi-document summarization [268], perhaps because constructing such corpora can be expensive and difficult. (We describe our own experiences in Chapter 3). Given such challenges, future research might focus on creative and automatic methods for constructing evaluation corpora, much like recent work towards creating resources for traditional, single-document summarization [132, 156]. Alternately, future work might focus on adapting existing datasets for our lexically-oriented setting (as in Chapter 5). Regardless of how we construct corpus resources, we expect that evaluating machine-generated text against any sort of reference summary will require attention and effort; there is much active research into automatic metrics for text generation tasks [52], such as summarization.

[3]Kryscinski et al. [191, Sec. 3] suggest expressing the user's need as a question. They also point out the importance of minimizing noise in creating corpus resources for summarization.

Moving past a binary definition of salience

Throughout our work, we imagine that the user wishes to review mentions of Q in context. Therefore, we assume that including Q in summary output is sufficient for presenting the most salient information in passages or documents. In Chapter 4, which focuses on single-sentence summarization, we say that this assumption imposes a binary definition of salience (also called "importance").

This binary definition has obvious shortcomings. For instance, a user reviewing mentions of Q ="Aristide" might also wish to see information about related concepts, such as General Raoul Cedras. (Cedras lead a coup against Aristide in 1991, see Chapter 6). In theory, work from query-focused or multi-document summarization [82] might help identify such query-related related concepts using patterns in word frequencies [249, 83, 135]. Therefore, in the future, equipped with corpus resources, we might experiment with applying such methods in our lexical setting, in order to expand our binary definition of salience.

Using language models to reduce reliance on supervision

In this work, we use natural supervision and human annotation to try and identify well-formed summary output (e.g. Chapter 3 or Chapter 4). This reliance on supervision makes it harder for future researchers to build on our work. For instance, our supervised method for creating ConceptMaps (Chapter 3) likely could not be used to build different kinds of lexical interfaces. Moreover, our annotated dataset from Chapter 4 likely does not cover many aspects of how people perceive shortened English sentences, and could not be applied to shorten sentences in other languages.[4]

[4]We do try our best to mitigate the risk of overfitting to supervision. In Chapter 3, we test our approach on multiple kinds of text, to try and reduce the risk that our headline-supervised method only works on news stories. Similarly, in Chapter 4, our work is motivated by a belief that modeling human cognitive processes can be used to create NLP methods which are less tied to training data.

Given these limitations, we believe that future research should explore language modeling to predict well-formedness. A language model (LM) assigns a probability to a sequence of tokens [100, Chp. 6]. Prior study from Lau et al. [195] suggests that LM predictions correlate with human sentence-level judgments of well-formedness. Therefore, in the future, LMs might be used to identify well-formed summaries, much like how work in MT (e.g. Koehn et al. [190]) uses language models to identify more and less fluid translations.[5] The chief advantage of this approach is that LMs learn parameters from unlabeled text, and thus do not require specialized supervision; LM-based approaches could be trained on any corpus that a user wishes to explore. Future LM-based techniques might also leverage a large body of engineering-oriented (e.g. KenLM [148] or RoBERTa [204]) and experimental work [59, 177] on language modeling, in service of specialized user-facing systems.

Nevertheless, exploring LM-based approaches to predicting well-formedness would present research challenges. To begin, language models may have many parameters, which will have to be stored and updated using compute resources. Moreover, current state-of-the-art neural language models such as BERT [87] require access to a specialized GPU. Future research might consider how to best use such large language models in computationally-limited settings (e.g. a laptop with a single CPU). KenLM's efficient implementation of a traditional N-gram language model [148], and current study focusing on reducing the size of large pretrained language models such as BERT [287, Section 6.2] may suggest possible paths forward.

Yet applying LMs to predict well-formedness also presents algorithmic challenges. If every word within K tokens of an occurrence of Q may appear in a summary, then there are 2^{2K} possible ways to represent a single occurrence of Q in context. (Any word within the $2K$ window may be included or not-included in the output.) Searching this

[5]Using LMs for summarization is not new. For instance, Clarke and Lapata [67] use language models for sentence compression.

large space to find the most well-formed extractions (according to a LM) presents a research challenge. Approaches to decoding from machine translation [250] and work on applying optimization techniques towards text summarization [224, 202] may offer useful starting points.

Finally, note that while we expect that LMs will prove useful for predicting well-formedness, linguists and cognitive scientists do disagree about the extent to which probabilistic methods can explain or predict human linguistic behavior. Both Chater and Manning [56] and Lau et al. [195] each describe these debates. Within NLP, recent successes of language modeling for downstream tasks have inspired a corresponding literature focused on identifying shortcuts or heuristics underlying apparent linguistic reasoning. Rogers et al. [287] offer a partial review of this work, focusing on BERT. Ultimately, limits of language modeling in representing human linguistic behavior could imply corresponding limits in the use of language models to predict summary well-formedness.

Considering multi-sentence properties of output summaries

Finally, future work might consider other multi-sentence properties of output summaries. For instance, systems might take care to include important background information from one sentence in order to frame foreground events in another sentence; such as mentioning the fact that "Cedras took power" before mentioning that "Aristide fled" (Chapter 6). NLP offers a number of different ways to represent such discourse phenomena [100, Chp. 16], which might be explored in future work. Formalizing and accounting for such discourse structures would likely prove challenging, but offers the potential to improve summary output.

Attention to complex semantic relationships between sentences, such as the relationships described in MacCartney and Manning [210], might also be helpful for showing query mentions in context. For instance, in Chapter 3, we discuss how a

summary which asserts that a couple has divorced should not waste budget informing readers that the couple once married. In this case, determining that one textual unit entails another (i.e. if one unit is true, the other must be true [210]) allows some sentences to be omitted from a summary. Similarly, if a summary presents an apparent contradiction (e.g. asserting that both Aristide and Cedras hold power in Haiti), it may have to also include other supporting information to resolve and explain the inconsistency. Thus, research on identifying and representing complex semantic relations between sentences should be explored in future research.

8.3 Future work towards user-facing lexical systems

8.3.1 Needfinding from text at scale

In Part III of this work, we argue for the importance of researching the needs of individual user groups to design new systems for lexical corpus analysis. We then demonstrate the importance of such user research in Chapter 7, by conducting and analyzing user interviews for our ClioQuery software. In our work on ClioQuery, interviews helped us correct a (deeply flawed) initial system design, provided nuanced insights into how historians search, and revealed limitations of current summarization methods in NLP. Yet while one-on-one interviews proved valuable, we still observed well-known limitations in this traditional HCI research technique:

- Because we had to schedule, conduct and code each meeting with an individual participant, collecting qualitative user data for ClioQuery was time-intensive. Developers of future lexical systems who meet with users would incur similar time costs.

- Because we could only conduct a small number of interviews, we ran the risk of drawing conclusions from what Shneiderman [309] describes as a "small samples of users who offer biased perceptions or reports." While speaking with a few

participants selected via non-probabilistic sampling provided rich and open-ended feedback, findings from our ClioQuery user studies may not generalize beyond our small sample.

- Because our meetings were only one hour long, we could only discuss limited topics during limited time with experts. Manual data gathering thus imposed limits in the coverage of our data.

- Perhaps because interviews relied on self-reported descriptions of historical research practices, we observed some puzzling inconsistencies in expert feedback. Some participants told us that social researchers should never draw conclusions based on curated evidence from opaque software. But others admitted that in practice they do trust ranking results from black-box search engines. This inconsistency might reflect a known limitation of interview-based methods. As Blomberg and Burrell [41] note, "what people say they do and what they actually do are frequently quite different."

Noting such strengths and limitations, we propose that interviews offer just one particular way of "studying people to identify their unmet needs [269]," a process known in the HCI literature as needfinding.[6] We then propose a broader study of NLP-based needfinding in future work.

Our planned research begins from the observation that people use text to express their frustrations, desires, goals, practices and reasoning across heterogeneous digital platforms such as social networks, collaborative workspaces and free-text surveys. In the future, HCI researchers, software organizations and technology entrepreneurs might use insights gleaned from analyzing this text data with NLP methods to enrich understanding of users at scale.

[6]Stanford engineer Robert McKim is said to have first coined this term in the 1970s [269].

NLP offers many possible advantages for needfinding:

- NLP methods might draw insights from large datasets, to achieve broad coverage. For instance, NLP approaches might help identify rare user experiences or long-tail use cases, which might not be detected in small ethnographic or interview studies, due to limited manual data gathering. Prior work has already started to explore this idea. For instance, Schaffhausen [293] offers a dissertation focused on analyzing large numbers of need statements. Similarly, Griffin and Hauser [131] try to estimate gaps in coverage by modeling the probability that a given customer voices a given need during an interview.

- NLP offers the possibility of quantitative measurement of user requirements. For example, NLP might might help researchers reason more accurately about the incidence of particular use cases, which could help prioritize the application of limited design or engineering resources. Estimating the incidence of user requirements from small, traditional interview datasets would likely be very noisy.

- NLP methods might lower the costs of interview-based needfinding research. For instance, one frequently-cited study from Griffin and Hauser [131] found that 20 to 30 one-hour interviews coded by four to six analysts were needed to identify 90 to 95 percent of customer needs (for a given product). The authors also estimate the financial costs of an interview at around $1000 to $2000 (at a U.S. company, in 1993). Reducing the costs of needfinding could enable faster iteration and time to market.

- Finally, NLP methods scale to large datasets. Therefore, like traditional survey research, NLP might mitigate some threats to validity from a small sample size.

Yet while computational methods for analyzing text data offer exciting possibilities, in many ways applying NLP methods to perform needfinding extends existing re-

search traditions in HCI.[7] For instance, motivated by inevitable gaps between human words and actions, some HCI researchers already turn to ethnographic needfinding techniques to gain insights about users, via digital and non-digital observational data like field notes, video recordings, social network structures, blogs, sensors, social media posts and computer logs [41, 244, 263]. NLP methods for analyzing user-generated text can be thought of as a way of supporting these existing needfinding practices at scale.[8] Similarly, NLP can be thought of as a particular tool for mixed methods HCI research [334], which seeks to combine the strengths of qualitative and quantitative inquiry [76].

Of course, like any research technique, NLP needfinding methods would also have downsides and limitations. First, users who comment about needs and practices on any particular channel (e.g. on Reddit.com), may not represent the broader user population. Thus, like traditional interview research, NLP-based needfinding may suffer from threats to validity from sample bias. Second, while it might be possible to design around limitations of predictive text processing [10], HCI practitioners may still be skeptical of NLP methods, which might limit adoption. Limited evidence already suggests that some practitioners may not trust qualitative conclusions from automated systems [174], and may dislike the idea of replacing a traditional codebook with a machine-generated schema [218]. Third, NLP methods may also miss extra-linguistic information such as gesture or tone of voice available in interviews, which may be important for understanding some user perspectives. Moreover, unlike traditional interview respondents, users who contribute to large corpora (e.g. by posting on Reddit) likely could not provide consent to participate in a needfinding study.

---

[7] Thanks to Jed Brubaker and Morgan Klaus Scheuerman for helpful pointers in this area.

[8] During our own work on ClioQuery, we gathered helpful observational data by reading through descriptions of search behavior from social science papers discovered via the JSTOR search engine, a language technology. We consider this to be a form of NLP-based observational needfinding.

Finally, and perhaps most importantly, traditional ethnography has a well-established code of ethics and well-developed theory on the epistemic limits of direct observation, particularly across cultures (see Blomberg and Burrell [41]). In much the same way that traditional ethnographers outside HCI sometimes grossly misunderstand their subjects, it seems likely that computational tools for large-scale analysis of digital text will sometimes grossly misrepresent user needs and practices. For instance, NLP tools might only infer user needs expressed in dominant dialects [39], and thus unexpectedly ignore feedback from marginalized user groups. Thus while NLP methods do have much to offer, responsible technical development will also require corresponding theory describing the proper interpretation and use of new sources of information, much like the existing body of research surrounding established interview, survey or observational techniques [86].

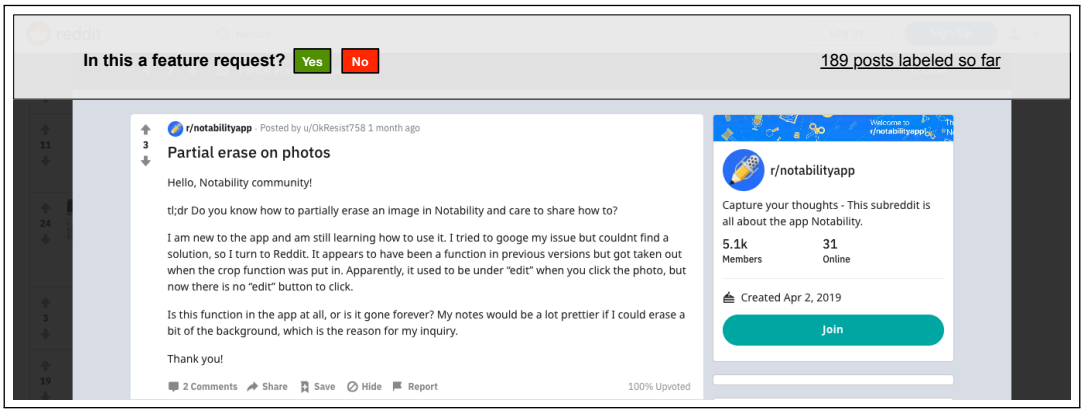


Figure 8.1: A user labels social media posts describing feature requests.

8.3.2 A hypothetical case study in needfinding from text

One possible NLP-based approach to needfinding might focus on tools for human-in-the-loop sensemaking and corpus analysis, much like efforts described throughout this work. For instance, imagine an entrepreneurial software developer looking to create a new note-taking app for tablets. The developer might start off with a suspicion that there are gaps in the market for note-taking software, but might not understand

the exact limitations of existing solutions. To investigate, the developer might begin by performing supervised document-level classification to find a set of posts which describe feature requests (Figure 8.1) on subreddits devoted to existing note taking tools (such as the subreddits `r/Notability`, `r/OneNote` and `r/Evernote` on Reddit.com).

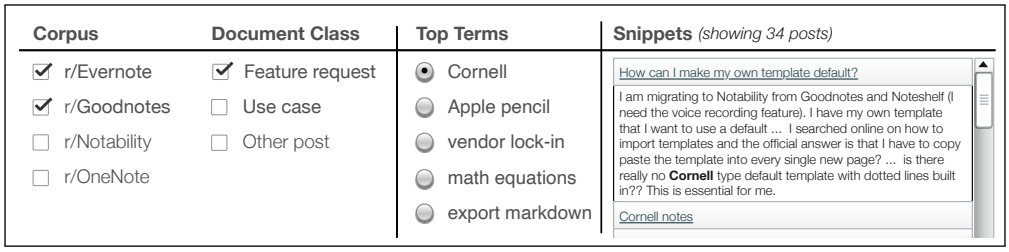


Figure 8.2: A user employs NLP methods to identify design gaps in current note taking applications. After training a supervised (Figure 8.1) 2-way classifier, the user clicks to select documents included in a (predicted) class called "Feature request"(s). On click, the application automatically populates a Top Terms list. The user selects the top term "Cornell," and the snippets populate (right) to show mentions of "Cornell" in context.

After identifying a set of posts which describe feature requests using a supervised classifier, the developer might investigate this new dataset using a lexically-oriented interface much like ROOKIE. For example, the imagined interface in Figure 8.2 ranks the corpus or subcorpus vocabulary according to a word importance score to create a list of top terms (e.g. top terms by word count, from posts describing feature requests). After observing top terms, the developer might discover that feature requests for note taking apps often contain the unigram "Cornell". The user might then click to read snippets from posts that mention "Cornell," and learn that some people use the "Cornell" method to take notes [270, Chp. 10]. They might also learn that some existing note taking apps have poor support for the Cornell method. Equipped with preliminary evidence of a design gap, the developer might then proceed with more traditional user testing, prototyping and market research for an app devoted to Cornell-style note taking. This example shows one way in which NLP might be applied to support needfinding in HCI.

8.4 Final remarks: conclusions from user-facing NLP

This thesis adopts a user-centered view of natural language processing. We develop NLP methods and NLP systems designed for specific user groups, while soliciting user feedback to guide our work. This user-centered vantage point is somewhat unusual in the field. Much work in NLP focuses on developing or investigating language technologies using automatic metrics, while often overlooking how people might actually apply conclusions from NLP research in user-facing settings. While abstracting away from users is sometimes appropriate,[9] our work suggests that the practice of methodically investigating the needs of specific classes of users, and the practice of methodically testing applications built for particular use cases, has much to offer NLP. We detail broad lessons from our work below.

User interfaces shape, constrain and reveal NLP problems

People interact with language technologies via specific user interfaces, which in turn are employed in specific contexts. Throughout our study, we have often found that the details of how such interfaces actually work, and the details of what people actually expect them to do, often shape and constrain natural language processing problems. For instance in Chapter 3, we consider ConceptMap interfaces, which have a particular visual format. Because ConceptMaps require descriptions of relationships for display on a directed graph, our work on ConceptMaps lead us to investigate (and make progress towards) NLP methods for summarizing relationships. In this case, the visual format of the ConceptMap interface helped define and

[9] There are many good reasons to sometimes ignore users in NLP research. Most importantly, automatic evaluation (e.g. F1 scores, Chapter 5) or simplified human evaluation (e.g. Likert judgements, Chapter 3) may help improve language technologies, which might later be deployed in user-facing systems. In such cases, slow and expensive holistic evaluation (e.g. evaluating a finished application with a user study) may by unnecessary. See Eisenstein's related discussion of intrinsic and extrinsic evaluation [100, Sec. 6.4 and 14.6].

shape a natural language processing problem; the interface was not a mere vehicle for applying existing NLP techniques.

Working with users can help refine brittle NLP tasks

Based on our experiences, we believe that studying user-facing systems often reveals new NLP problems, because NLP tasks[10] often encode assumptions which are poorly matched to particular user-facing settings. For example, while working on ROOKIE (Chapter 6), we found that summary quality was far less important than system speed. ROOKIE is effective in part because it can quickly update summaries in response to mouse gestures. Yet work on the traditional summarization task typically ignores system latency, and instead evaluates with the recall-based ROUGE metric against "gold" references [82]. Therefore, progress on summarization (as measured by ROUGE), might not help ROOKIE users in their work answering questions from news archives.

Our experiences with ROOKIE were consistent with other findings from our work. In Chapter 5, we found that standard approaches to the sentence compression task could not be applied in interfaces which accept user queries, and are poorly suited to settings (e.g. journalism) where practitioners do not have access to specialized hardware like GPUs. Likewise, in our work on CONCEPTMAPS, we found that existing research on the relation extraction task was poorly suited to helping people investigate connections between concepts in a corpus. These experiences suggest that testing language technologies with actual users can refine and sharpen NLP research, by revealing hidden assumptions in brittle NLP tasks.

---

[10]We say that a *task* is defined by a set of corpora, a set of shared human or automatic evaluation metrics, and a set of assumptions about possible solutions. For instance, the extractive sentence compression task (Chapter 5) is devoted to shortening sentences by removing words, using statistical and neural NLP techniques. NLP researchers who study extractive compression have agreed to measure possible approaches using token-level F1 scores, as well as human readability and importance judgments. They train and evaluate their work using a benchmark corpus [107].

Ignoring users risks developing language technologies no one wants

Finally, in much the same way that developers should undertake user testing to mitigate the risk of building software no one wants [112], we propose that NLP researchers should investigate user needs to reduce the risk of developing NLP methods that no one will use.[11] This is most clear to us from our work on ClioQuery (Chapter 7), where testing summarization methods with historians revealed problematic assumptions underlying extensive research on summarization techniques (see Section 7.8). Understanding and investigating what at least some group of people want from a particular language technology might help mitigate the risk of conducting time-consuming research based on unreasonable assumptions that will hinder adoption.

Final remarks

This chapter details future work towards lexical corpus analysis, and sketches broader lessons from our study of user-facing natural language processing. In total, we hope that our current and planned work can serve as a case study in user-facing NLP, and enable powerful new tools for user-facing corpus analysis.

[11] However, in some cases, researching user needs may not be necessary. For example, it seems very likely that some users would want real-time machine translation, with the fluency of faithfulness of expert human translators. It seems safe to strive for such technology, without first conducting a needfinding study.

BIBLIOGRAPHY

[1] Content explorer: Discover and analyze top-performing content in your niche. `https://web.archive.org/web/20210104144506/https://ahrefs.com/content-explorer`. Accessed: 2020-01-10.

[2] Wordseer 3.0 in detail. `https://web.archive.org/web/20200812143155/https://wordseer.berkeley.edu/wordseer-3-0/`. Accessed: 2020-11-24.

[3] S. Abney. *Part-of-Speech Tagging and Partial Parsing*, pages 118–136. Springer Netherlands, Dordrecht, 1997.

[4] Jacqueline Aguilar, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis. A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet Annotation Standards. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, 2014.

[5] Enrique Alfonseca, Daniele Pighin, and Guillermo Garrido. Heady: News headline abstraction through event pattern clustering. In *ACL*, 2013.

[6] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of the Broadcast News Understanding and Transcription Workshop*, 1998.

[7] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFST: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer, 2007.

[8] Robert B. Allen and Robert Sieczkiewicz. How historians use historical newspapers. In *Proc. ASIS & T*. American Society for Information Science, 2010.

[9] Miguel Almeida and Andre Martins. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *ACL*, 2013.

[10] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for Human-AI Interaction. In *CHI*, 2019.

[11] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *ACL*, 2016.

[12] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *ACL*, 2015.

[13] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

[14] Markus Bader and Jana Häussler. Toward a model of grammaticality judgments. *Journal of Linguistics*, 46(2):273–330, 2010.

[15] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *ACL*, 1998.

[16] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *ACL*, 1998.

[17] Paul Baker. *Using corpora in discourse analysis*. Continuum, New York, 2006.

[18] Timothy Baldwin and Su Nam Kim. Multiword expressions. In *Handbook of Natural Language Processing*, 2010.

[19] Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, 2015.

[20] Krisztian Balog. *Entity-oriented search*. Springer, Cham,Switzerland, 2018.

[21] Joshua Bambrick, Minjie Xu, Andy Almonte, Igor Malioutov, Guim Perarnau, Vittorio Selo, and Iat Chong Chan. NSTM: Real-time query-driven news overview composition at Bloomberg. In *ACL: System Demonstrations*, 2020.

[22] David Bamman. Natural language processing for the long tail. *Digital Humanities*, 2017.

[23] David Bamman and Noah A. Smith. Unsupervised discovery of biographical structure from text. *TACL*, 2014.

[24] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 2013.

[25] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew J Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[26] Cory Barr, Rosie Jones, and Moira Regelson. The linguistic structure of english web-search queries. In *EMNLP*, 2008.

[27] Regina Barzilay and Kathleen R. McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3), 2005.

[28] Eric P. S. Baumer, David Mimno, Shion Guha, Emily Quan, and Geri K. Gay. Comparing grounded theory and topic modeling: Extreme divergence or unlikely convergence? *Journal of the Association for Information Science and Technology*, 68(6):1397–1410, 2017.

[29] Kenneth R. Beesley and Lauri Karttunen. *Finite-state morphology: Xerox tools and techniques*. CSLI,Stanford, 2003.

[30] Nicolas Garcia Belmonte. Extracting and visualizing insights from real-time conversations around public presentations. In *IEEE VAST*, 2014.

[31] Ram Ben-Shalom. Me'ir nativ: The first hebrew concordance of the bible and Jewish bible study in the fifteenth century,in the context of Jewish-Christian polemics. *Aleph*, 11(2):289–364, 2011.

[32] Emily M. Bender. Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax. *Synthesis Lectures on Human Language Technologies*, 6(3), 2013. Publisher: Morgan & Claypool Publishers.

[33] Kenneth Benoit and Paul Nulty. More than unigrams can say: Detecting meaningful multi-word expressions in political text. *MPSA Working Paper*, 2015.

[34] Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. An empirical investigation of statistical significance in NLP. In *EMNLP*, 2012.

[35] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *JMLR*, 13:281–305, 2012.

[36] Adrian Bingham. The digitization of newspaper archives: Opportunities and challenges for historians. *Twentieth Century British History*, 21(2):225 – 231, 2010.

[37] William R. Black. How watermelons became black: Emancipation and the origins of a racist trope. *Journal of the Civil War Era*, 8(1):64–86, 2018.

[38] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[39] Su Lin Blodgett and Brendan T. O'Connor. Racial disparity in natural language processing: A case study of social media african-american english. *ArXiv*, 2017. URL `https://arxiv.org/abs/1707.00061`.

[40] Su Lin Blodgett, Lisa Green, and Brendan T. O'Connor. Demographic dialectal variation in social media: A case study of african-american english. In *EMNLP*, 2016.

[41] Jeanette Blomberg and Mark Burrell. An ethnographic approach to design. In Julie A. Jacko, editor, *The Human–Computer Interaction Handbook*. CRC Press, New York, 2012.

[42] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[43] M. Brehmer, S. Ingram, J. Stray, and T. Munzner. Overview: The design, adoption, and analysis of a visual document mining tool for investigative journalists. *IEEE TVCG*, 20(12), 2014.

[44] David J. Brenes, Daniel Gayo-Avello, and Kilian Pérez-González. Survey and evaluation of query intent detection methods. In *Proceedings of the 2009 Workshop on Web Search Click Data*, 2009.

[45] David R. Brillinger. Exploratory data analysis. In Bertrand Badie, Dirk Berg-Schlosser, and Leonardo Morlino, editors, *International Encyclopedia of Political Science*. SAGE, Thousand Oaks, April 2021.

[46] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, 1998.

[47] Ted Briscoe, John Carroll, and Rebecca Watson. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 2006.

[48] Erika Bsumek, Matthew O'Hair, Ian Diaz, and Braeden Kennedy. Cliovis, Accessed Dec 24, 2020. URL `https://web.archive.org/web/20201224213752/https://cliovis.org/`.

[49] Andreas Buja, John Alan McDonald, John Michalak, and Werner Stuetzle. Interactive data visualization using focusing and linking. In *Proceedings of the 2nd conference on Visualization*. IEEE, 1991.

[50] Carrie J. Cai, Emily Reif, Narayan Hegde, Jason Hipp, Been Kim, Daniel Smilkov, Martin Wattenberg, Fernanda Viegas, Greg S. Corrado, Martin C. Stumpe, and Michael Terry. Human-centered tools for coping with imperfect algorithms during medical decision-making. In *CHI*, 2019.

[51] Donald Owen Case. The collection and use of information by some american historians: A study of motives and methods. *The Library Quarterly*, 61(1): 61–82, 1991.

[52] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. Evaluation of text generation: A survey. *arXiv*, 2020. URL `https://arxiv.org/abs/2006.14799`.

[53] Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative event chains. In *ACL*, 2008.

[54] Allison Chaney, Hanna Wallach, Matthew Connelly, and David Blei. Detecting and characterizing events. In *EMNLP*, 2016.

[55] Alexandra Chassanoff. Historians and the use of primary source materials in the digital age. *The American Archivist*, 76(2):458–480, 2013.

[56] Nick Chater and Christopher D. Manning. Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, 10(7):335–344, 2006. Special issue: Probabilistic models of cognition.

[57] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, October 2014.

[58] Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/Daily mail reading comprehension task. In *ACL*, 2016.

[59] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *ACL*, 1996.

[60] Daniel Chomsky and Scott Barclay. The mass media, public opinion, and lesbian and gay rights. *Annual Review of Law and Social Science*, 6(1):387–403, 2010.

[61] Noam Chomsky. *Aspects of the theory of syntax*. M.I.T. Press, Cambridge, 1965.

[62] Jason Chuang, Christopher D. Manning, and Jeffrey Heer. "Without the clutter of unimportant words": Descriptive keyphrases for text visualization. *ACM Trans. on Computer-Human Interaction*, 19, 2012.

[63] Jason Chuang, Christopher D. Manning, and Jeffrey Heer. Termite: Visualization techniques for assessing textual topic models. In *AVI*, 2012.

[64] Jason Chuang, Daniel Ramage, Christopher D. Manning, and Jeffrey Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *CHI*, 2012.

[65] Jason Chuang, Sonal Gupta, Christopher D. Manning, and Jeffrey Heer. Topic model diagnostics: Assessing domain relevance via topical alignment. In *ICML*, 2013.

[66] Charles L. A. Clarke, Eugene Agichtein, Susan Dumais, and Ryen W. White. The influence of caption features on clickthrough patterns in web search. In *SIGIR*, 2007.

[67] James Clarke and Mirella Lapata. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429, 2008.

[68] Trevor Cohn and Mirella Lapata. Sentence compression beyond word deletion. In *COLING*, 2008.

[69] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, 2017.

[70] Chrome Contributors. Search the web on chrome, 2021. URL `https://web.archive.org/web/20201105231921/https://support.google.com/chrome/answer/95440`.

[71] Internet Archive Contributors. Internet archive, 200x. Accessed Dec 8, 2020.

[72] "Lexis+ Contributors". Lexis+, 201x. URL `https://www.lexisnexis.com/en-us/products/lexis-plus.page`. Accessed Dec 16,2020.

[73] Lucene contributors. Apache lucenetm 7.4.0 documentation, 2020. URL `https://lucene.apache.org/core/7_4_0/index.html`. Accessed Jun 15, 2020.

[74] Omeka Contributors. Omeka, 200x. Accessed Dec 15, 2020.

[75] Janet Cotterill. Domestic discord, rocky relationships: Semantic prosodies in representations of marital violence in the O.J. Simpson trial. *Discourse & Society*, 12(3):291–312, 2001.

[76] John Creswell. *A concise introduction to mixed methods research*. SAGE, Thousand Oaks,California, 2015.

[77] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*. Pearson Education, 2015.

[78] Alexander Cruden. *A complete concordance to the Holy Scriptures of the Old and New Testament*. Printed for W. Owen, London, 1794. URL `https://catalog.hathitrust.org/Record/012449047`.

[79] Weiwei Cui, Shixia Liu, Li Tan, Conglei Shi, Yangqiu Song, Zekai Gao, Huamin Qu, and Xin Tong. Textflow: Towards better understanding of evolving topics in text. *IEEE TVCG*, 17(12), 2011.

[80] Weiwei Cui, Shixia Liu, Zhuofeng Wu, and Hao Wei. How hierarchical topics evolve in large text corpora. *IEEE TVCG*, 20(12), 2014.

[81] Margaret Stieg Dalton and Laurie Charnigo. Historians and their information sources. *College & Research Libraries*, 65(5), 2004.

[82] Dipanjan Das and André FT Martins. A survey on automatic text summarization. Technical report, Carnegie Mellon University, 2007.

[83] Hal Daumé III and Daniel Marcu. Bayesian query-focused summarization. In *ACL*, 2006.

[84] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC*, 2006.

[85] Luciano Del Corro and Rainer Gemulla. Clausie: clause-based open information extraction. In *WWW*, 2013.

[86] Norman K. Denzin and Yvonna S. Lincoln, editors. *The Sage handbook of qualitative research*. Sage, 2018.

[87] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[88] N. Diakopoulos, M. Naaman, and F. Kivran-Swaine. Diamonds in the rough: Social media visual analytics for journalistic inquiry. In *IEEE VAST*, 2010.

[89] Anthony Don, Elena Zheleva, Machon Gregory, Sureyya Tarkan, Loretta Auvil, Tanya Clement, Ben Shneiderman, and Catherine Plaisant. Discovering interesting usage patterns in text collections: Integrating text mining with visualization. In *CIKM*, 2007.

[90] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017. URL `https://arxiv.org/abs/1702.08608`.

[91] Aron Dotan. Masorah. In Michael Berenbaum and Fred Skolnik, editors, *Encyclopaedia Judaica*, volume 13, pages 603–656. Macmillan Reference USA, Detroit,MI, 2nd edition, 2007.

[92] W. Dou, X. Wang, R. Chang, and W. Ribarsky. Paralleltopics: A probabilistic approach to exploring document collections. In *IEEE VAST*, 2011.

[93] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul):2121–2159, 2011.

[94] Wendy Duff, Barbara Craig, and Joan Cherry. Historians' use of archival sources: Promises and pitfalls of the digital age. *The Public Historian*, 26 (2):7–22, 2004.

[95] Wendy M. Duff and Catherine A. Johnson. Accidentally found on purpose: Information-seeking behavior of historians in archives. *The Library Quarterly*, 72(4):472–496, 2002.

[96] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–74, 1993.

[97] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference,typing,and linking. *TACL*, 2014.

[98] M. Dörk, S. Carpendale, C. Collins, and C. Williamson. Visgets: Coordinated visualizations for web-based information exploration and discovery. *IEEE TVCG*, 14(6), 2008.

[99] Maud Ehrmann, Estelle Bunout, and Marten Düring. Historical newspaper user interfaces: A review. In *International Federation of Library Associations World Library Information Conference*, 2019.

[100] Jacob Eisenstein. *Introduction to Natural Language Processing.* MIT Press, Cambridge, MA, 2019.

[101] Matan Eyal, Tal Baumel, and Michael Elhadad. Question answering as an automatic evaluation metric for news article summarization. In *NAACL*, 2019.

[102] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[103] Tobias Falke and Iryna Gurevych. Bringing structure into summaries: Crowdsourcing a benchmark corpus of concept maps. In *EMNLP*, 2017.

[104] Tobias Falke and Iryna Gurevych. Utilizing automatic predicate-argument analysis for concept map mining. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*, 2017.

[105] Henry Feild, Ryen W. White, and Xin Fu. Supporting orientation during search result examination. In *CHI*, 2013.

[106] Katja Filippova and Enrique Alfonseca. Fast k-best sentence compression. *CoRR*, abs/1510.08418, 2015.

[107] Katja Filippova and Yasemin Altun. Overcoming the lack of parallel data in sentence compression. In *EMNLP*, 2013.

[108] Katja Filippova and Michael Strube. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, 2008.

[109] Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with LSTMs. In *EMNLP*, 2015.

[110] Norbert Finzsch. CONCORD A Program for Concordance Analyses on a Personal Computer. *Historical Social Research / Historische Sozialforschung*, 14 (4 (52)):156–158, 1989. Publisher: GESIS - Leibniz-Institute for the Social Sciences,Center for Historical Social Research.

[111] John Rupert Firth. *A Synopsis of Linguistic Theory,1930-1955*, chapter Special volume of the Philological Society,chapter 1, pages 1 – 32. Blackwell, 1957.

[112] Rob Fitzpatrick. *The mom test: how to talk to customers and learn if your business is a good idea when everyone is lying to you.* CreateSpace Publishing, 2013.

[113] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[114] Eric Foner. *Free soil, free labor, free men: The ideology of the Republican Party before the Civil War.* Oxford University Press, New York, 1995.

[115] Matthew Francis-Landau, Greg Durrett, and Dan Klein. Capturing semantic similarity for entity linking with convolutional neural networks. In *NAACL*, 2016.

[116] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. Automatic recognition of multi-word terms: The c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130, 2000.

[117] Susanne Gahl, Dan Jurafsky, and Douglas Roland. Verb subcategorization frequencies: American english corpus data, methodological studies, and cross-corpus comparisons. *Behavior Research Methods, Instruments, & Computers*, 36(3):432–443, 2004.

[118] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *NAACL*, 2013.

[119] Alexandra García-Marrugo. What's in a name? The representation of illegal actors in the internal conflict in the Colombian press. *Discourse & Society*, 24 (4):421–445, 2013. Publisher: Sage Publications,Ltd.

[120] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, 2018.

[121] William Gaver. Science and design: The implications of different forms of accountability. In Judith S. Olson and Wendy A. Kellogg, editors, *Ways of Knowing in HCI.* Springer, New York, NY, 2014.

[122] Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. Bottom-up abstractive summarization. In *EMNLP*, 2018.

[123] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *ACL*, 2011.

[124] Lisa Given and Kristie Saumure. Convenience sample. In Lisa Given, editor, *The SAGE Encyclopedia of Qualitative Research Methods*. SAGE Publications, Inc., Thousand Oaks, California, 2008.

[125] Natalie Glance, Matthew Hurst, Kamal Nigam, Matthew Siegler, Robert Stockton, and Takashi Tomokiyo. Deriving marketing intelligence from online discussion. In *KDD*, 2005.

[126] Carsten Görg, Zhicheng Liu, and John Stasko. Reflections on the evolution of the Jigsaw visual analytics system. *Information Visualization*, 2013.

[127] John D. Gould and Clayton Lewis. Designing for usability: Key principles and what designers think. *Commun. ACM*, 28(3):300–311, 1985.

[128] Shawn Graham, Scott Weingart, and Ian Milligan. Getting started with topic modeling and mallet. `https://programminghistorian.org/en/lessons/topic-modeling-and-mallet`, 2012. Accessed: 2021-06-02.

[129] Lisa Green. *African American English: A linguistic introduction*. Cambridge University Press, Cambridge,U.K. New York, 2002.

[130] Spence Green, Jeffrey Heer, and Christopher D. Manning. Natural language translation at the intersection of AI and HCI: Old questions being answered with both AI and HCI. *Queue*, 13(6):30–42, 2015.

[131] Abbie Griffin and John R. Hauser. The voice of the customer. *Marketing science*, 12(1):1–27, 1993.

[132] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL*, 2018.

[133] LLC Gurobi Optimization. Gurobi optimizer reference manual (v8), 2018.

[134] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *NAACL*, 2018.

[135] Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization. In *NAACL*, 2009.

[136] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. In *ACL*, 2016.

[137] Bassam Hammo, Sane Yagi, Omaima Ismail, and Mohammad AbuShariah. Exploring and exploiting a historical corpus for arabic. *Language Resources and Evaluation*, 50(4):839–861, 2016.

[138] Abram Handler and Brendan O'Connor. Rookie: A unique approach for exploring news archives. In *Workshop on Data Science + Journalism at KDD*, 2017.

[139] Abram Handler and Brendan O'Connor. Relational summarization for corpus analysis. In *NAACL*, 2018.

[140] Abram Handler and Brendan O'Connor. Query-focused sentence compression in linear time. In *EMNLP*, 2019.

[141] Abram Handler, Su Lin Blodgett, and Brendan T. O'Connor. Visualizing textual models with in-text and word-as-pixel highlighting. *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[142] Abram Handler, Matthew Denny, Hanna Wallach, and Brendan O'Connor. Bag of what? Simple noun phrase extraction for text analysis. In *Proceedings of the First Workshop on NLP and Computational Social Science*, 2016.

[143] Abram Handler, Brian Dillon, and Brendan T. O'Connor. Human acceptability judgements for extractive sentence compression. *ArXiv*, 2019. URL `https://arxiv.org/pdf/1902.00489`.

[144] Abram Handler, Premkumar Ganeshkumar, Brendan O'Connor, and Mohamed AlTantawy. Summarizing relationships for interactive concept map browsers. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, 2019.

[145] Lauren A. Hannah and Hanna Wallach. Summarizing topics: From word lists to phrases. In *NIPS Workshop on Modern Machine Learning and Natural Language Processing*, 2014.

[146] Zellig S. Harris. Distributional Structure. *WORD*, 10(2-3):146–162, 1954.

[147] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE TVCG*, 8(1):9–20, Jan 2002.

[148] Kenneth Heafield. KenLM: Faster and smaller language model queries. In *EMNLP: Sixth Workshop on Statistical Machine Translation*, 2011.

[149] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, 1992.

[150] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *CHI*, 1995.

[151] Marti A. Hearst. *Search user interfaces*. Cambridge University Press, 2009.

[152] Jeffrey Heer. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences*, 116(6), 2019.

[153] Jeffrey Heer and Ben Shneiderman. Interactive dynamics for visual analysis. *Commun. ACM*, 55(4):45–54, 2012.

[154] Michael Heilman, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault. Predicting grammaticality on an ordinal scale. In *ACL*, 2014.

[155] Edward S. Herman and Noam Chomsky. *Manufacturing consent : the political economy of the mass media*. Pantheon Books, New York, NY, 1988.

[156] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

[157] David Leon Higdon. The concordance: Mere index or needful census? *Text*, 15:51–68, 2003.

[158] Tim Hitchcock, Robert Shoemaker, Clive Emsley, Sharon Howard, and Jamie McLaughlin. The Old Bailey proceedings online, 1674-1913, 2012. URL `https://www.oldbaileyonline.org`.

[159] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[160] O. Hoeber and Xue Dong Yang. The visual exploration of web search results using hotmap. In *Tenth International Conference on Information Visualisation (IV'06)*, 2006.

[161] Jennifer Hoewe and Geri Alumit Zeldes. Overturning anti-miscegenation laws: News media coverage of the lovings' legal case against the state of virginia. *Journal of Black Studies*, 43(4):427–443, 2012.

[162] Enamul Hoque and Giuseppe Carenini. Interactive topic modeling for exploring asynchronous online conversations: Design and evaluation of ConVisIT. *TiiS*, 6(1), 2016.

[163] Eric Horvitz. Principles of mixed-initiative user interfaces. In *CHI*, 1999.

[164] Eduard Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. Events are not simple: Identity,non-identity,and quasi-identity. In *Workshop on Events: Definition, Detection, Coreference, and Representation*, 2013.

[165] Yuening Hu, Jordan Boyd-Graber, and Brianna Satinoff. Interactive topic modeling. In *ACL*, 2011.

[166] Rodney Huddleston and Geoffrey K. Pullum. *The Cambridge Grammar of the English Language*. Cambridge University Press, 2002.

[167] Mans Hulden. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL*, 2009.

[168] Daniel H H Ingalls. Design principles behind smalltalk. *Byte*, 1981. URL `https://archive.org/details/byte-magazine-1981-08/page/n299/mode/1up`.

[169] E. Isaacs, K. Damico, S. Ahern, E. Bart, and M. Singhal. Footprints: A visual search tool that supports discovery and coverage tracking. *IEEE TVCG*, 20 (12):1793–1802, 2014.

[170] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36:207–227, 2000.

[171] Mahmood Jasim, Pooya Khaloo, Somin Wadhwa, Amy X. Zhang, Ali Sarvghad, and Narges Mahyar. Communityclick: Towards improving inclusivity in town halls. In *CSCW*, 2020.

[172] Colman Jerman O.P. Hugh of st. cher. *Dominicana*, 44(4), 1959. URL `https://web.archive.org/web/20200824160615/https://www.dominicanajournal.org/wp-content/files/old-journal-archive/vol44/no4/dominicanav44n4hughstcher.pdf`.

[173] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, 2017.

[174] Jialun Aaron Jiang, Kandrea Wade, Casey Fiesler, and Jed R. Brubaker. Supporting Serendipity: Opportunities and Challenges for Human-AI Collaboration in Qualitative Analysis. *CSCW*, 5(1), 2021.

[175] Hongyan Jing and Kathleen R McKeown. Cut and paste based text summarization. In *NAACL*, 2000.

[176] Hongyan Jing, Regina Barzilay, Kathleen McKeown, and Michael Elhadad. Summarization evaluation methods: Experiments and analysis. In *AAAI Spring Symposium*, 1998.

[177] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv*, 2016. URL `https://arxiv.org/pdf/1602.02410.pdf`.

[178] John S. Justeson and Slava M. Katz. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(01):9–27, 1995.

[179] Katharina Kann, Sascha Rothe, and Katja Filippova. Sentence-Level Fluency Evaluation: References Help,But Can Be Spared! In *CoNNL*, 2018.

[180] Chris Kedzie, Kathleen McKeown, and Fernando Diaz. Predicting salient updates for disaster summarization. In *ACL*, 2015.

[181] Katherine Keith, Su Lin Blodgett, and Brendan O'Connor. Monte Carlo syntax marginals for exploring and using dependency parses. In *NAACL*, 2018.

[182] Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. Controlling output length in neural encoder-decoders. In *EMNLP*, 2016.

[183] Adam Kilgarriff and Iztok Kosem. Corpus tools for lexicographers. In Sylviane Granger and Magali Paquot, editors, *Electronic Lexicography*, pages 31–56. Oxford University Press, 2012.

[184] Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. Overview of BioNLP shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, 2011.

[185] Su Nam Kim, Timothy Baldwin, and Min-Yen Kan. Evaluating n-gram based evaluation metrics for automatic keyphrase extraction. In *COLING*, 2010.

[186] Svetlana Kiritchenko and Saif Mohammad. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *ACL*, 2017.

[187] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *CHI*, 2008.

[188] Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. Improving sentence compression by learning to predict gaze. In *NAACL*, 2016.

[189] Kevin Knight and Daniel Marcu. Statistics-based summarization - step one: Sentence compression. In *AAAI*, 2000.

[190] Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *NAACL*, 2003.

[191] Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Neural text summarization: A critical evaluation. In *EMNLP*, 2019.

[192] Steven Langsford, Amy Perfors, Andrew T Hendrickson, Lauren A Kennedy, and Danielle J Navarro. Quantifying sentence acceptability measures: Reliability, bias, and variability. *Glossa: A journal of general linguistics*, 3(1), 2018.

[193] Richard K Larson. On the syntax of disjunction scope. *Natural Language & Linguistic Theory*, 3(2):217–264, 1985.

[194] Jey Han Lau, Alexander Clark, and Shalom Lappin. Unsupervised prediction of acceptability judgements. In *ACL*, 2015.

[195] Jey Han Lau, Alexander Clark, and Shalom Lappin. Grammaticality,acceptability,and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, 41(5):1202–1241, 2017.

[196] Anthony Laurence. Antconc: A learner and classroom friendly, multi-platform corpus analysis toolkit. *Proceedings of IWLeL: An Interactive Workshop on Language E-learning 2004*, 2004.

[197] LexisNexis Legal. Key words in context view on lexis, 2020. URL `https://www.youtube.com/watch?v=-DXXRSpAZBA`.

[198] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, 2014.

[199] Mark Liberman. "A briefe and a compendious table", 2004. URL `https://web.archive.org/web/20201222233836/http://itre.cis.upenn.edu/~myl/languagelog/archives/000537.html`. "Accessed December 22,2020".

[200] Chin-Yew Lin and Eduard Hovy. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, 2000.

[201] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *NAACL*, 2010.

[202] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *ACL*, 2011.

[203] Shixia Liu, Michelle X. Zhou, Shimei Pan, Yangqiu Song, Weihong Qian, Weijia Cai, and Xiaoxiao Lian. Tiara: Interactive, topic-based visual text summarization and analysis. *ACM TIST.*, 3(2), 2012.

[204] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*, 2019. URL `https://arxiv.org/pdf/1907.11692.pdf`.

[205] Zhicheng Liu and Jeffrey Heer. The effects of interactive latency on exploratory visual analysis. *IEEE TVCG*, 2014.

[206] J.J. Louviere. Best-worst scaling: A model for the largest difference judgments. Technical report, University of Alberta, 1991.

[207] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[208] H. P. Luhn. Key word-in-context index for technical literature (kwic index). *American Documentation*, 11(4):288–295, 1960.

[209] Jack Lule. Myth and terror on the editorial page: The New York Times responds to september 11, 2001. *Journalism & Mass Communication Quarterly*, 79(2):275–293, 2002.

[210] Bill MacCartney and Christopher D Manning. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*, 2009.

[211] Neil A Macmillan and C Douglas Creelman. Response bias: Characteristics of detection theory, threshold theory, and "nonparametric" indexes. *Psychological Bulletin*, 107(3):401, 1990.

[212] Neil A. Macmillan and C. Douglas Creelman. *Detection theory: A user's guide*. Psychology Press, 2004.

[213] Trent MacNamara. Why "race suicide"? Cultural factors in U.S. fertility decline, 1903–1908. *The Journal of Interdisciplinary History*, 44(4):475–508, 2014.

[214] Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. Sentence Compression for Arbitrary Languages via Multilingual Pivoting. In *EMNLP*, 2018.

[215] Christopher Manning and Carson Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge,Mass, 1999.

[216] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[217] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL: System Demonstrations*, 2014.

[218] Megh Marathe and Kentaro Toyama. Semi-automated coding for qualitative research: A user-centered inquiry and initial prototypes. In *CHI*, 2018.

[219] Gary Marchionini. Exploratory search: From finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.

[220] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. Twitinfo: Aggregating and visualizing microblogs for event exploration. In *CHI*, 2011.

[221] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[222] Eduard Mark. In Re Alger Hiss: A final verdict from the archives of the KGB. *Journal of Cold War Studies*, 11(3):26–67, 2009.

[223] Howard Markel, Harvey B. Lipman, J. Alexander Navarro, Alexandra Sloan, Joseph R. Michalsen, Alexandra Minna Stern, and Martin S. Cetron. Nonpharmaceutical Interventions Implemented by U.S. Cities During the 1918-1919 Influenza Pandemic. *JAMA*, 298(6):644–654, 2007.

[224] André FT Martins and Noah A Smith. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, 2009.

[225] Michael Matthews, Pancho Tolchinsky, Roi Blanco, Jordi Atserias, Peter Mika, and Hugo Zaragoza. Searching through time in the New York Times. In *HCIR*, 2010.

[226] Roger C. Mayer, James H. Davis, and F. David Schoorman. An integrative model of organizational trust. *The Academy of Management Review*, 20(3): 709–734, 1995.

[227] Thomas R. McCoy and Tal Linzen. Non-entailed subsequences as a challenge for natural language inference. In *SCIL*, 2019.

[228] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice. *CSCW*, 2019.

[229] Ryan McDonald. A study of global inference algorithms in multi-document summarization. In *ECIR*, 2007.

[230] Tony McEnery and Costas Gabrielatos. English Corpus Linguistics. In *The Handbook of English Linguistics*. John Wiley & Sons, Ltd, 2006.

[231] Tony McEnery and Andrew Hardie. *Corpus Linguistics: Method,Theory and Practice*. Cambridge Textbooks in Linguistics. Cambridge University Press, 2011.

[232] Kathleen McKeown and Dragomir R. Radev. Generating summaries of multiple news articles. In *SIGIR*, 1995.

[233] Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. Tracking and summarizing news on a daily basis with Columbia's newsblaster. In *HLT*, 2002.

[234] Michael Meyer, Renate Buber, and Anahid Aghamanoukjan. In search of legitimacy: Managerialism and legitimation in civil society organizations. *Voluntas: International Journal of Voluntary and Nonprofit Organizations*, 24(1):167–193, 2013.

[235] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Holberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 2010.

[236] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL `http://arxiv.org/abs/1301.3781`.

[237] Matthew B. Miles, A. M. Huberman, and Johnny Saldaña. *Qualitative data analysis: A methods sourcebook.* SAGE, 3 edition, 2014.

[238] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38 (11):39–41, 1995.

[239] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL*, 2009.

[240] John W. Mohr and Petko Bogdanov. Topic models: What they are and why they matter. *Poetics*, 41:545–569, 2013.

[241] Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403, 2008.

[242] Aditi Muralidharan and Marti Hearst. Wordseer: Exploring language use in literary text. In *Fifth Workshop on Human-Computer Interaction and Information Retrieval*, 2011.

[243] Aditi Muralidharan, Marti A. Hearst, and Christopher Fan. WordSeer: A Knowledge Synthesis Environment for Textual Data. In *CIKM*, 2013.

[244] Dhiraj Murthy. Digital ethnography: An examination of the use of new technologies for social research. *Sociology*, 42(5):837–855, 2008.

[245] Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, 2011.

[246] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2), 2009.

[247] Ani Nenkova and Kathleen McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3):103–233, 2011.

[248] Ani Nenkova and Kathleen McKeown. A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer, 2012.

[249] Ani Nenkova and Lucy Vanderwende. The impact of frequency on summarization. Technical report, Microsoft Research, 2005.

[250] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *ArXiv*, 2017. URL `https://arxiv.org/abs/1703.01619`.

[251] New York Times contributors. The *New York Times* search page, 2020. URL `https://www.nytimes.com/search`. Accessed June 12, 2020.

[252] Roger Newson. Parameters behind "nonparametric" statistics: Kendall's tau, Somers' D and median differences. *The Stata Journal*, 2(1):45–64, 2002.

[253] Newspaper.com contributors. newspapers.com, 2020. URL `https://www.newspapers.com`. [Online; accessed 15-June-2020].

[254] Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. Summarizing sporting events using twitter. In *IUI*, 2012.

[255] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco,CA,USA, 1993.

[256] Jakob Nielsen. 10 usability heuristics for user interface design. https://web.archive.org/web/20210106204738/https://www.nngroup.com/articles/ten-usability-heuristics/, 1994. Accessed: 2021-01-06.

[257] Janni Nielsen, Torkil Clemmensen, and Carsten Yssing. Getting access to what goes on in people's heads? Reflections on the think-aloud technique. In *NordiCHI*, 2002.

[258] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *International Conference on Parsing Technologies*, 2003.

[259] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In *LREC*, 2016.

[260] Seyednaser Nourashrafeddin, Ehsan Sherkat, Rosane Minghim, and Evangelos E. Milios. A visual approach for interactive keyterm-based clustering. *TiiS*, 8(1), 2018.

[261] Michael P. Oakes. Concordancing, collocations and dictionaries. In Tony McEnery and Andrew Wilson, editors, *Statistics for Corpus Linguistics*, pages 149–198. Edinburgh University Press, 1998.

[262] Brendan O'Connor. MiTextExplorer: Linked brushing and mutual information for exploratory text data analysis. In *Proceedings of the Workshop on Interactive Language Learning,Visualization,and Interfaces*, 2014.

[263] Adam Oliner, Archana Ganapathi, and Wei Xu. Advances and challenges in log analysis. *Commun. ACM*, 55(2), 2012.

[264] Bryan Orme. Maxdiff analysis: Simple counting, individual-level logit, and hb. Technical report, Sawtooth Software, 2009.

[265] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *NAACL*, 2013.

[266] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English Gigaword fifth edition, 2011.

[267] S. M. Parrish and James Allan Painter. *Editor's Preface*, pages v–xxviii. Cornell University Press, 1963.

[268] Ramakanth Pasunuru, Asli Celikyilmaz, Michel Galley, Chenyan Xiong, Yizhe Zhang, Mohit Bansal, and Jianfeng Gao. Data augmentation for abstractive query-focused multi-document summarization. In *AAAI*, 2021.

[269] Dev Patnaik and Robert Becker. Needfinding: The why and how of uncovering people's needs. *Design Management Journal*, 10(2):37–43, 1999.

[270] Walter Pauk and Ross J. Q. Owens. *How to study in college*. Wadsworth, Boston, MA, 10th ed edition, 2011.

[271] Ellie Pavlick and Chris Callison-Burch. So-called non-subsective adjectives. In **SEM*, 2016.

[272] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830, 2011.

[273] Fabian Pedregosa-Izquierdo. *Feature extraction and supervised learning on fMRI: From practice to theory*. PhD thesis, Université Pierre et Marie Curie - Paris VI, 2015.

[274] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *LREC*, 2012.

[275] Peter R. Petrucci and Michael Head. Hurricane katrina's lexical storm: The use of "refugee as" a label for american citizens. *Australasian Journal of American Studies*, 25(2):23–39, 2006.

[276] Paul Pierson. *Politics in time: History, institutions, and social analysis*. Princeton University Press, Princeton, New Jersey, 2004.

[277] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, volume 5, 2005.

[278] Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *NAACL*, 2018.

[279] Proquest LLC. *Proquest*, 2020. URL `https://search.proquest.com/`. Accessed June 12, 2020.

[280] George Psathas. Problems and Prospects in the Use of a Computer System of Content Analysis. *The Sociological Quarterly*, 7(4):449–468, 1966. Publisher: Midwest Sociological Society.

[281] James Pustejovsky. The syntax of event structure. *Cognition*, 41(1):47–81, 1991.

[282] Lara Putnam. The Transnational and the Text-Searchable: Digitized Sources and the Shadows They Cast. *The American Historical Review*, 121(2):377–402, 2016.

[283] William Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois, 2020. R package version 2.0.9.

[284] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *NAACL*, 2013.

[285] Emmanuel Roche and Yves Schabes. *Finite-State Language Processing*. MIT Press, 1997.

[286] Geoffrey Rockwell, Stéfan G. Sinclair, Stan Ruecker, and Peter Organisciak. Ubiquitous text analysis. *paj: The Journal of the Initiative for Digital Humanities, Media, and Culture*, 2(1), 2010.

[287] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A Primer in BERTology: What We Know About How BERT Works. *TACL*, 8, 2021.

[288] Stephen Roller, Douwe Kiela, and Maximilian Nickel. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *ACL*, 2018.

[289] R.H. Rouse and M.A. Rouse. The Verbal Concordance to the Scriptures. *Archivum Fratrum Praedicatorum*, XLIV (44):5–30, 1974. URL `http://catalog.hathitrust.org/Record/000598885`.

[290] Jeffrey Rubin, Dana Chisnell, and Jared Spool. *Handbook of usability testing: how to plan, design and conduct effective tests*. John Wiley & Sons, Indianapolis, Indiana, 2nd edition, 2008.

[291] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

[292] Evan Sandhaus. The New York Times Annotated Corpus. *Linguistic Data Consortium*, LDC2008T19, 2008.

[293] Cory R. Schaffhausen. *Large-Scale Needfinding Methods,Quality Metrics,and Need Prioritization in User-Centered Design.* PhD thesis, 2015.

[294] Benjamin M Schmidt. Words alone: Dismantling topic models in the humanities. *Journal of Digital Humanities*, 2(1):49–65, 2012.

[295] Philip A Schrodt, Deborah J Gerner, Rajaa Abu-Jabr, ömür Yilmaz, and Erin M Simpson. Analyzing the dynamics of international mediation processes in the middle east and balkans. In *Annual Meeting of the American Political Science Association*, 2001.

[296] Sebastian Schuster and Christopher D. Manning. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *LREC*, 2016.

[297] Carson Schütze. *The empirical base of linguistics: Grammaticality judgments and linguistic methodology.* University of Chicago Press, Chicago, Il, 1996.

[298] Carson T Schütze and Jon Sprouse. Judgment data. In Robert J. Podesva and Devyani Sharma, editors, *Research methods in linguistics.* Cambridge University Press, 2014.

[299] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, 2020.

[300] Cecilia Di Sciascio, Vedran Sabol, and Eduardo Veas. Supporting exploratory search with a visual user-driven approach. *TiiS*, 7(4), 2017.

[301] Wolfgang Seber. *Argus Homericus sive Index vocabulorum in omnia Homeri poemata. By Wolfgangus Seberus.* Early European Books: Printed sources to 1700. Prost. apud J. Janssonium, 1651.

[302] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE TVCG*, 18(12), 2012.

[303] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Trains of thought: Generating information maps. In *WWW*, 2012.

[304] Dafna Shahaf, Jaewon Yang, Caroline Suen, Jeff Jacobs, Heidi Wang, and Jure Leskovec. Information cartography: Creating zoomable, large-scale maps of information. In *KDD*, 2013.

[305] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues,techniques,and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2), 2015.

[306] Ehsan Sherkat, Evangelos E. Milios, and Rosane Minghim. A visual analytics approach for interactive document clustering. *TiiS*, 10(1), 2019.

[307] Jeannie N. Shinozuka. Deadly perils: Japanese beetles and the pestilential immigrant, 1920s-1930s. *American Quarterly*, 65(4):831–852, 2013.

[308] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, 1996.

[309] Ben Shneiderman. Claiming success,charting the future: Micro-HCI and macro-HCI. *Interactions*, 18(5):10–11, 2011.

[310] Jaspreet Singh, Wolfgang Nejdl, and Avishek Anand. Expedition: A time-aware exploratory search system designed for scholars. In *SIGIR*, 2016.

[311] Jaspreet Singh, Wolfgang Nejdl, and Avishek Anand. History by diversity: Helping historians search news archives. In *CHIR*, 2016.

[312] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*, 2008.

[313] Janine Solberg. Googling the archive: Digital tools and the practice of history. *Advances in the History of Rhetoric*, 15(1), 2012.

[314] John Sprouse and Carson Schütze. *Research Methods in Linguistics*, chapter Judgment Data. Cambridge University Press, Cambridge, UK, 2014.

[315] Jon Sprouse and Diogo Almeida. Design sensitivity and statistical power in acceptability judgment experiments. *Glossa*, 2(1):1, 2017.

[316] Gabriel Stanovsky and Ido Dagan. Creating a large benchmark for open information extraction. In *EMNLP*, 2016.

[317] John Stasko, Carsten Görg, and Zhicheng Liu. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2): 118–132, 2008.

[318] Adrian Staub, Charles Clifton, and Lyn Frazier. Heavy np shift is the parser's last resort: Evidence from eye movements. *Journal of memory and language*, 54(3):389–406, 2006.

[319] Emilia Stoica, Marti Hearst, and Megan Richardson. Automating creation of hierarchical faceted metadata structures. In *NAACL*, 2007.

[320] Jonathan Stray. What do Journalists do with Documents? In *Computation+Journalism Symposium*, 2016.

[321] Michael Stubbs. Corpus semantics. In Nick Riemer, editor, *The Routledge handbook of semantics*. Routledge, New York, NY, 2015.

[322] Hillel Taub Tabib, Micah Shlain, Shoval Sadde, Dan Lahav, Matan Eyal, Yaara Cohen, and Yoav Goldberg. Interactive extractive search over biomedical corpora. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, 2020.

[323] Jenifer Tidwell. A pattern language for human-computer interface design, 1999. URL `https://web.archive.org/web/20201212151556/http://www.mit.edu/~jtidwell/common_ground.html`. [Online; accessed 16-Jan-2021].

[324] Jenifer Tidwell, Charles Brewer, and Aynne Valencia. *Designing Interfaces*. O'Reilly Media, Inc, 3rd edition, 2020.

[325] Anastasios Tombros and Mark Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR*, 1998.

[326] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*, 2003.

[327] Edward Tufte. *Beautiful Evidence*. Graphics Press, Cheshire, Conn, 2006.

[328] John Tukey. *Exploratory data analysis*. Pearson, 1977.

[329] Daniel Tunkelang. Faceted search. *Synthesis lectures on information concepts, retrieval, and services*, 1(1):1–80, 2009.

[330] Ted Underwood. Theorizing Research Practices We Forgot to Theorize Twenty Years Ago. *Representations*, 127(1):64–72, 2014.

[331] David Vadas and James R. Curran. Parsing noun phrases in the Penn Treebank. *Computational Linguistics*, 37(4), 2011.

[332] Stef van den Elzen and Jarke J. van Wijk. Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum*, 32(3): 191–200, 2013.

[333] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(86):2579–2605, 2008.

[334] Koen van Turnhout, Arthur Bennis, Sabine Craenmehr, Robert Holwerda, Marjolein Jacobs, Ralph Niels, Lambert Zaad, Stijn Hoppenbrouwers, Dick Lenior, and René Bakker. Design patterns for mixed-method research in HCI. In *NordiCHI*, 2014.

[335] Fernanda B. Viégas, Scott Golder, and Judith Donath. Visualizing email content: Portraying relationships from conversational histories. In *CHI*, 2006.

[336] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. Learning to explain entity relationships in knowledge graphs. In *ACL*, 2015.

[337] Nir Vulkan. An economist's perspective on probability matching. *Journal of economic surveys*, 14(1):101–118, 2000.

[338] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, 2019.

[339] Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. Can syntax help? Improving an LSTM-based sentence compression model for new domains. In *ACL*, 2017.

[340] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. In *AVI*, 2000.

[341] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *TACL*, 7:625–641, 2019.

[342] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan and Claypool Publishers, 2009.

[343] Ryen W. White, Gary Marchionini, and Gheorghe Muresan. Evaluating exploratory search systems: Introduction to special topic issue of information processing and management. *Information Processing and Management*, 44(2), 2008. Evaluating Exploratory Search Systems Digital Libraries in the Context of Users' Broader Activities.

[344] Hadley Wickham. The tidy tools manifesto. `https://web.archive.org/web/20190110110838/https://github.com/tidyverse/tidyverse/blob/master/vignettes/manifesto.Rmd`, 2018.

[345] Wikipedia contributors. Clio, 2021. URL `https://en.wikipedia.org/wiki/Clio`. [Online; accessed 20-Jan-2021].

[346] Garry Wills. Untitled Review of Concordances to Homer. *The American Journal of Philology*, 85(4):445–447, 1964. Publisher: Johns Hopkins University Press.

[347] Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *TACL*, 2015.

[348] J. Yang, D. Luo, and Y. Liu. Newdle: Interactive visual exploration of large online news collections. *IEEE Computer Graphics and Applications*, 30(5):32–41, 2010.

[349] David Yanofsky. If you're using an Android phone, Google may be tracking every move you make, 2018. URL `https://web.archive.org/web/20210203232323/https://qz.com/1183559/if-youre-using-an-android-phone-google-may-be-tracking-every-move-you-make/`.

[350] Dani Yogatama, Lingpeng Kong, and Noah A. Smith. Bayesian optimization of text representations. In *EMNLP*, 2015.

[351] Polle T. Zellweger, Bay-Wei Chang, and Jock D. Mackinlay. Fluid links for informed and incremental link transitions. In *Hypertext*, 1998.

[352] Amy X. Zhang and Justin Cranshaw. Making sense of group chat through collaborative tagging and summarization. *CSCW*, 2018.

[353] Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling. In *The Proceedings of the 12th International Conference on Computational Semantics (IWCS)*, 2017.

[354] Elena Zheleva. Vox media dataset. In *KDD DS + J Workshop*, 2017.

[355] Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. A monolingual tree-based translation model for sentence simplification. In *COLING*, 2010.

[356] Jens O. Zinn. The Proliferation of 'at risk' in The Times: A Corpus Approach to Historical Social Change,1785-2009. *Historical Social Research*, 43(2 (164)), 2018. Publisher: GESIS - Leibniz Institute for the Social Sciences.

[357] George K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.