INTERACTIVE VISUAL TEXT ANALYTICS

A Dissertation
Presented to
The Academic Faculty

By

Hannah Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering
College of Computing

Georgia Institute of Technology

December 2020

INTERACTIVE VISUAL TEXT ANALYTICS

Thesis committee:

Dr. Haesun Park
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Chao Zhang
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Alex Endert
School of Interactive Computing
Georgia Institute of Technology

Dr. Nan Cao
College of Design and Innovation and College of Software Engineering
TongJi University

Dr. Duen Horng (Polo) Chau
School of Computational Science and Engineering
Georgia Institute of Technology

Date approved: December 2, 2020

ACKNOWLEDGMENTS

I first would like to thank my advisor, Dr. Haesun Park for her guidance and support during my PhD years. I also like to thank Dr. Alex Endert, who is like the second advisor to me, for inspiring me and guiding me. Without Haesun and Alex, this dissertation would not have been possible.

I would like to thank the rest of my committee members, Dr. Polo Chau, Dr. Chao Zhang, and Dr. Nan Cao for their insightful comments and feedback. I am grateful for my mentors: Jaegul, whose mentorship shaped my research, Eser and Albert, who took a chance on me. I would like to acknowledge my collaborators including Barry, BC, Chandan, Jingu, and so many more. Thank you to my labmates, Changhyun, Ramki, Da, Seungyeon, Woosang, Rundong, Qingqing, Srini, Dongjin, and Koby for the camaraderie. Thank you to my vismates including Alex, Arjun, Arpit, Bahador, Chad, Emily, Fred, Grace, Haekyu, Hayeong, HP, John, Matt, Minsuk, Ramik, Shenyu, Subhajit, Terrance, Tim, and Yi for your advice and friendship. Thank you to my studymates, Bonggun, Chanho, Eunji, Hyeokhyen, Hyungjoon, Jinwoo, Sungtae, and Susie for making DL fun. To my personal friends, Hyojong, Hyoukjun, Jaehoon, Chaeyi, Sookyung, Joonseok, Hyemin, Taeheon, Edward, Kisung, Joohwan, Jiyeon, and Jaewon: thank you for all the moments we shared.

I cannot thank my parents and my sister enough for their endless support and love. Thank you for always believing in me, praying for me, and loving me unconditionally.

Above all, thank you Jesus for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Human-in-the-Loop machine learning leverages both human and machine intelligence to build a smarter model. Even with the advances in machine learning techniques, results generated by automated models can be of poor quality or do not always match users' judgment or context. To this end, keeping human in the loop via right interfaces to steer the underlying model can be highly beneficial. Prior research in machine learning and visual analytics has focused on either improving model performances or developing interactive interfaces without carefully considering the other side.

In this dissertation, we design and develop interactive systems that tightly integrate algorithms, visualizations, and user interactions, focusing on improving interactivity, scalability, and interpretability of the underlying models. Specifically, we present three visual analytics systems to explore and interact with large-scale text data. First, we present interactive hierarchical topic modeling for multi-scale analysis of large-scale documents. Second, we introduce interactive search space reduction to discover relevant subset of documents with high recall for focused analyses. Lastly, we propose interactive exploration and debiasing of word embeddings.

CHAPTER 1
INTRODUCTION

Machine learning enables computers to automatically learn and improve from experience [1]. That is, machine learning techniques build a model based on training data without being explicitly programmed. Powered by massive data growth, machine learning is affecting various aspects of our daily lives nowadays. From shopping recommendations, fraud detection, and health analytics, to self-driving cars, it is almost harder to find an enterprise that is not partly automated by machine learning. The problem is that machine learning models need *good* data, both in quantity and quality. Machine learning cannot perform well when there is not enough data, the data is unbalanced, it is noisy or of poor quality, or it contains bad information such as human biases. In these cases, learning from data is not enough - human intervention is necessary.

Human-in-the-loop machine learning leverages both human and machine intelligence to build a smarter model. During the model building process, human-in-the-loop machine learning allows users to interact with the model through interactive interfaces. Numerous works have been done in interactive machine learning in the machine learning community and visualization community. However, many of these works have focused on either improving model performances using user feedback or developing interactive interfaces for human interaction, without carefully considering the other side.

We advocate that both the algorithmic side and the user side should be taken into consideration when designing a human-in-the-loop machine learning system. To do so, we choose interactive visualization as a medium for machine-human interaction. Interactive visualization can help the users quickly understand data by showing the overview of the data and allowing them to explore items of interest. Also, visualization could be used as an interactive interface for the users to steer the underlying model, through direct parameter

adjustments or semantic interactions on visual objects [2]. In this dissertation, *we design and develop interactive systems that tightly integrate algorithms, visualizations, and user interactions, focusing on improving interactivity, scalability, and interpretability of the underlying models.* Specifically, we focus on analyzing large-scale text data.

Over the past decades, there has been a deluge of text data from traditional sources such as news and research articles as well as recent electronic sources such as web pages, social networking services, online forums, and online encyclopedia. Due to the sheer volume and the noisy unstructured nature of these data, discovering useful knowledge from large document collections is a challenge. Numerous text mining methods have been introduced to effectively summarize and organize documents without going through every document. However, automatically generated topic summaries do not always match human judgment or context, can be of poor quality, or just do not make sense sometimes. To this end, keeping human in the loop to steer the text mining process can be highly beneficial for generating better quality topics that align with users' needs.

In the following chapters, we try to solve these three research questions (RQs) using interactive visual text analytics:

RQ1: Understanding and Organizing Large-scale Text Data (Chapter 3)

Human-in-the-loop topic modeling allows users to explore and steer topics to produce better quality topics that align with their needs. When integrated into visual analytic systems, existing topic models are given interactive parameters that allow users to tune or adjust them. However, this has limitations, as non-interactive algorithms are often used and adapted for this interactivity. Instead, we present the concept of tight integration, which advocates for the need to co-develop interactive algorithms and interactive visual analytic systems in parallel to achieve *flexible interactivity*. Another challenge in many existing systems is *scalability*. We utilize hierarchical topic model to allow multi-scale sensemaking for large-scale text data. To instantiate our concept, we present ArchiText, a prototype system for interactive hierarchical topic modeling that offers fast, flexible, and algorithmi-

cally valid analysis via tight integration. Utilizing interactive hierarchical topic modeling, our technique lets users generate, explore, and flexibly steer hierarchical topics to discover more informed topics and their document memberships.

RQ2: Finding the Relevent Parts of the Data (Search Space Reduction) (Chapter 4)

Often in practice, users want to focus on specific aspects or "targets" rather than the entire corpus. For example, given a large collection of documents, users may want only a smaller subset which more closely aligns with their interests, tasks, and domains. In particular, Chapter 4 focuses on large-scale document retrieval with high recall where any missed relevant documents can be critical. A simple keyword matching search is generally not effective nor efficient as 1) it is difficult to find a list of keyword queries that can cover the documents of interest before exploring the dataset, 2) some documents may not contain the exact keywords of interest but may still be highly relevant, and 3) some words have multiple meanings, which would result in irrelevant documents included in the retrieved subset. We present TopicSifter, an *interactive* and *scalable* visual analytics system for search space reduction. Our system utilizes targeted topic modeling based on nonnegative matrix factorization and allows users to give relevance feedback in order to refine their target and guide the topic modeling to the most relevant results.

RQ3: Improve Interpretability and Fairness in Text Analytics (Chapter 5)

Word embeddings are commonly used as building blocks for most natural language processing models. However, their dimensions are not easily interpretable and they are shown to reflect stereotypes and biases on sensitive attributes (e.g., gender, race, religion, etc). For example, 'man' is to 'computer programmer' as 'woman' is to 'homemaker' (Bolukbasi et al., 2016). Several recent works have proposed methods to measure these negative associations and ultimately debias word embeddings. However, most of these techniques are based on the selection of attribute-specific words (such as 'man' and 'woman') and attribute-neutral words (such as 'computer programmer' and 'homemaker'), which can be subjective and may vary depending on the used corpus or the application context. In Chap-

ter 5, we introduce a visualization system for interactive exploration and debiasing of word embeddings. Our system allows users to construct *interpretable* attribute axes, visually examine attribute biases within word embeddings, and iteratively *debias* them.

1.1 Thesis Statement

We design and develop interactive visual text analytics systems that tightly integrate algorithms, visualizations, and user interactions, focusing on improving interactivity, scalability, and interpretability of the underlying models. In particular, we propose:

1. Interactive Hierarchical Topic Modeling for multi-scale analysis of large-scale documents
2. Interactive Search Space Reduction to discover relevant subset of documents for focused analyses
3. Interactive Exploration and Mitigation of Biases in word embedding

1.2 Outline of Thesis

In Chapter 2, we review previous works on text visualizations. Chapter 3 introduces interactive hierarchical topic modeling with tight integration between algorithm, interaction, and visualization. Next, Chapter 4 proposes interactive search space reduction through targeted topic modeling. Chapter 5 presents an interactive interface to examine and mitigate biases in word embeddings. Finally, Chapter 6 concludes the dissertation and suggests future work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we review recent literature on text visualization techniques. Sections 2.1 and 2.2 describe visualization techniques summarizing corpus contents using topic modeling or document clustering. Finally, Section 2.3 offers some conclusions. More in-depth related work for specific applications will be described in the later chapters.

2.1 Corpus Summarization/Topic Modeling

Organizing large, unstructured document collections into semantically meaningful topics and visualizing them has been a widely studied problem in the visualization community. Even before the introduction of modern topic modeling methods such as Probabilistic Latent Semantic Analysis (PLSA) [3], Latent Dirichlet Allocation (LDA) [4], and Nonnegative Matrix Factorizatoin (NMF) [5, 6, 7, 8], earliest works including Topic Island [9] and IN-SPIRE [10] visualize static themes extracted from documents and let users explore

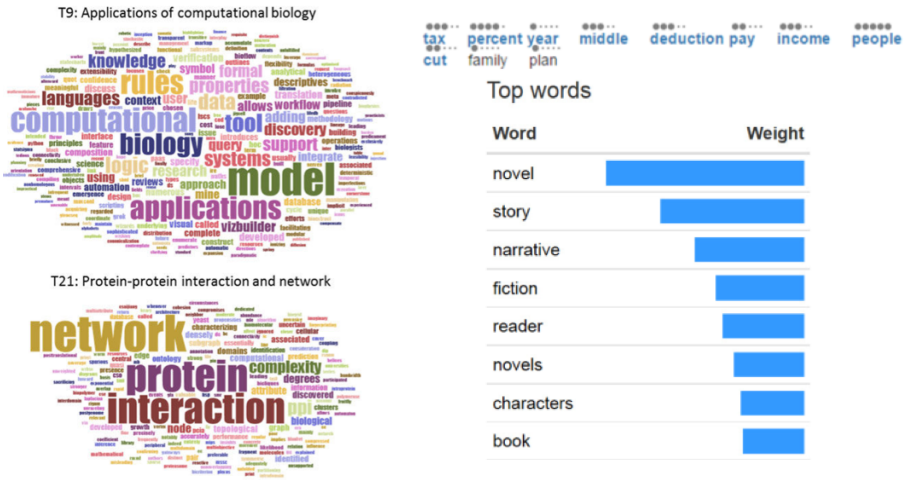


Figure 2.1: Topics visualized as wordcloud (left) and lists of keywords with glyphs (right-top) or a bar chart (right-bottom).

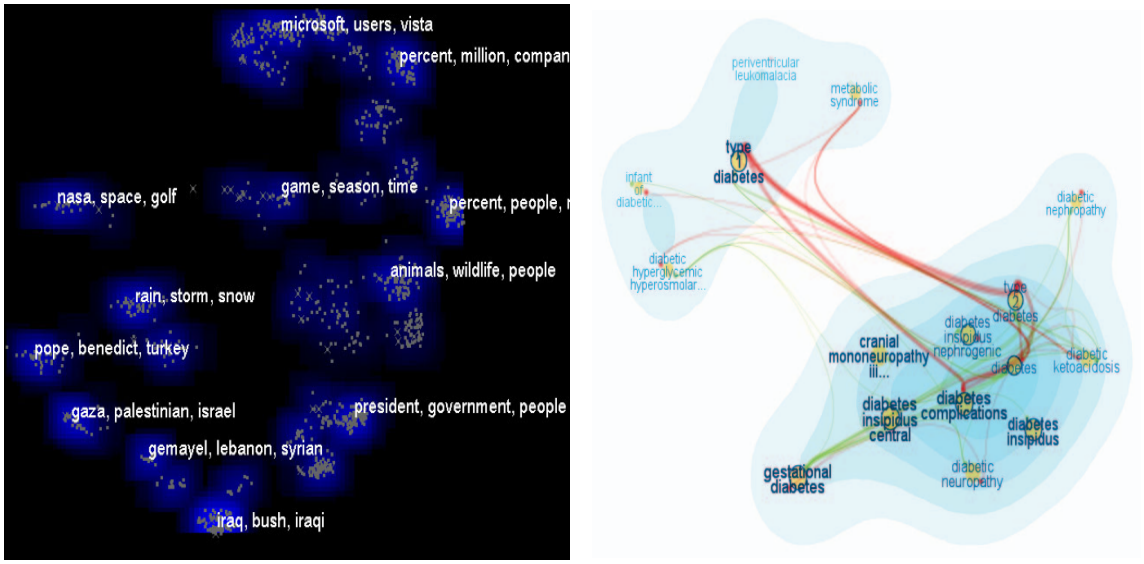
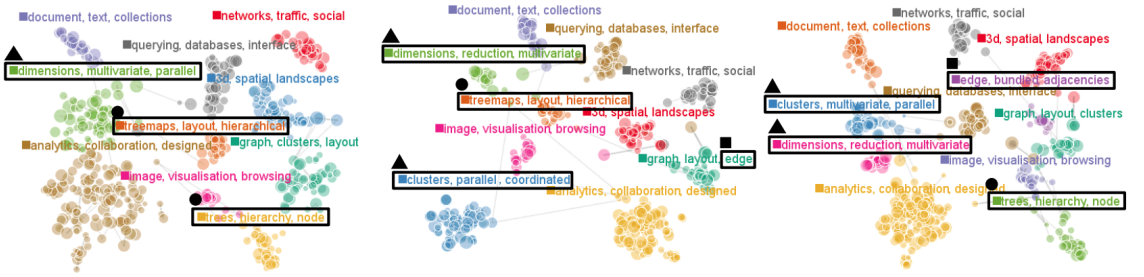


Figure 2.2: Topics visualized as a scatterplot in INSPIRE [10] (left) and contours in FacetAtlas [11] (right).



(a) The initial visualization

(b) The visualization after *topic keyword refinement* (c) The visualization after *keyword-induced topic creation* and *topic splitting*

Figure 2.3: Interaction with topics in UTOPIAN [12]. InfoVis-VAST dataset is used.

them. With the modern topic modeling methods available, research on text visualization has been substantially accelerated. Specifically, many methods have focused on visualizing topic analysis results. Topics are generally represented as a set of most representative keywords. Wordclouds are a frequently-used visualization technique to illustrate a topic and its keywords (Fig. 2.1(left)). Another frequently used visualization technique is an ordered list of keywords with font size or additional bar chart indicating their importance (Fig. 2.1 (right)). While simple, this approach allows users to see a ranking of the most important words and quick comparisons.

In many cases, similarity between topics or documents is taken into account. Docu-

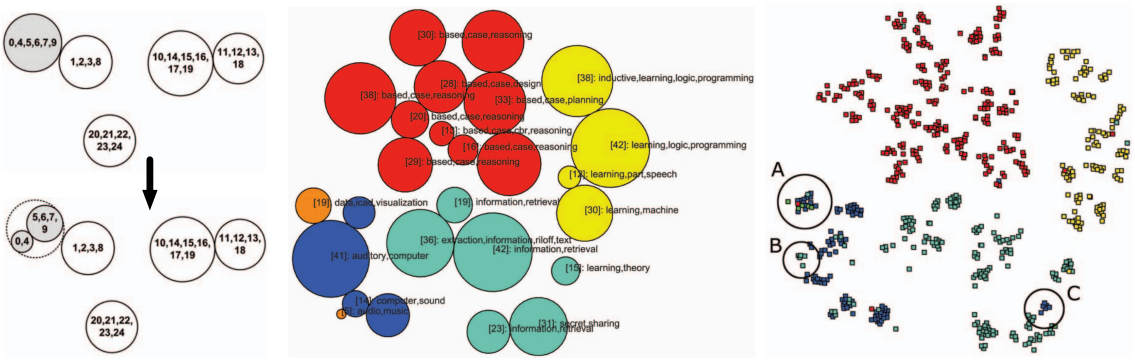


Figure 2.4: Visualizations by HiPP [21]. (Left) A topic node is expanded into two nodes. (Middle) Top level clusters of scientific papers. (Right) All clusters are expanded to data-instance level.

ments are represented as two-dimensional or three-dimensional points by applying dimension reduction techniques to the topic analysis results. In this way, similar documents and similar topics are placed closer to each other. For instance, UTOPIAN [12] maps documents onto 2D scatter plot, on which clusters are labeled with keywords, and users can interact with the topic modeling algorithm. Other examples include iVisClustering [13], TopicPanorama [14], and TopicLens [15]. Several works such as ContexTour [16], FacetAtlas [11], SolarMap [17], and [18] adopt contours to represent static topics and relationship between them (Fig. 2.2).

Often, automatically generated topics can be of low quality and noisy; or may not align well with user's mental model. In these cases, human-in-the-loop topic model techniques allow users to steer underlying topic models to obtain better results, e.g., UTOPIAN [12] (Fig. 2.3). Various topic-level, document-level, word-level interactions are introduced to add, modify, split, combine, and remove topics and its representative documents and keywords [19, 13, 12, 20, 15, 18].

2.2 Multiscale Document Analysis

As the text data and vocabulary grow larger, visualizing a large number of topics has become more challenging. To this end, some systems adopt hierarchical document clustering

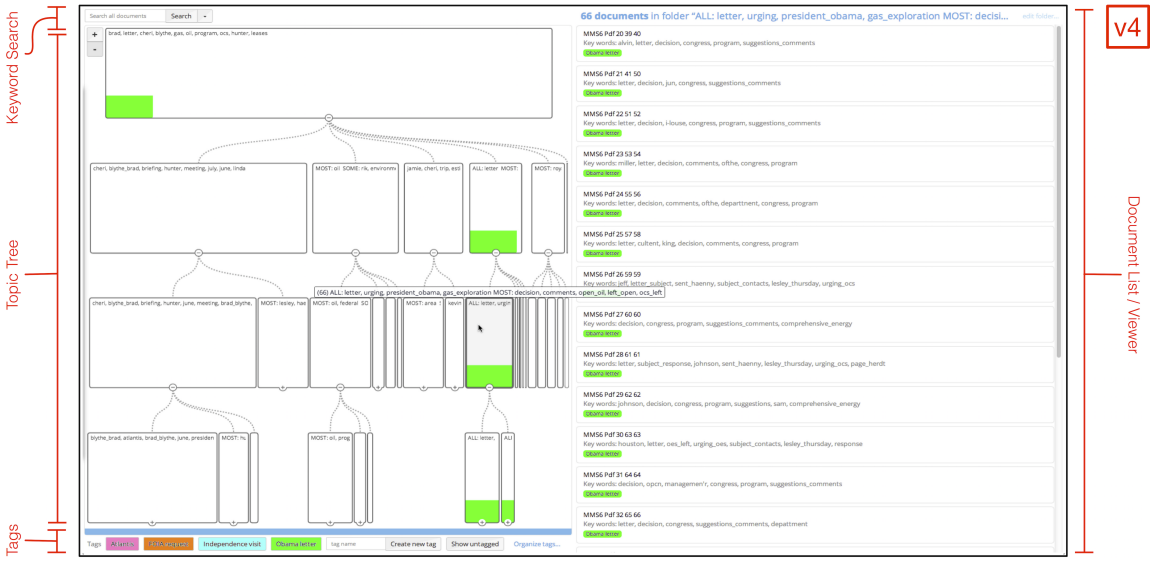


Figure 2.5: White house emails hierarchically clustered and visualized as a tree (left) by Overview [22]. Clusters containing documents having a tag "obama letter" are highlighted in green.

techniques to support multiscale (or multilevel) document analysis. Documents are hierarchically clustered according to their topic similarity, which results in a topic hierarchy. Users navigate through topic hierarchy visualizations to find coarser (higher nodes) or finer grain topics (lower nodes).

HiPP [21] uses a hierarchical circle packing algorithm, in which a topic or a document is represented as a circle. In Fig. 2.4, topic circles can be expanded into sub-topic circles down to individual documents.

Other systems directly visualize a topic tree using node-link style visualizations. For example, Brehmer et al. [22] introduces Overview[1], a visual document mining tool for investigative journalists. Overview allows users to explore the topic hierarchy and annotate relevant documents for later use (Fig. 2.5). Similarly, Dou et al.. [23] visualize topics and their temporal patterns as tree and themeriver charts, respectively (Fig. 2.6). More recently, Hoque and Carenini [24] utilize a simple collapsible tree in an online conversation analytics system and allow users to explore and revise topic hierarchy by moving topic nodes.

[1]https://www.overviewdocs.com/

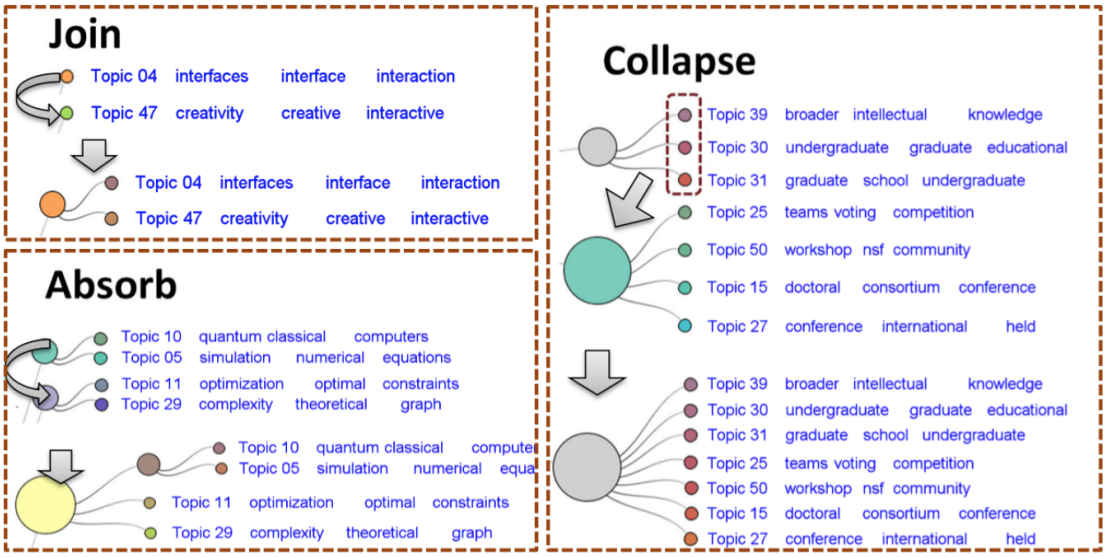


Figure 2.6: Topic tree visualization in [23]. Topics are represented as a node with its top keywords. Three topic hierarchy modifications are supported.

Sunburst visualizations [25] are also used to visualize topic hierarchies [26, 27]. As seen in Fig. 2.7, concentric circles represent different levels of topic hierarchy starting from the the center (source node) to the outermost (leaf nodes). This visualization technique relies strongly on user interaction to allow users to expand sub-components of the tree if requested.

2.3 Summary

We reviewed recent results on various text visualization systems. Most text visual analytics systems show overall visual summary first, and support detail-on-demand through user exploration processes. In these systems, topics or document cluster are explored first due to the high-dimensional nature of text data, and then documents or words. Typically used visualization techniques are wordclouds and keyword lists for individual topics; scatterplots, node-link diagrams, and tables for topic/document relations; circle packing, sunbursts, treemap, and tree for topic hierarchy; and line charts and themeriver for temporal trends. Various user interactions are utilized for sub tasks such as filtering, zooming, mod-

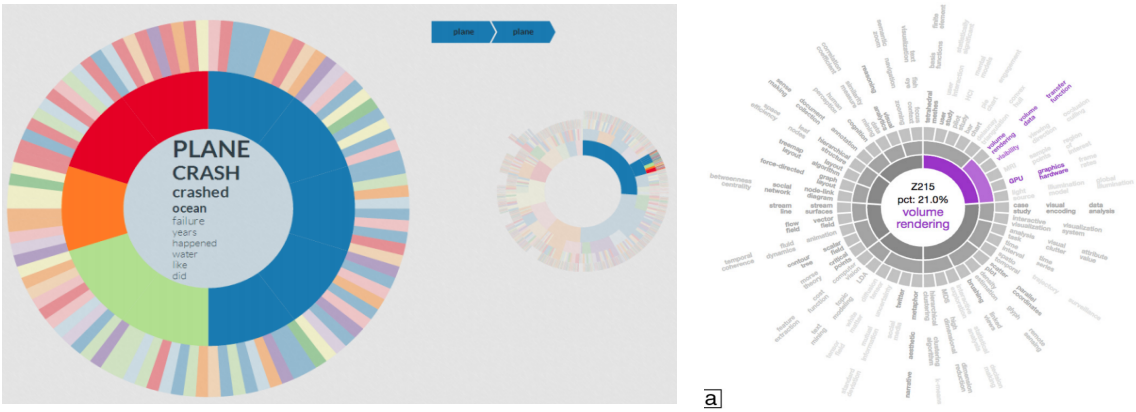


Figure 2.7: (Left) A topic subtree and its location in the original tree in Hierarchie [26]. (Right) Document cluster hierarchy in VISTopic [27].

ifying topics, etc.

In many systems, interaction capabilities are static in the sense that they update visualization based on pre-computed results and not based on the updated computed results (e.g., zooming, filtering). Only a number of systems (e.g., [13, 12, 15, 24]) allows users to actively steer underlying models, but the types of supported interactions are limited based on algorithmic convenience rather than users' needs. In my thesis work, our goal is to develop more flexible algorithms with user-centered interaction design. In addition, when steering models, some systems re-run the entire algorithm with different parameters. Instead, we plan to adaptively update the models, which is more efficient and is possible by tight integration between user interaction and computation. Finally, most of current research utilizes toy-sized data. For the interactive systems to be usable for real world tasks, we need to design a scalable system in terms of both algorithms and visualizations.

CHAPTER 3

ARCHITEXT: INTERACTIVE HIERARCHICAL TOPIC MODELING

In this chapter, we present our interactive hierarchical topic modeling technique to explore, understand, and organize large-scale text data. Unlike previous works with limited steerability, our work offers users various flexible interactions that are tightly integrated with underlying algorithms and visualizations. Also, our system can handle large-scale corpus to offer better scalability. This chapter is adapted from our work that is publised in TVCG.

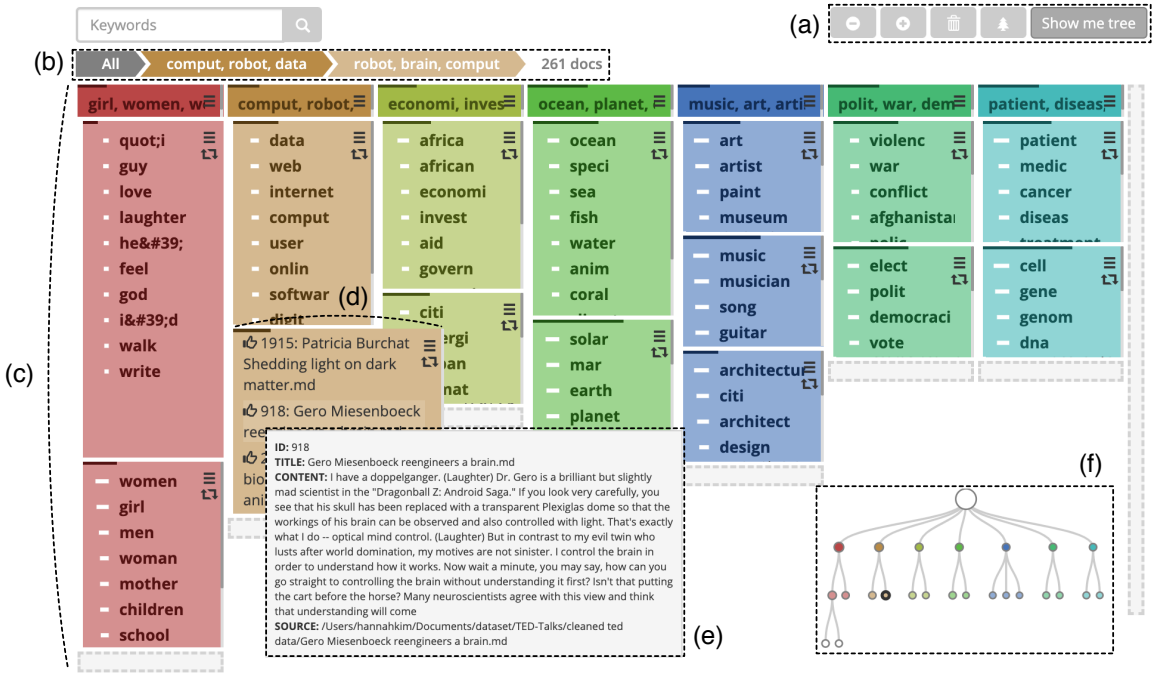


Figure 3.1: The ArchiText system. The topic workspace mode has (a) a control bar, (b) a breadcrumb view, (c) a topic card view, and (f) a mini overview. The topic card view shows topic cards, which can be flipped to show (d) documents in the selected topic. (e) A detail view pops up when the mouse hovers over a document.

3.1 Introduction

Analysis of large-scale text collections has been a widely studied research topic in the data analytics community. In particular, it is challenging to obtain an effective overview of text data and discover useful insights without going through each data item. This is untenable due to the sheer volume and the noisy, unstructured nature of text data. To solve this, various computational topic modeling techniques such as Probabilistic Latent Semantic Analysis (PLSA) [3], Latent Dirichlet Allocation (LDA) [4] and Nonnegative Matrix Factorization (NMF) [28] have been developed in recent decades. These methods provide content overviews by computing semantically meaningful topics as keyword distributions, and organize documents within the topics.

Recent advances in topic modeling have resulted in many new formulations and algorithms. However, even with the advances in topic modeling methods, results generated by these completely automated computational approaches depend largely on the problem formulation involving some objective functions and the corresponding algorithms. As a consequence, the computed results do not always match human expectations or context, and can be of poor quality, or difficult to make sense of [29, 30]. Human-in-the-loop approaches can be highly beneficial for generating higher quality topics that align with users' needs and domain expertise. Furthermore, interactive exploration can be critical to foster understanding through discovery [31].

Many visual analytics systems for text data have been developed for human-in-the-loop topic modeling. These systems present a high-level overview of the text data by visualizing the topics generated by automated topic modeling algorithms and allow users to explore items of interest and, if possible, steer the underlying model. However, there are two major challenges in existing interactive topic modeling systems: *scalability* and *limited steerability*. First, most existing interactive topic modeling approaches can handle hundreds or thousands of documents, but they are not suited for large-scale datasets in

real world scenarios. One reason is that in many systems, the underlying topic model is treated as a black box, which is recomputed from scratch after each interaction. In order to utilize human-in-the-loop topic modeling in real world applications, we need an efficient and scalable way to interact with a massive set of documents. Next, many visual analytics systems are often not well-suited for interactive analysis since they are built on top of existing automated topic modeling methods. In other words, interaction capabilities offered in these systems are limited by the convenience of the adopted methods [30]. An example is LDA [4], one of the most celebrated methods in topic modeling. Despite its popularity, LDA has several issues which hinder its integration with visual analytics systems. For instance, its parameters are not easy to understand and tune for non-experts. Sometimes even a small parameter change results in unpredictable side effects [30]. Also, LDA results are less consistent over different runs [12], which makes it difficult for users to trust the model and to see if interactions are properly reflected.

In this chapter, we propose *hierarchical* topic modeling in contrast to *flat* (non-hierarchical) topic modeling in the context of visual analytics. Flat topic modeling is limited for visualizing large-scale text data. As the text data and vocabulary grow larger, the need for interacting and visualizing a larger number of documents and topics also grows, and it becomes more challenging to better represent the underlying data. However, since the computation of a very large number of topics at once is limited by computation capacity, display size, and visual understanding, the number of topics generated by flat models tends to be limited, and accordingly the topics are rather general and coarse-grained. On the other hand, hierarchical topic modeling offers better understanding of the data corpus by representing information at multiple levels of detail, and allowing people to interactively provide feedback at different aggregation levels. Using a hierarchy, users can explore high-level coarse topics and zoom in on fine-grained topics. Users can drill down into a subset of data to increase understanding (sense-making) and organize computed topics into a hierarchy that matches users' mental model. By focusing on steering unclear parts and leaving the rest

to the computational methods, more efficient and comprehensive discovery is possible. In addition, flat topic modeling methods assume that all topics are at the same level, regardless of semantic granularity or size. For example, flat topics generated from sports articles during world cup season may contain many topics related to soccer but few topics on other sports. On the other hand, users may want to organize topics into various levels according to their mental models, e.g., by sports.

Despite the aforementioned advantages of hierarchical topic models over flat models, limited work has been done on interactive hierarchical topic modeling. A few visual analytics systems that support hierarchical topic modeling offer interactions to explore multiple levels of the underlying hierarchy, but their capability to modify or steer the underlying model is limited. For flexible steerability, we propose that the visualization systems, underlying computational algorithms, and users' interactions should be *tightly integrated*. Tight integration refers to the algorithm, visualization, and user interaction being jointly developed, where all three components are considered throughout the design process. Visualizations should not only show the outputs of models, but serve as the medium for interaction. Interaction should not be limited to controlling some parameters of algorithms, but allow higher-level operations that support the discovery process. Finally, algorithm development should take into account not only automated performance metrics, but consider interactivity and transparency for visualization in their formulation. This process of co-development goes beyond the adaptation of existing methods to meet the needs of users or interactive tasks, but instead co-designs algorithms, interactions, and visualizations simultaneously to ensure proper synchronization, compatibility, and performance. The proposed work explores the paradigm of tight integration and proposes a new way to implement tight integration for interactive hierarchical topic modeling.

In this chapter, we present ArchiText, a visual analytic system using hier**Archi**cal **I**nteractive topic modeling for large-scale **Text** data. ArchiText visualizes hierarchical topics and offers various interactions to steer the topics and their hierarchical structure.

ArchiText closely integrates the computational formulation of the model with the interactions provided to support flexible and rapid updates. The primary contributions of this work include:

- Development of an interactive and hierarchical topic modeling algorithms that achieve tight integration among visualization, computational model, and visual representation.
- Implementation of a visualization prototype system for large-scale document analysis utilizing our interactive hierarchical topic modeling framework.

3.2 Related Work

3.2.1 Visualization of Text Corpora

Organizing large, unstructured document collections into semantically meaningful topics and visualizing them has been a widely studied problem in the visualization community [38]. The earliest works including Topic Island [9] and IN-SPIRE [10] visualize document items and extracted themes. With the introduction of modern topic modeling methods such as Probabilistic Latent Semantic Analysis (PLSA) [3], Latent Dirichlet Allocation (LDA) [4], and Nonnegative Matrix Factorization (NMF) [28], research on text visualization has been substantially accelerated. Specifically, many results have focused on visualizing topic analysis results and allowing interactive exploration of topics and data items without the ability to steer the topic modeling results.

Once computed, topics are generally represented as a set of the most representative keywords. In many cases, the similarities between topics or documents are taken into account. Documents are represented as two-dimensional or three-dimensional points by applying dimension reduction techniques to the topic analysis results. In this way, similar documents and similar topics are placed closer to each other. For instance, UTOPIAN [12] maps documents into a 2D scatterplot, in which clusters are labeled with keywords, and users can interact with the topic modeling results. Other examples include iVisCluster-

Table 3.1: User interaction tasks for model steering supported (or suggested) in previous works. Parenthesis indicates suggested interaction tasks.

Unit	Interaction Tasks	Reasons/Goals	Prior Work
Word	Create a topic by seed word(s)	Need a new topic around the word(s).	[12, 32]
	Add word(s) to a topic	The word(s) is relevant to the topic.	[33, 32] ([30])
	Move word(s) from a topic to another.	The word(s) is more relevant to another topic.	
	Remove word(s) from a topic	The word(s) is not relevant to the topic.	[33, 32] ([30])
	Confirm or reject a word from a topic	The word(s) is definately relevant/irrelevant for the topic.	[34, 35]
	Change word distribution of a topic	The topic is better represented with new word distribution.	[12, 13, 33] ([30])
	Add word(s) to stopword list	The word(s) is not good representative of the data.	[34, 33]
Doc	Create a topic by seed document(s)	Need a new topic around the topic of the document(s).	[12]
	Move document(s) from a topic to another	The document(s) belongs to another topic.	[32, 13] ([30])
	Remove document(s) from a topic	The document(s) is irrelevant or of low quality.	[33, 13] ([30])
	Confirm or reject a document from a topic	The document(s) is relevant/irrelevant for the topic.	[36]
Topic	Merge two topics into a topic	Two topics are very similar.	[12, 13, 37, 33, 23, 32]
	Split a topic into subtopics	The topic is too broad or not coherent.	[12, 13, 37, 33, 32] ([30])
	Move a topic	The topic belongs to another branch.	[13, 24, 23]
	Remove a topic	The topic is irrelevant, uninteresting, or of bad quality.	[13, 24, 32] ([30])
	Restore a topic	Need to undo 'remove topic'	
	Fix/freeze a topic	The topic is final and no more refinement is needed.	
	Collapse topics (show fine-grained)	Topics are too general; there are not enough topics.	[24]
	Aggregate topics (show coarse-grained)	Topics are too specific; there are too many topics.	[24]

ing [13] and TopicLens [15]. Several works such as ContexTour [16], FacetAtlas [11], SolarMap [17], and Concept Visualizer [18] adopt contours to represent static topics and relationships among them.

3.2.2 Interactions in Topic Modeling

Fully leveraging interactivity provided by visual analytics, several systems have incorporated a 'human-in-the-loop' approach to interactively modify the underlying topic model. Automatically generated topics often can be of low quality and noisy; or may not align well with user's mental model. In these cases, human-in-the-loop topic modeling techniques allow users to steer underlying topic models to obtain better results [29]. For example, if automatically generated topics contain two similar topics, users may want to merge them into a single topic. To this end, various interactions are introduced including *add, modify, split, combine, and remove* topics, documents representing the topics, and keywords [12, 13, 37, 33, 23, 32, 34, 24, 35, 36]. We have surveyed interactive topic modeling systems with model steerability and organized the user interactions into word-level, document-level, and topic-level based on the unit of interactions. Specifically, the word-level interactions include user's activities of refining topics by adding words to a topic, moving words between topics, removing words from a topic, re-weighting word importance for a topic, and creating a new topic using selected words. Similarly, the document-level interactions involve user's editing of topics (which can be viewed as document clusters) by moving documents from one topic to another topic, removing documents from its parent topic, re-weighting document importance for a topic, and creating a new topic using selected words. The topic-level interactions occur when users perform group-wise interactions such as merging, splitting and removing topics. A complete list of user interactions that are available in some of the existing interactive topic model systems is summarized in Table 3.1. However, user interactions in many prior works have been designed for algorithmic convenience rather than user tasks [30], and thus are not tightly coupled with the underlying algorithms.

3.2.3 Visual Analytics for Hierarchical Topic Modeling

In a number of recent papers, the topics are organized with a hierarchy. Hierarchical topic modeling and hierarchical document clustering techniques organize documents into various granularity of topics. In this way, large text corpora can be analyzed and understood through multi-scale analysis. In hierarchical visual analytics systems, users interactively navigate through topic hierarchy visualizations to find coarser grained (higher nodes) or finer grained (lower nodes) topics. For instance, HiPP [21] uses a hierarchical circle packing algorithm, in which a topic or a document is represented as a circle. Topic circles can be expanded into sub-topic circles down to individual documents. Other systems directly visualize a topic tree using node-link style visualizations. For example, Brehmer et al. [22] introduces Overview, a visual document mining tool for investigative journalists. Overview allows users to explore the topic hierarchy and annotate relevant documents for later use. Similarly, Dou et al. [23] visualize topics and their temporal patterns as tree and themeriver charts, respectively. Other works focus on the evolution of topic hierarchies [20, 39] and matching topic hierarchies from multiple sources [40]. A work more closely related to what we propose here is by Hoque and Carenini [24] which utilizes a simple collapsible tree in an online conversation analytics system and allows users to explore and revise a topic hierarchy by moving topic nodes. Sunburst visualizations [25] are also used to visualize topic hierarchies [26, 27], where concentric circles represent different levels of the topic hierarchy starting from the the center (source node) to the outermost (leaf nodes). This technique relies strongly on user interaction to allow users to expand sub-components of the tree if requested.

Among these hierarchical topic modeling systems, only a few support an interactive modification of the underlying hierarchical model [23, 24]. These systems only offer group-wise or topic-level organizational operations such as merging topics, splitting a topic, and moving a topic under a new parent. Therefore, users are unable to steer the underlying topic model to a finer degree (e.g., by words and/or documents). More recently, IHTM [32]

proposes a mixed-initiative approach where a human can intervene during the incremental model building process. Users are asked to choose from several interaction strategies with the help of a preview of the expected outcome of each strategy. While this work has many advantages that work well for very small datasets, it is not scalable to large-scale datasets because the underlying topic hierarchy is optimized every time a data item is entered. Also, the interaction strategies available in the IHTM system are limited and its word-level interactions are offered only before the algorithm starts.

3.2.4 Interactive Model Steering in Visual Analytics

Mixed-Initiative systems [41], which combine human knowledge and human intelligence to create a collaborative system between users and machines, are closely related to what we propose in this chapter. Adopting this principle, in mixed-initiative visual analytics systems, users interact with the machine via visual interfaces to *steer the model* by controlling different model parameters. Some systems offer direct manipulation of model parameters through control panels. However, direct manipulation of model parameters requires a deep understanding of the underlying model mechanism and its parameters. Endert et al. [2] introduce 'semantic interaction' to steer the models using native user interactions on visual objects rather than model parameters. For instance, Disfunction [42] and Observation-Level Interaction [43] allow users to move points in a 2D scatterplot to update the underlying distance function. Podium [44] updates an SVM model as users change the order of data items. Other examples include [45, 46, 47, 48].

3.3 Interactive Topic Modeling with Tight Integration

In this section, we propose our novel approaches for interactive hierarchical topic modeling. We first identify design goals for tight integration in interactive hierarchical topic modeling. Then we propose our modular interaction design to support flexible user feedback. Finally, we describe the underlying algorithms for base operations and interaction tasks.

3.3.1 Design Goals for Tight Integration

Tight integration advocates for the visualization accurately representing the computational result with reasonable responsiveness, user interaction being accurately interpreted taking advantage of more detailed information that the underlying algorithm offers, and flexibility in the model to accommodate a wide range of user tasks and goals. Here, we list the design goals of tight integration and how ArchiText achieves them.

Fast, Adaptive, and Interaction-conductive Model and Algorithm. The foundational algorithm should be designed and developed with user interaction considered from the start. Tight integration synchronizes updates in the underlying computation with the interpretation of the user interaction. This update cycle is iterative, where the underlying computational methods guide the changes in the intermediate results taking the user interaction into account. To achieve fast and accurate visualization updates, the underlying updates of the computational result should not involve recomputing the solution from scratch. Rather, underlying computations should be tailored to allow incremental, timely, and responsive updates. In our proposed system, results are adaptively updated based on intermediate solutions and user refinement.

Visualization of Various Degrees of Information and User Feedback. The computational results, the internal factors, and characteristics of the algorithms should be exposed in various degrees of information level, likely through multiscale visual representations. Interactively, users may perform operations on specific information, yet the algorithmic interpretation of the action will need to consider additional information. For instance, removing a topic is likely based on a subset of keywords for a topic shown in the visualization. However, the underlying algorithm contains many additional details about the topic, such as the complete keyword distribution (what a topic is) and the topic distribution for each document (how close a document is to a topic). Without careful interpretation of user intention incorporated into algorithms, the results can quickly be distorted after multiple interactions because of the limited information users are shown in the visualization. In ad-

dition, with a hierarchical topic modeling, visualizations can show multiple levels of detail, aggregating or de-aggregating sub-hierarchies depending on the level of detail requested by users.

Capability to Support a Variety of User Feedback Types. The computational models and algorithms should be flexible enough to incorporate various user intentions and tasks. Various model steering interactions have been identified as important (Table 3.1), but not all interactions were supported in a single system previously. One reason is that most topic modeling methods have many parameters and settings that are difficult to properly tune to produce the results that meet user expectations. We chose Nonnegative Matrix Factorization (NMF) [28] as our underlying foundational topic modeling method due to its flexibility and efficiency. Multiple advantages of NMF that we have observed and analyzed in our previous work [6, 12, 7] include fast algorithms, higher quality and more consistent solutions, flexibility to changes in tasks, adaptive updating methods [49, 50, 51], and interpretability of results. In addition, since interactions can be formulated as constrained NMF problems, we can identify a set of primitives that are common over various interaction tasks and build core computational modules that can be utilized across them as will be described in the next section. These important features of our algorithm combine to facilitate our tight integration methodology.

3.3.2 Interaction Primitives for Hierarchy Steering

To achieve the goal of the tight integration, we propose to break down a large suite of interaction tasks into basic operations called primitives. These primitive operations can be optimized individually and then combined to implement specific subtasks. Surveying interaction tasks supported or suggested in existing works, we observed that all interaction tasks can be further divided into tree operations and/or supervised topic computation. For instance, *Move a topic into a new parent* (**MoveT**) can be achieved by 'cut out a topic' followed by 'insert the topic under a new parent' with re-computations. Based on interac-

Table 3.2: Key notations used in Chapter 3.

Notation	Description
T_i	The i-th topic node
$D(T_i)$	Indices of documents that belong to T_i
$p(T_i)$	The parent topic node of T_i
C	Trash can, i.e., the set of removed topics
m	The number of keywords
n	The number of documents
n_i	The number of documents in T_i
k_i	The number of child topics under T_i
$X^{(i)}$	The $m \times n_i$ word-document matrix of T_i
$W^{(i)}$	The $m \times k_i$ word-topic matrix of child topics of T_i
$w_p^{(i)}$	The p-th column of $W^{(i)}$
$H^{(i)}$	The $k_i \times n_i$ topic-document matrix of child topics of T_i
$h_q^{(i)}$	The q-th column of $H^{(i)}$
$\mathbb{R}_+$	The set of nonnegative real numbers
$\|\| \cdot \|\|_F$	The Frobenius norm
$A_{r\cdot}$	The r-th row of matrix A
$A_{\cdot r}$	The r-th column of matrix A
$argmax(a)$	The index of the largest element in vector a

tion tasks supported in existing systems and our design goals, we come up with five base operations, whose combination can form the interaction tasks in Table 3.1. This modular implementation makes it possible to optimize the tightly coupled system performance by fine tuning the five base operations. The five base operations are as follows:

1. **makechildren(T)**: Create two child topics for a leaf node topic T
2. **merge(T1, T2)**: Merge sibling topic nodes T1 and T2
3. **insert(T1, T2)**: Insert a topic node T1 under a new parent node T2
4. **cut(T1)**: Cut out a topic node T1
5. **recompute(T)**: Recompute child topics of T using a constrained NMF topic model

Utilizing the base operations, we simultaneously design user interactions and corresponding algorithms. The complete list of supported interaction tasks is described in Tables 3.3-3.5 and Section 3.3.4.

3.3.3 NMF for Topic Modeling

We define our notation and topic modeling formulations as follows. Conceptually, a topic T is identified as a keyword distribution and has a set of documents that belong to the topic (i.e., the documents whose topic distribution has the strongest weight in the topic). A topic node in the topic hierarchy is denoted as $(T, D, p(T))$ where T denotes the topic, D the set of documents in the topic, and $p(T)$ a reference to the parent topic of the topic T. We denote the i-th topic as T_i and the indices of documents that belong to T_i as $D(T_i) = \{d_{i_1}, \cdots, d_{i_{n_i}}\}$, where n_i is the number of documents in the topic. We reference the parent topic node of a child topic T_i as $p(T_i)$. Note that the documents in a topic are the union of documents that belong to its child topics, i.e., $D(T_i) = \cup_{p(T_j)=T_i} D(T_j)$.

Let $X \in \mathbb{R}_+^{m \times n}$ be the data matrix of topic T, where m is the number of words in the corpus and n is the number of documents that belong to the topic. The p-th column of X represents the bag-of-words representation of document d_p with respect to m keywords. A standard Nonnegative Matrix Factorization (NMF) approach solves a low-rank approximation as follows:

$$\min_{\{W,H\} \geq 0} ||X - WH||_F^2, \tag{3.1}$$

where $W \in \mathbb{R}_+^{m \times k}$ and $H \in \mathbb{R}_+^{k \times n}$ are factor matrices and k is the number of child topics under T. W describes topics and H describes document-topic memberships. The p-th child topic under T is calculated as w_p, the p-th column of W. High values in w_p indicate that the corresponding words are strongly associated with the p-th child topic. Next, the q-th column of H, h_q, represents document d_q as a weighted combination of k topics. We say document d_q belongs to the p-th child topic if the p-th element of h_q is its largest element, i.e., $p = argmax(h_q)$. Notations are summarized in Table 5.1.

In order to steer topics, we modify the NMF formulation as follows:

$$\min_{\{W,H,U\} \geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2 + \beta||M_H \circ (E - HU)||_F^2 \tag{3.2}$$

The second term influences topics' keyword descriptions by forcing W to be similar to W', which represents topic keywords selected by the users. The third term affects topic-document memberships by forcing H to be similar to E, which represents topic-document memberships assigned by the users, with the help of a scaling matrix U. Parameters α and β determines the amount of user control for word-level and document-level interactions, respectively. When $\alpha = 0$ (or $\beta = 0$), the word- (or document-) level interaction is not reflected into the model. Larger α, β leads to stronger incorporation of user steering, but it may result in less truthful representation of the underlying data. α, β are set to be proportional to the number of interacted topics by word-level and document-level tasks, respectively. $M_W,\ M_H$ are masking matrices where $(M_W)_{\cdot r} = 0$ and $(M_H)_{r\cdot} = 0$ for $r \notin \{\text{steered indexes}\}$ and $(M_W)_{\cdot r} = 1$ and $(M_H)_{r\cdot} = 1$ for $r \in \{\text{steered indexes}\}$. More detail will be described in Section 3.3.4.

3.3.4 Algorithm

Base Operations

In this section, we describe the algorithms for our base operations based on which our interaction tasks can be composed.

$[T_1, T_2] =$ **makechildren**(T_0)

---

makechildren applies a rank-2 NMF algorithm to the documents in a topic node T_0 to create two of its children nodes T_1, T_2.

Solve $\min_{\{W^{(0)},H^{(0)}\}\geq 0} ||X^{(0)} - W^{(0)}H^{(0)}||_F^2$ where $W^{(0)} \in \mathbb{R}_+^{m\times 2},\ H^{(0)} \in \mathbb{R}_+^{2\times n_0}$

$D(T_k) = \{d_q \in D(T_0) | argmax(H_{\cdot q}^{(0)}) = k\}$, $p(T_k) = T_0$ for $k = 1, 2$

$T_0 =$ **merge**(T_1, T_2)

---

merge creates a new parent T_0, which is the union of selected sibling topic nodes T_1, T_2, under their original parent $p(T_1)$.

$p(T_0) = p(T_1)$ and $D(T_0) = D(T_1) \cup D(T_2)$

$p(T_k) = p(T_0)$ for $k = 1, 2$

---

insert(T_1, T_2)

insert adds a topic T_1 under the selected node T_2 and updates its ancestors.

$p(T_1) = T_2$ and $D(T_2) = D(T_1) \cup D(T_2)$

$parent = p(T_2)$

while parent is not the top node **do**

$D(parent) = D(parent) \cup D(T_1)$

$parent = p(parent)$

end while

---

cut(T_1)

cut removes a topic T_1 from its ancestors.

$parent = p(T_1)$

$p(T_1) = null$ and $D(p(T_1)) = D(p(T_1)) \backslash D(T_1)$

while parent is not the top node **do**

$parent = p(parent)$

$D(parent) = D(parent) \backslash D(T_1)$

end while

$[T'_{01}, \cdots, T'_{0k_0}]$ = **recompute**$(T_0, (W'), (H'))$

recompute applies a flat NMF algorithm with constraints on a topic T_0 to re-partition its children $T'_{01}, \cdots, T'_{0k_0}$. The second and third terms incorporate word-level and document-level supervision, respectively.

Solve $\min_{\{W^{(0)}, H^{(0)}, U^{(0)}\} \geq 0} ||X^{(0)} - W^{(0)}H^{(0)}||_F^2 + \alpha||M_W^{(0)} \circ (W^{(0)} - W'^{(0)})||_F^2 + \beta||M_H^{(0)} \circ (E^{(0)} - H^{(0)}U^{(0)})||_F^2$ where $W^{(0)} \in \mathbb{R}_+^{m \times k_0}$, $H^{(0)} \in \mathbb{R}_+^{k_0 \times n_0}$, $U^{(0)} \in \mathbb{R}_+^{k_0 \times k_0}$

$D(T_k) = \{d_q \in D(T_0) | argmax(H_{q\cdot}^{(0)}) = k\}$ for $k = 1, \cdots, k_0$

When performing **recompute** on a parent topic before **cut** on a child topic, we re-distribute documents that are not strongly relevant to the child topic (i.e., $max(H_{kq}) < threshold$) into their sibling topics. As a result, when moving or removing topics, key-

words, or documents, only the documents that are strongly relevant to the moved or removed topics, keywords, or documents are cut out.

Hierarchical Topic Initialization

The proposed system generates the initial hierarchical topics where the upper level topics are more general and larger, and the lower level topics are more specific, finer-grained, and more tightly related. We adopted a hierarchical topic modeling algorithm called HierNMF2 [52, 53], which uses a fast rank-2 NMF [7] and a binary tree splitting rule. In other words, we recursively split a topic by solving Eqn. 3.1 with $k = 2$ topics. By utilizing the simple computation to obtain a rank-2 NMF, some very substantial speedups have been achieved for computing topic modeling results, which can be highly beneficial for achieving real-time interaction.

Hierarchical Topic Revision

After the initial hierarchical topics are computed, users can steer the model by performing various tasks described in Tables 3.3-3.5. We grouped user tasks by interaction unit types: topics, words, and documents. Note that previous hierarchical topic modeling systems allow topic reorganization through some topic-level interactions, but they offer none to highly limited word-level or document-level topic steering.

Topic-level Tasks

Topic-level tasks in existing hierarchical topic modeling systems affect all documents in the interacted topic. For example, moving a topic would relocate all the associated documents into a new parent topic. However, the decisions to do so are based on limited information shown on the screen to the users. Thus, our approach does not move all documents of a topic when moving/removing topics, but rather moves only a strongly relevant subset using constrained NMF. Now we define topic-level tasks as follows:

$T_0 = \textbf{MergeT}(T_1, T_2)$

MergeT combines selected topics T_1 and T_2 to create a new parent topic T_0. If T_1 and T_2 are not siblings, T_1 is moved under T_2's parent before merging.

if $p(T_1) \neq p(T_2)$ **then**

  MoveT$(T_1, p(T_2))$

end if

$T_0 = \textbf{merge}(T_1, T_2)$

$[T_1, T_2] = \textbf{SplitT}(T_0)$

SplitT partitions topic T_0 into two child topics T_1 and T_2 using rank-2 NMF.

$[T_1, T_2] = \textbf{makechildren}(T_0)$

MoveT(T_1, T_2)

MoveT detaches a topic T_1 and attaches it under a new parent T_2. Before detaching T_1, we redistribute less relevant documents in T_1 into its sibling topics to move only a strongly relevant subset. After attaching, we solve another NMF for the new parent T_2 to find more suitable child topics with the incoming topic T_1.

recompute$(p(T_1), w_1)$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2$ where $W'_{\cdot 1} = w_1$.

cut(T_1)

insert(T_1, T_2)

recompute$(p(T_2))$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2$

RemoveT(T_1)

RemoveT discards the most relevant documents of the selected topic T_1 into the trash C rather than all documents in T_1. The remaining less-relevant documents are redistributed to its sibling topics.

recompute$(p(T_1), w_1)$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2$ where $W'_{\cdot 1} = w_1$.

cut(T_1)

insert(T_1, C)

RestoreT(T_1, T_2)

RestoreT moves a previously deleted topic T_1 from the trash C into a selected parent topic T_2.

cut(T_1)

insert(T_1, T_2)

recompute$(p(T_2))$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2$

Fix(T_1)

Fix freezes a topic T_1 so that it will not be changed in later stages. It can be used when the quality of a current topic is determined good and the user wants to consider the topic as final. The **Fix** task does not involve any computation but marks the topic as fixed so that the topic is not modified in any subsequent computation or interaction.

Word-level Tasks

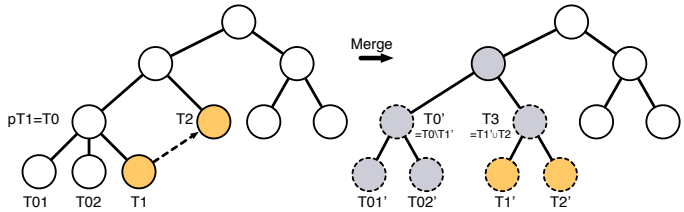
Word-level interactions influence keyword descriptions of topics so that a specific topic becomes more (or less) about certain words. As a result, some documents are redistributed according to the new topic descriptions. Previously, word-level interactions were only supported in some flat topic model systems but not in hierarchical systems. To our knowledge, ArchiText is the first to allow word-level refinement of hierarchical topics. For simplicity

Table 3.3: Topic-level User Interactions Supported by ArchiText.

Topic-level Operated Affected Changed

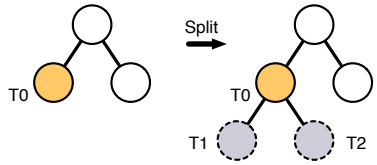
Merge two topics T_1 and T_2

MergeT(T_1, T_2) = **MoveT**(T_1, T_2) + $merge(T_1, T_2)$



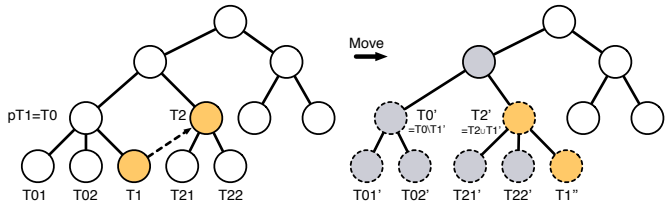
Split a topic T_0 into child topics

SplitT(T_0) = $makechildren(T_0)$



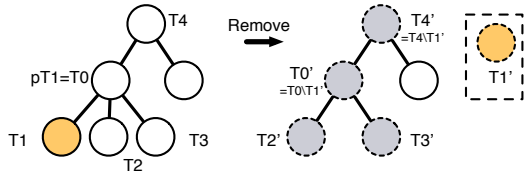
Move a topic T_1 into a new parent T_2

MoveT(T_1, T_2) = $recompute(p(T_1), w_1) + cut(T_1) + insert(T_1, T_2) + recompute(T_2)$



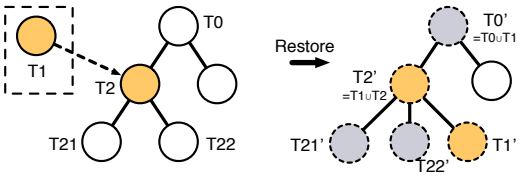
Remove a topic T_1

RemoveT(T_1) = $recompute(p(T_1), w_1) + cut(T_1) + insert(T_1, C)$



Restore a topic T_1 under T_2 from trash

RestoreT(T_1, T_2) = $cut(T_1) + insert(T_1, T_2) + recompute(T_2)$



Fix a topic T_1

FixT(T_1): makes T_1 unaffected by $recompute(p(T_1))$

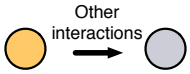
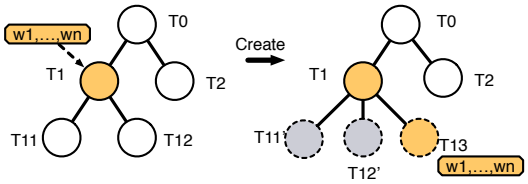


Table 3.4: Word-level User Interactions Supported by ArchiText.

Word-level Operated Affected Changed

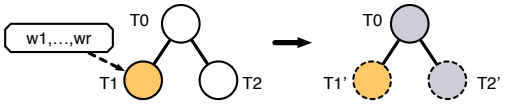
Create a topic T_w by seed words w under a parent T_1

CreateTW$(w, T_1) = recompute(T_1, w)$



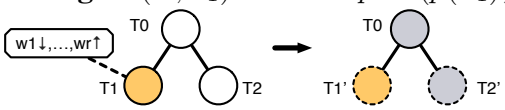
Add words w to a topic T_1

AddW$(w, T_1) = recompute(p(T_1), w)$



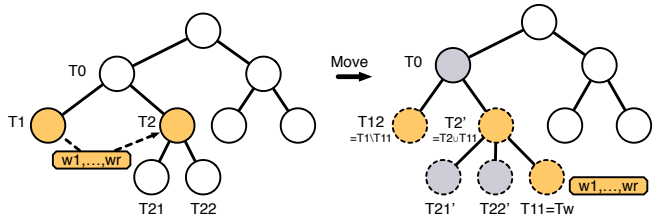
Change word distribution of a topic T_1

ChangeW$(w, T_1) = recompute(p(T_1), w)$



Move words w from a topic T_1 to another T_2

MoveW$(w \in T_1, T_2) = makechildren(T_1) + recompute(T_1, w) + cut(T_w) + insert(T_w, T_2) + recompute(T_2)$



Remove words w from a topic T_{11}

RemoveW$(w, T_{11}) = makechildren(T_{11}) + recompute(T_{11}, w) + cut(T_w) + insert(T_w, C)$

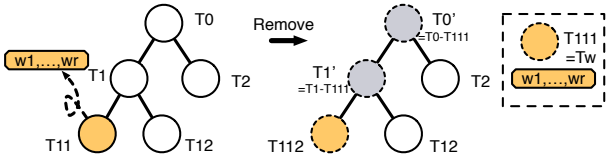
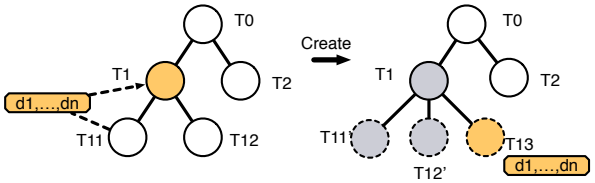


Table 3.5: Document-level User Interactions Supported by ArchiText.

Document-level Operated Affected Changed

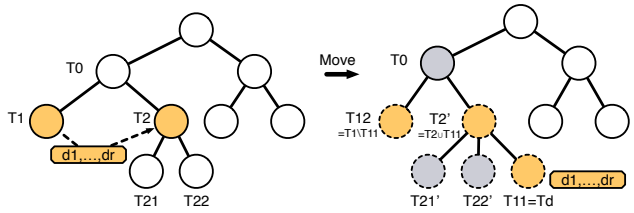
Create a topic T_d by seed docs d under a parent T_1

CreateTD$(d \in T_1) = recompute(T_1, d)$



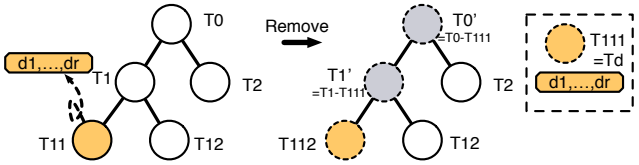
Move documents d from a topic T_1 to another T_2

MoveD$(d \in D(T_1), T_2) = makechildren(T_1) + recompute(T_1, d) + cut(T_d) + insert(T_d, T_2) + recompute(T_2)$



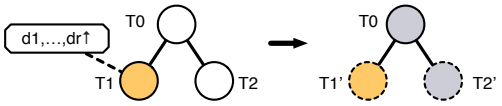
Remove documents d from a topic T_{11}

RemoveD$(d, T_{11}) = makechildren(T_{11}) + recompute(T_{11}, d) + cut(T_d) + insert(T_d, C)$



Like a document d for a topic T_1

LikeD$(d \in T_1) = recompute(p(T_1), d)$



and efficiency, we limit the scope of affected topics to one level. We define word-level tasks as follows:

CreateTW(w, T_1)

CreateTW creates a new topic with seed words w under a parent topic T_1.

$k_1 = k_1 + 1$

recompute$(p(T_1), w)$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2$ where $W'_{\cdot 1} = w$.

AddW(w, T_1)

AddW adds new terms w to a topic T_1 to steer it toward the selected words.

recompute$(p(T_1), w)$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2$ where $W'_{\cdot 1} = w$.

ChangeW(w, T_1)

ChangeW changes the word distribution w of a topic T_1 to steer the topic based on the re-weighted words..

recompute$(p(T_1), w)$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2$ where $W'_{\cdot 1} = w$.

MoveW$(w \in T_1, T_2)$

MoveW aims to subtract the selected terms w from a topic T_1 and add them into another topic T_2 by moving the most relevant documents in the topic.

$[T_w, T_{\bar{w}}] =$ **makechildren**(T_1) with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2$ where $W'_{\cdot 1} = w$.

MoveT(T_w, T_2)

RemoveW(w, T_1)

RemoveW discards the documents in a topic T_1 that are most relevant to the selected words w. The remaining less relevant documents are redistributed to sibling topics.

$[T_w, T_{\bar{w}}] =$ **makechildren**(T_1) with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \alpha||M_W \circ (W - W')||_F^2$ where $W'_{\cdot 1} = w$.

RemoveT(T_w)

Document-level Tasks

Document-level interactions influence document-topic memberships to steer a topic to be similar (or dissimilar) to the selected documents. As a result, keyword descriptions of affected topics can change accordingly. Following the tight integration principles, our document-level tasks not only involve the selected documents, but also affect documents that are similar or relevant to the selected documents. We define document-level tasks as follows:

CreateTD(d, T_1)

CreateTD creates a new topic with seed document d under a parent topic T_1.

$k_1 = k_1 + 1$

recompute$(p(T_1), d)$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \beta||M_H \circ (E - HU)||_F^2$ where $E_{1d} = 1$.

MoveD$(d \in T_1, T_2)$

MoveD aims to subtract the selected documents d (and similar ones) from a topic T_1 and add them into another topic T_2.

$[T_d, T_{\bar{d}}] =$ **makechildren**(T_1) with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \beta||M_H \circ (E - HU)||_F^2$ where $E_{1d} = 1$.

MoveT(T_d, T_2)

RemoveD(d, T_1)

RemoveD discards the selected documents d and the similar ones from a topic T_1. The remaining less-relevant documents are redistributed to sibling topics.

$[T_d, T_{\bar{d}}] =$ **makechildren**(T_1) with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \beta||M_H \circ (E - HU)||_F^2$ where $E_{1d} = 1$.

RemoveT(T_d)

LikeD$(d \in T_1)$

LikeD steers a topic T_1 to be more like the liked document d.

recompute$(p(T_1))$ with $\min_{\{W,H\}\geq 0} ||X - WH||_F^2 + \beta||M_H \circ (E - HU)||_F^2$ where $E_{1d} = 1$.

3.4 System

In this section, we describe ArchiText, our prototype visual analytics system for interactive hierarchical topic modeling with tight integration. The proposed system is built using the D3.js visualization library, Flask framework, sqlite database, and a fast rank-2 nonnegative matrix factorization algorithm and the proposed constrained low rank approximation shown in Eqn. 3.2 written in Matlab™.

3.4.1 System Design

Our system has two modes: a topic workspace mode (Fig. 3.1) and a hierarchy view mode (Figs. 3.2-3.3). The topic workspace mode is designed to facilitate flexible user interactions for tuning and interacting with the hierarchical topic representation. The hierarchy view mode is primarily for inspecting the overall structure of the computed topic tree. Users can alternate between the two modes by clicking the blue button on the top right corner shown in Fig. 3.1a.

Figure 3.2: The hierarchy view mode presents a high-level overview of the topic tree with a control panel. A topic can be expanded (or collapsed) to show (or hide) its child topics. Expandable topics are represented as filled circles.

Topic Workspace Mode contains the main topic card view, a control bar, a breadcrumb view, and a mini overview.

The main topic card view (Fig. 3.1c) visualizes topics up to selected depths in the computed hierarchical topic tree. Each topic is visualized as an indented equal-width card where the height of each card is proportional to the number of documents that belong to the topic. Each topic's most representative keywords and their importance weights are visualized as a sorted list with bars. This design allows users to quickly understand topics well [54] and easily compare keyword weights across different topics. Note that all keywords are in stemmed forms as we use the porter stemmer during the data preprocessing step. On the top right corner of a topic card are the menu button and the flip button. A topic card can be flipped to show its documents (Fig. 3.1d) with their 'Thumbs up' buttons to like the corresponding documents. Hovering the mouse over a document shows its detailed information such as document id, title, authors, etc (Fig. 3.1e). Topic coherence scores are visualized as bars on top of the topic cards. As a topic coherence metric, we use pointwise

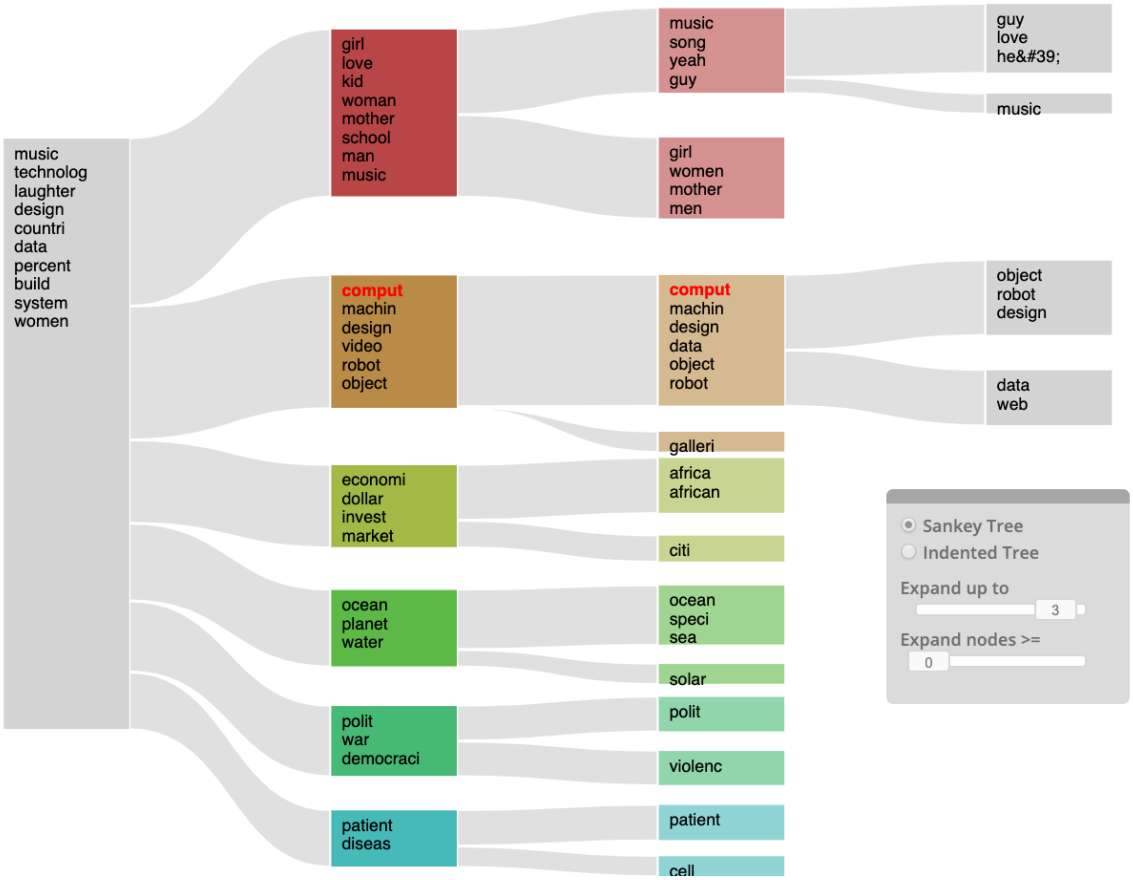


Figure 3.3: An alternate Sankey tree visualization in the hierarchy view mode. Placing the mouse over a keyword highlights the keyword in other topics.

mutual information (PMI), which is highly correlated with human judgement [55]. This can guide users to focus on refining and improving low-quality topics and observe how their interactions affect topic quality by monitoring the bars. Topic cards, keywords, and documents can be interacted with to steer the underlying topics and their hierarchy, which will be described in Section 3.4.2 in detail.

The control bar contains buttons to update the main topic card view. A sliding trash panel is toggled by clicking a trash button in the control bar (Fig. 3.1a). The plus and minus buttons in the control bar (Fig. 3.1a) change the visualized depth of the topic tree to support multi-level exploration. Users can 'drill down/zoom in' on a topic for more details and 'zoom out' to see higher-level topics as indented topic cards, respectively. For example, Fig. 3.4a shows the first level topics. By zooming in, Fig. 3.4b shows the first

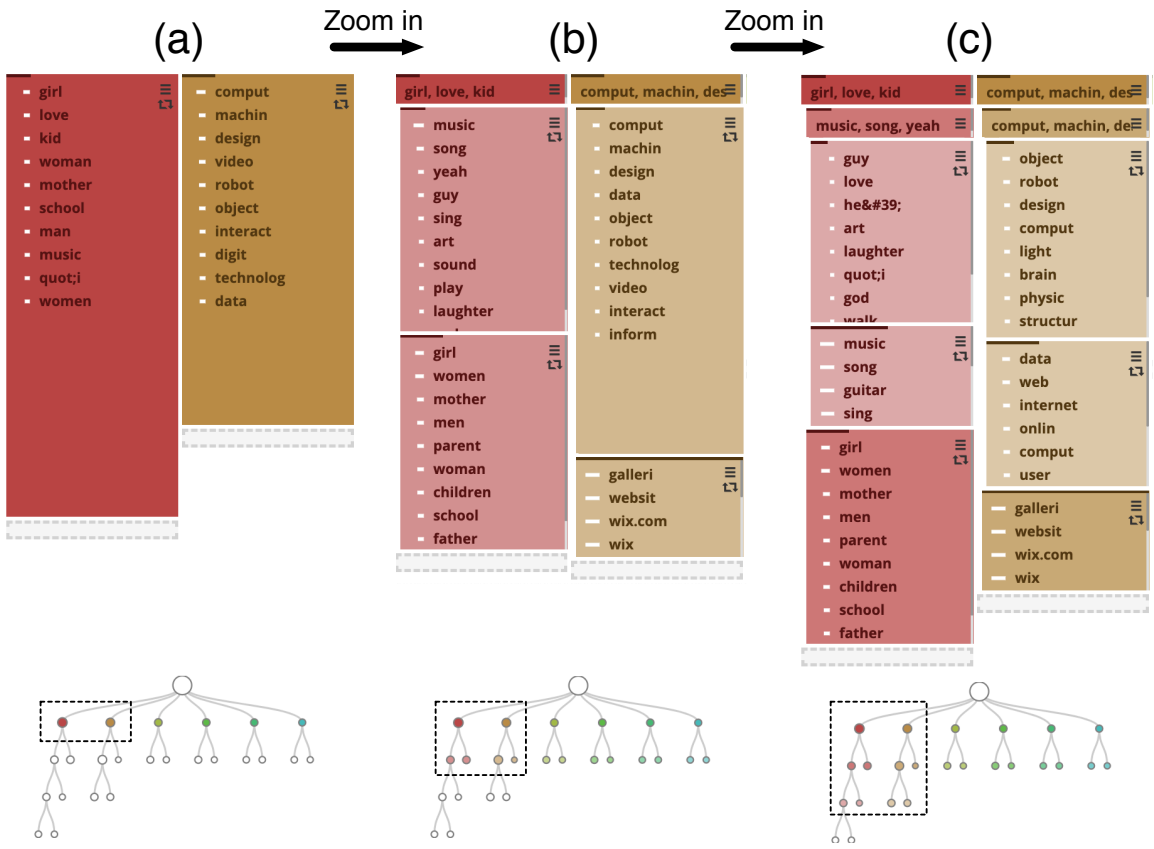


Figure 3.4: Zooming-in from (a) to (c) by clicking the + button reveals deeper levels of hierarchical topics interactively.

level and the second level topics where the deepest visible topic card is shown and the rest are collapsed. By zooming in again, Fig. 3.4c shows topics up to the third level. Note that parent and children topics have the same color hue, but with different saturation; lighter colors and longer indentations represent deeper node depths.

The mini overview displays the overall topic hierarchy as either a weighted tree (Fig. 3.1f) or an icicle visualization. Topics visualized as topic cards are colored accordingly and the remaining topics with deeper depths are colored gray. When hovering over a topic, the breadcrumb view (Fig. 3.1b) shows the trail from the top node to the current node (orange-colored topic in Fig. 3.1c), and the corresponding node in the mini overview is highlighted with black solid line (orange-colored circle in Fig. 3.1f).

Hierarchy View Mode offers two types of tree visualizations, an indented tree (Fig. 3.2)

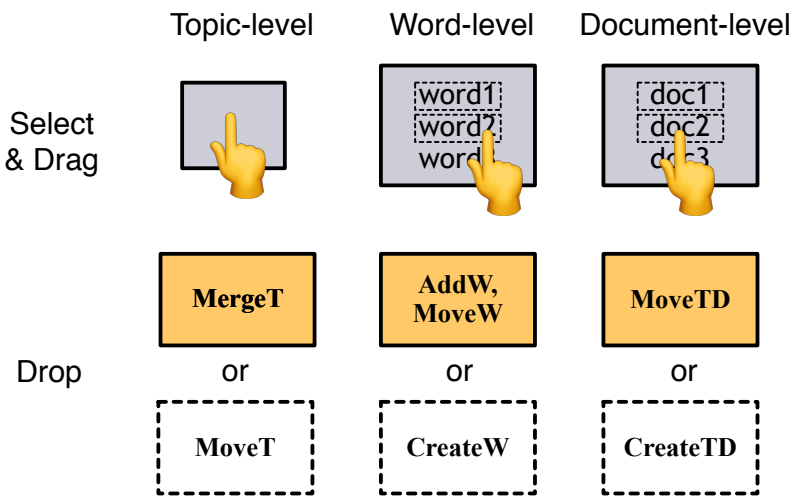


Figure 3.5: User interaction design for supported tasks.

and a Sankey tree (Fig. 3.3), which can be selected in the control panel (Fig. 3.2a). In both views, topic colors correspond to those in the workspace mode. Using the control panel (Fig. 3.2a), users can collapse topics by their depths or sizes. The indented tree view (Fig. 3.2) visualizes the topics and the hierarchy as an indented tree where indentation reflects tree depth. For each topic, its ID, topic size, and top ten keywords are shown. Clicking a topic's circle collapses/expands the topic to hide/show its children topics. Filled circles represent collapsed (and thus expandable) topics. The Sankey tree view (Fig. 3.3) visualizes the topic tree from left (top node) to right (deeper level node) where node height reflects topic size (number of documents in a topic). Each topic node displays up to ten keywords depending on its size. Hovering over a topic node pops up a detail view showing all ten keywords and the size of a topic. When hovering over a keyword in a topic node, the same keywords in other topics are highlighted to show term patterns (red keywords in Fig. 3.3).

3.4.2 User Interaction Design

Fig. 3.5 demonstrates supported interaction tasks to steer the underlying hierarchical topics. All interactions are designed to be executed by clicking buttons (**SplitT**, **FixT**, **LikeD**) or simply dragging and dropping visual components such as words, documents, and topic

cards as follows. When modifying existing topics (**MergeT**, **AddW**, **MoveW**, **MoveD**), drop recipients are the topic cards being modified. When creating a new topic (**CreateTW**, **CreateTD**) or moving a topic (**MoveT**, **RestoreT**), the drop recipient is a dotted space which represents a temporary topic card. When deleting words (**RemoveW**), documents (**RemoveD**), or a topic (**RemoveT**), the drop recipient is the trash button.

Interaction Assistant

In the proposed tight integration framework, users incrementally update models through a wide variety of interactions. During the process, our system guides users by predicting and recommending interaction tasks. Many of the interactions in our system start with selecting and dragging and end with dropping. Our interaction assistant is triggered when the user selects or drags something and then it predicts the next step to complete the interaction. This can be beneficial for users who are exploring potential alternatives for how to organize and construct their topics.

Multi-selection: When steering a topic using word-level (**CreateTW, AddW, MoveW, RemoveW**) or document-level (**CreateTD, MoveD, RemoveD**) interaction tasks, selecting multiple words or documents can convey clearer meaning than selecting a single word or a document. However, going over many words or documents can be time-consuming. To foster efficient multi-selection, our interaction assistant visually recommends selection candidates. When a word (or a document) is first selected, the system highlights frequently co-occurring words (or similar documents) in the same topic to be selected along with the first selected word (or document).

Drop: After selecting the words, documents, or a topic, the next step is to drag and drop them into another topic or into the trash. Our interaction assistant predicts and recommends the locations to drop them during an interaction. When the user starts dragging a topic,

the system highlights similar topics as drop recipients to foster **MergeT** or **MoveT** tasks. Similarly, when the user starts dragging words or documents, the system highlights topics that are similar to the dragged words or documents as drop recipients to foster **AddW, MoveW, MoveD** or **CreateTW, CreateTD** tasks. If the selected words or documents are not coherent (not similar to each other), the system highlights the trash to foster **RemoveW, RemoveD** tasks. In addition, users can preview the expected hierarchy in the mini overview while placing the mouse over the drop recipients.

3.5 Experiments

In this section, we present a quantitative evaluation to show the scalability of our approach. For our experiments, we use a patent dataset[1]. This dataset contains about 7 million granted patents and their information, e.g., ID, type, title, abstract, year, etc. After filtering out non-utility patents, we are left with 6,248,456 utility patents.

We report computation time using different sizes of patent subsets. We select 10,000, 50,000, 100,000, 500,000, and 1,000,000 data items from the patent dataset to create multiple subsets of different sizes. For each subset, we report the running time of building the initial hierarchical topics as well as performing an interaction task on a topic that contains about 10% of the documents. Experiments were performed on a MacBook Pro with Intel Core i7 3GHz, 4 cores, 8GB memory. Table 3.6 shows that building an initial topic model with ten leaf nodes from a million documents takes about 5 minutes. Also, most tasks are finished within several seconds, supporting accurate and timely visualization in our tight integration methodology as discussed in Section 3.3.1.

In general, interactions with recomputations such as moving take longer computation time than interactions without recomputation such as merging (siblings) or splitting. This is because recomputation on a topic runs a flat NMF algorithm with constraints (Eqn. 3.2) on its parent topic, and the changes are propagated to its descendants. To reduce interaction

[1] http://www.patentsview.org/download/, March 12, 2019 Version

Table 3.6: Computation times (in seconds) for hierarchy initialization with ten leaf nodes and several interaction tasks (merging siblings, splitting a topic, moving a topic–with and without recomputation). All results are averaged over 10 runs.

Datasets	p10K	p50K	p100K	p500K	p1M
# Documents	10,000	49,995	99,989	499,968	999,941
# Words	6,585	15,414	22,702	60,131	92,966
Initialization	3.64	13.73	27.35	142.19	300.01
Merge	0.005	0.014	0.025	0.097	0.191
Split	0.066	0.138	0.308	1.164	7.009
Move	0.598	1.515	2.927	22.133	49.735
Move (w/o **re**)	0.023	0.036	0.050	0.186	0.356

latency caused by recomputation time, we suggest the following strategies. First, we could reduce the number of recomputation. Instead of performing full recomputation every time, the system can decide when to skip or perform recomputation. For instance, in the case of **recompute** before **cut** or after **insert**, recomputation can be skipped unless a large portion of the interacted topic is changed or a certain number of interaction tasks have been applied to the interacted topic without recomputation. Second, we could limit the number of iterations when solving Eqn. 3.2. Since our algorithm utilizes previous topics to initialize factor matrices W, H, we could reduce the number of total iterations per one recomputation and still reach a near optimal solution. Next, we can recommend users to keep the size of interaction small since recomputation time depends on the size of its parent topic. That is, focus on splitting topics into smaller ones rather than directly steering bigger topics. Because our recomputation is local, it only affects siblings. Regardless of the size of the entire dataset, users can use this strategy to achieve fast interaction.

In this section, we do not report topic quality measures as those depend on user decisions of which topics to interact with. For instance, merging any two random topics would degrade the overall quality of the topic hierarchy. Instead, we show two use cases that showcase the effectiveness of our tight integration approach in Section 3.6.

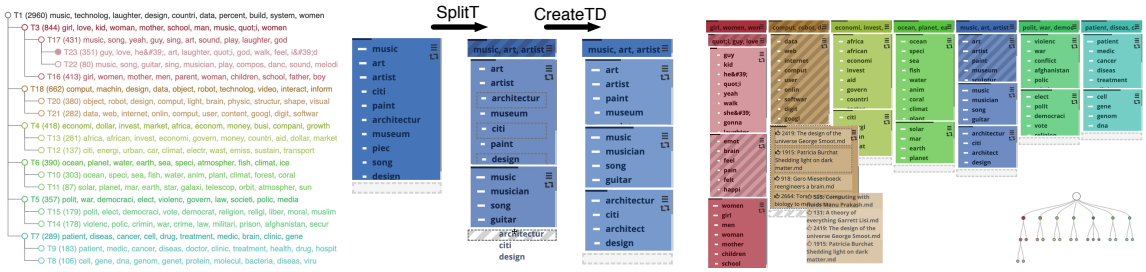


Figure 3.6: The topic hierarchy after removing T19 (left). Split a topic into three sub-topics (middle). Create sibling topic with documents (right).

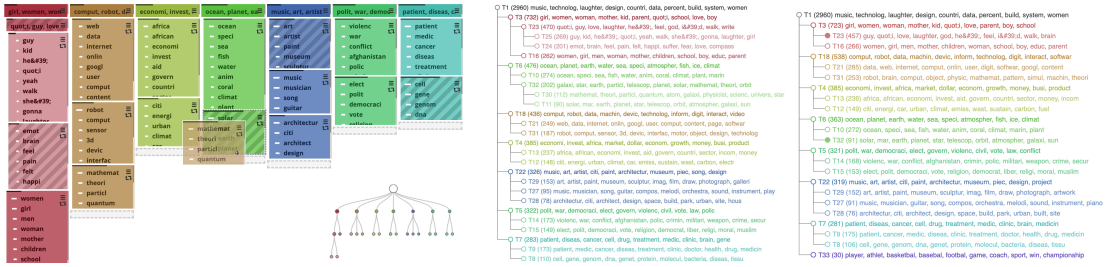


Figure 3.7: Merging two topics. Interaction assistant recommends which topic to merge into (left). The topic hierarchy after merging (middle). The final topic hierarchy (right).

3.6 Use Case

3.6.1 TED Transcript Dataset

TED is a nonprofit organization that hosts influencial talks and shares the videos online. Various topics including technology, education, and self-help are covered in TED talks. Although the official TED website provides keyword search functionality and over 400 category tags, navigating about 3,000 talks and discovering talks of interest is not easy. In this section, we use ArchiText to understand main themes of the talks and organize them into hierarchical categories for easier navigation. We used the TED dataset containing 2,969 talk transcripts.[2]

A user starts by inspecting six top-level topics shown in the initial topic hierarchy (Fig. 3.2). In Fig. 3.2, there are clear and coherent topics like 'ocean, planet, water, earth, sea' (T6: limegreen) and 'patient, disease, cancer, cell, drug' (T7: turquoise). On the other hand, the red topic (T3) with 'girl, love, kid, woman, mother' keywords is more general

[2]Source: https://github.com/saranyan/TED-Talks

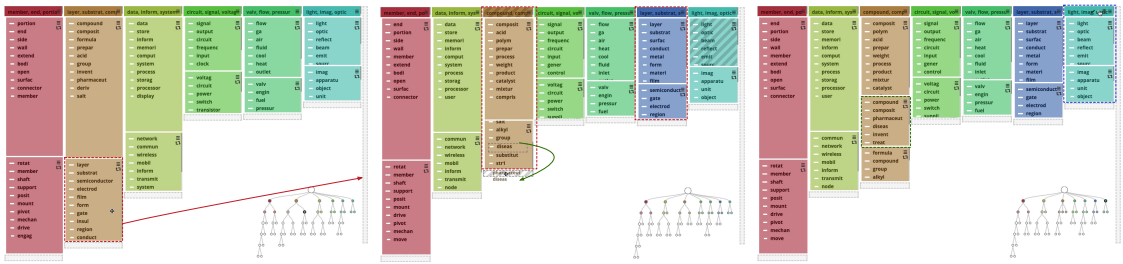


Figure 3.8: Initial topic hierarchy of patents (left). After moving a topic (middle). After moving keywords (right).

and ambiguous. The user zooms in to see child topics (Fig. 3.4). There is a strange topic with keyword 'galleri websit, wix.com, wix' (T19). Upon inspecting its documents, she notices that their contents are actually the same transcript of an advertisement. It turns out that the used web scrapper transcribed youtube commercials instead of the main talk video. She deletes the topic (**RemoveT**). The user continues exploring the unclear red topic by examining its child topics (Fig. 3.6(left)). She thinks that a red child topic with 'music, song, sing' keywords should be one of the top level topics, so she moves it under the top node (**MoveT**). As a result, there is a top-level topic on 'music, art, artist' in Fig. 3.6 (middle). The user further splits the blue art topic (**SplitT**). One of its child topics contains both art-related keywords and architecture-related keywords. She moves 'architecture, city, design' keywords to create a sibling topic (**CreateTW**). As a result, the art topic has three sub-topics on art, architecture, and music (Fig. 3.6 (middle)). Satisfied with the blue topic, she moves on to the brown topic, which is the second largest. The brown topics are mostly about computer technologies, but she notices that a few keywords 'mathemat, physic, simul, theori' are about natural sciences. She flips the topic and starts to drag several documents about physics and simulation with the goal of separating those out. While dragging, the interaction assistant recommends candidate drop zones with line patterns (Fig. 3.6 (right)). She decides to move the documents under the top brown topic to create a sibling topic (**CreateTD**). As a result, a new sub-topic about 'mathemat, theori, particl, quantum, galaxi' is created (Fig. 3.7 (left)). She merges the new topic with another natural science topic on 'solar, mar, earth' (lime green topic in Fig. 3.7 (left)). She

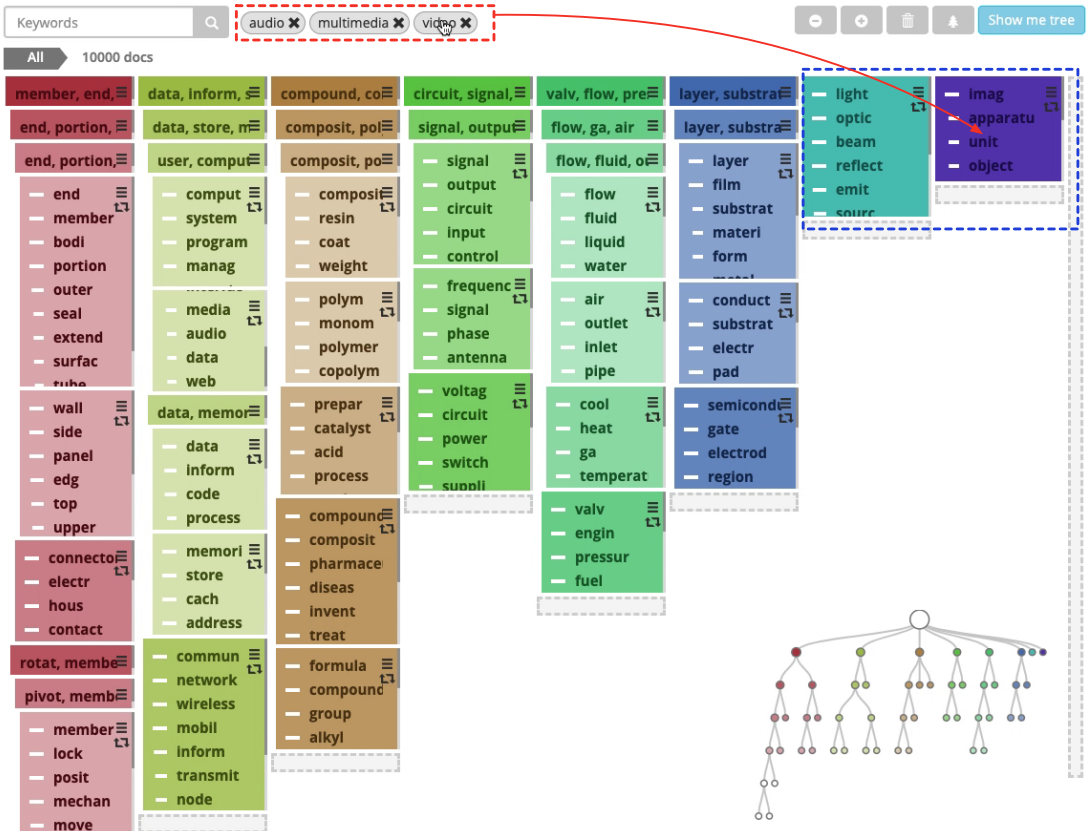


Figure 3.9: After splitting a topic into two topics (blue box). Add keywords into the purple topic.

is interested in sports, but she has not seen sports-related topics so far. She types in sports-related keywords such as sports, athletes, basketball, tennis, etc. to create a new topic (**CreateTW**). The new topic has a very small number of documents (30 talks in Fig. 3.7 (right)). She goes over each document and finds out that some talks are science/tech related (e.g., math behind basketball) or inspirational talks from athletes (e.g. Billie Jean King). She concludes that there are not many TED talks on the topic of sports and finishes the analysis.

3.6.2 Patent Dataset

The Patent office manages historical patents; and granted or rejected patent applications. Due to technology advances in various fields, there is a need to update the patent classification system. Using ArchiText, we will use interactive topic modeling to make sense

Figure 3.10: The final topic hierarchy.

of existing utility patents and build a new taxonomy based on their content. The dataset description can be found in Section 3.5.

A patent officer explores the initial topic hierarchy in Fig. 3.8. He notices that a brown topic about semiconductor process (red box in Fig. 3.8 (left)) is grouped with a brown chemistry topic. He moves the semiconductor topic under the top node (**MoveT**). As a result, two topics are separated (red boxes in Fig. 3.8 (middle)). He inspects one of the chemistry sub-topics and decides to move words (**MoveW**) about pharmaceutical patents (green arrow in Fig. 3.8 (middle)), which results in a new topic about 'pharmaceut, diseas, treat, treatment' compounds under the chemistry parent topic. Moving on to the rest of the topics, he splits a 'light, image' topic (blue box in Fig. 3.8 (right)) into an optics topic and an image-related topic (**SplitT**). He delves into sub-topics of all topics, and finds a media related topic under the green 'data, inform' topic (second from the left). Wanting to gather

all media related patents under a single top-level topic, he adds keywords 'audio, multimedia, video' into the purple image-related topic (**AddW**) while fixing other topics (**FixT**). As a result, the purple topic becomes larger and contains more media-related patents, some of which are moved from the yellowgreen topic (T13 in Fig. 3.10). He is now satisfied with the hierarchy and creates the new taxonomy based on the result.

3.7 Discussion

Interactive topic modeling systems, in order to steer the underlying models, users provide supervision in terms of user interaction. For the same user interaction, there are numerous ways to interpret intermediate results to understand the algorithmic updates depending on the visualization systems and their underlying algorithms. Two basic common factors that can be applied to all interactive topic modeling techniques are the *scope* and the *amount* of user control. First, *scope* determines how wide the impact of the interaction would be. For instance, when a user adds a document to a topic, we can safely assume that the user wants to update the interacted topic. Should this interaction affect only the interacted topic? Or should it also affect the neighboring topics? Or all the topics? Updating in local scope can be faster with less precise results. On the other hand, updating in global scope can provide more accurate results, but may cause unexpected changes in other parts of the model. Next, the *amount* of user control determines to what degree to apply the supervision. For example, when a user adds a keyword to a topic, the user expects the interacted topic to be (more) about the added keyword. In this case, should the updated topic have that keyword as its top keyword at any cost (hard supervision)? Or is increasing the importance weight of that keyword for the topic enough (soft supervision)? What if the topic and its corresponding documents are not related to that keyword, e.g., adding an irrelevant keyword? Some may prefer applying the hard supervision while others may argue for a more truthful representation of the data. In order to balance these trade-offs, we take a simple approach. Our system uses **recompute** operations to supervise the underlying model. **recompute** solves a con-

strained Nonegative Matrix Factorization (NMF) for the sibling topic nodes (local scope) of the interacted topic with two parameters α and β (Eqn. 3.2). We considered an option to let the users decide the amount of user control during each interaction, but decided against it. It can be burdensome to the users and it may slow down the analytic process.

3.8 Conclusion

In this chapter, we proposed interactive hierarchical topic modeling with tight integration among algorithm, visualization, and users interaction. Unlike some previous interactive systems which offer rather limited interaction functionality that may result in unexpected outcomes, our tightly integrated system incorporates user intentions flexibly without strange side effects. In addition, compared to existing interactive topic modeling systems that are not scalable, our system can handle large datasets. As a proof of concept, we developed ArchiText, a prototype system for interactive hierarchical topic modeling and showcased usage scenarios using real-world datasets.

For future work, we plan to take a more proactive approach for smart, convenient human-in-the-loop topic modeling. With tightly integrated topic modeling, users' knowledge and intentions can be flexibly incorporated step by step. In addition to this, we would like our system to remember and learn from previous interactions in order to predict and guide the users' next steps to expedite the model steering process.

CHAPTER 4

TOPICSIFTER: INTERACTIVE SEARCH SPACE REDUCTION THROUGH TARGETED TOPIC MODELING

In this chapter, we describe our interactive approach to solve large-scale document retrieval with high recall where any missed relevant documents can be critical. This chapter is adapted from our work that is presented at VAST 2019.

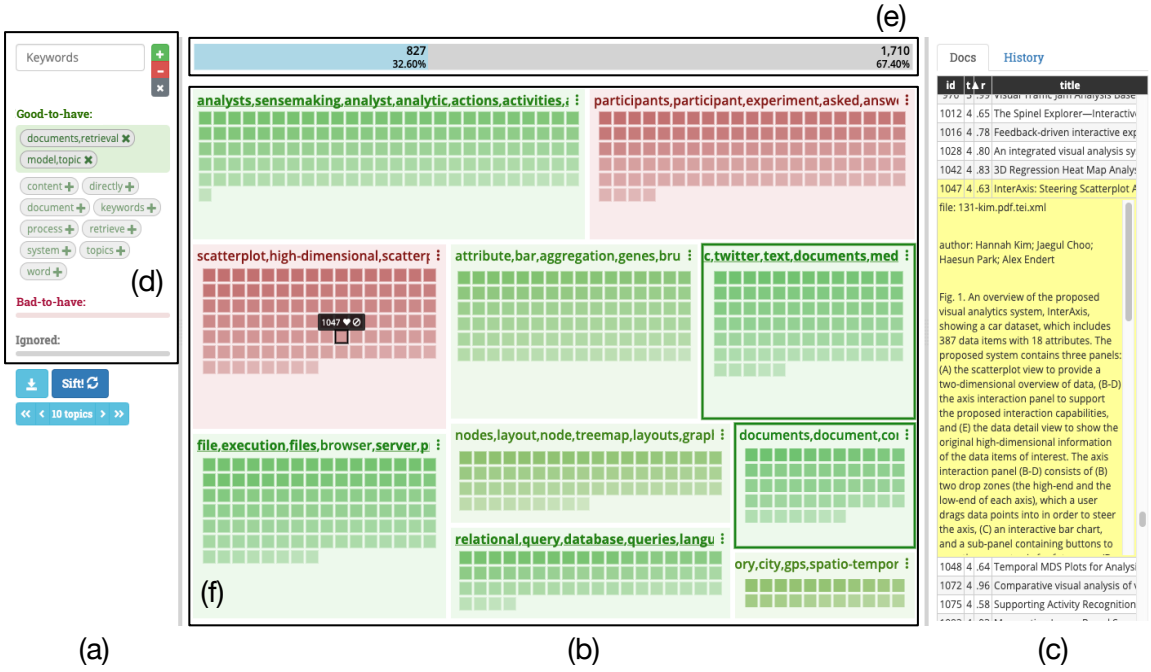


Figure 4.1: The TopicSifter system has (a) the control panel, (b) the main view, (c) the detail panel. The keyword module (d) in the control panel (a) shows the current set of good-to-have keywords, bad-to-have keywords, and stopwords and allows users to modify them. The system recommends additional keywords based on the current set of keywords. The main view (b) shows the sifting status bar (e) showing how many documents are retrieved from the total dataset and the topical overview (f) of current retrieved documents. The users can give positive or negative feedback on topics and documents to indicate relevancy. The detail panel (c) has two tab menus for showing document details and sifting history.

4.1 Introduction

As the world becomes increasingly digital and huge amounts of text data are generated every minute, it becomes more challenging to discover useful information from them for applications such as situational awareness, patient phenotype discovery, event detection [56], or the onset of violence within a diverse population. More often than not, topics of interests are only implicitly covered in vast amounts of text data and the relevant data items are sparse and not immediately obvious. This scenario is more prevalent especially in large scale data analytics where the data are obtained from passive sources and not all data items are relevant to the questions at hand. In these cases, users want to focus on a subset of documents about specific aspects or "targets", rather than analyzing entire document collections [57]. For example, a journalist may want to analyze social media data that are related to a specific event. Similarly, a marketing expert may want reviews that are relevant to certain products or brands only. Both examples require the *search space* of entire documents to be *reduced* to relevant documents.

Discovering and extracting data items of relevance from a large collection of documents is a challenging and important step in text analytics. In particular, we are interested in the *high recall* retrieval problem, where any missing relevant documents are critical [58]. For instance, a legal analyst searching for relevant cases from a large legal document collection may want to collect as many documents as possible even if some of them are only slightly relevant to her targets. Another example is a graduate student who is preparing a literature review and does not want to miss a related work. This is different from a traditional informational retrieval problem of finding a list of k results that are most relevant to a query, e.g., a student searching for the top 5 papers to learn about an unfamiliar research field. Our focus is on not missing relevant results in addition to high precision. To solve this, our goal is to retrieve documents that are relevant to targets from large scale document collections, which we will refer to as **search space reduction** throughout this chapter.

Traditional static keyword search is not suitable for our search space reduction setting. First, it is often difficult to know or express the target aspect in advance without exploring the dataset. Next, even when the users are familiar with their target concept, it is hard to cover all relevant keywords, which would result in false negative. Lastly, a keyword may have multiple unrelated meanings and when they are extracted out of context, static keyword match can result in false positive. More advanced approaches such as query expansion and relevance feedback have been introduced in information retrieval. These approaches expand query keywords and provide feedback on documents to update the query. However, since they are designed for high precision problems of retrieving a number of the most relevant data items, they may not cover all relevant data items.

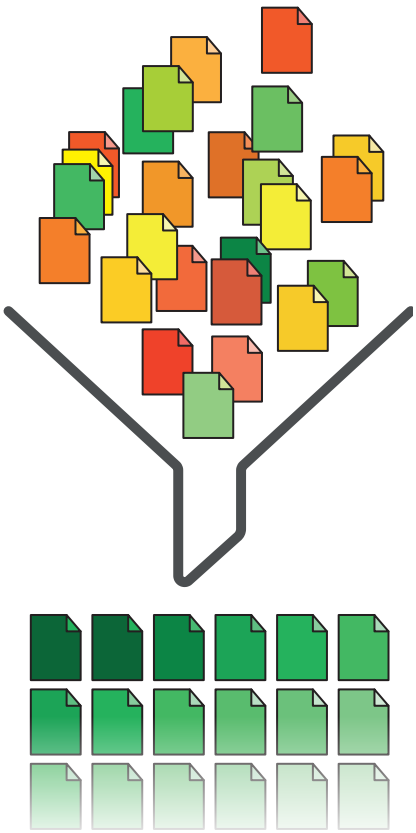


Figure 4.2: An illustration for search space reduction, retrieving relevant documents from large corpora with high recall.

To this end, we take a human-in-the-loop approach and advocate interactive and exploratory retrieval. In our framework, users explore retrieved documents, learn them, and interactively build targets, which will be used to sift through documents. Instead of users rating a number of retrieved documents generated by systems, our method allows the users to proactively modify target keywords and give relevance feedback. In addition, we adopt *targeted topic modeling* to support this process. Targeted topic modeling techniques find relevant topics and disregard irrelevant aspects from document collections. Utilizing results from targeted topic modeling, our approach allows users to discover relevant subtopics and refine the targets using topic-level relevance feedback.

In this chapter, we propose a novel framework for interactive search space reduction

along with an effective visual analytics system called TopicSifter. TopicSifter tightly integrates the underlying computational methods and interactive visualization to support topic model exploration and targeted topic modeling.

The primary contributions of this work include:

- A novel iterative and interactive technique for search space reduction through interactive target building, sifting, and targeted topic modeling.
- A visual analytics prototype, TopicSifter, that supports tight integration between the interactive visualization and the underlying algorithms.
- Experiments and use cases that illustrate the effectiveness of TopicSifter.

4.2 Related Work

In this section, we discuss prior works on information retrieval and topic modeling in the context of search space reduction.

4.2.1 Visualizing Search Results/Space

Various information visualization techniques have been applied to improve user interfaces for search. Some systems augment search result lists with additional small visualizations. For example, TileBars [59], INSYDER [60], and HotMap [61] visualize query-document relationships as icons or glyphs alongside search results. Another approach is to visualize search results in a spatial layout where proximity represents similarity. Systems such as InfoSky [62] and IN-SPIRE [10] are examples. FacetAtlas [11] overlays additional heatmaps to visualize density. ProjSnippet [63] visualizes text snippets in a 2-D layout. Many others cluster the search results and offer faceted navigation. FacetMap [64] and ResultMap [65] utilizes treemap-style visualizations to represent facets. These systems may guide users well in exploring search results, but they are mostly based on static search queries. Our system goes beyond search results exploration and offers interactive target (query) building.

4.2.2 Query Expansion and Relevance Feedback

Information retrieval is finding (unstructured) documents that satisfies an information need from large collections [66]. However, users of information retrieval systems may not have a clear idea of what to search for, may not know how to construct an optimal query, or may not understand what kind of information is available [67]. To this end, various interactive methods to assist the retrieval process have been proposed. Interactive query expansion [68] allows the users to choose additional query terms from the suggested list of keywords. Instead of lists, Fowler et al. [69] and Hoeber et al. [70] display keyword suggestions as graphs. Sparkler [71] visualize multiple query results so that users can compare and identify the best query from the expanded queries. In our system, we suggest additional keywords for queries in terms of two categories of good-to-have keywords *and bad-to-have* keywords. Another interactive approach is relevance feedback, meaning users are asked to mark documents as relevant to steer the system to modify the original query [67]. For instance, VisIRR [72] allows users to rate retrieved documents on a 5-star scale. IntentRadar [73, 74] models intents behind search queries and lets users give relevance feedback on the intents to interactively update them. We adopt a similar approach to give relevance feedback to documents *as well as groups of documents (topics)*. These existing systems are designed for the traditional information retrieval setting of obtaining the most relevant data items with high precision, and thus are not well-suited for our search space reduction setting which desires high recall. Closer to our work is ReQ-ReC [58] which combines iterative query expansion and iterative classifier refinements to solve high recall retrieval problem. A major difference is that ReQ-ReC system requires users to label given documents while our system allows the users to explore the documents and their topics and give relevance feedback if needed.

4.2.3 Aspect-Specific Topic Summarization

Although topic summarization has been studied for a long time, discovering topic summary of a specific aspect (or targets) is a relatively new research problem. TTM [57] is the first work to propose the term 'targeted topic modeling'. This work proposes a probabilistic model that is a variation of latent Dirichlet allocation (LDA) [4]. Given a static keyword list defining a particular aspect, the model identifies topic keywords related to this aspect. Wang et al. [75] identifies a list of target words from review data and disentangles aspect words and opinion words from the list. APSUM [76] assigns aspects to each word in a generative process. Since the aforementioned model generates topic keywords based on a static keyword list, a dynamic model is desired. An automatic method to generate keyword dynamically has been proposed [77]. This method focuses on the on-line environment of Twitter and automatically generates keywords based on the time-evolving word graph.

4.2.4 Interactive Topic Modeling

Interactive topic models allow users to steer the topics to improve the topic modeling results. Various topic steering interactions such as adding, editing, deleting, splitting, and merging topics have been introduced [19, 13, 12, 20, 15, 18, 33, 37, 24]. These interactions can be applied to refine relevant topics and remove irrelevant topics to identify targeted topics when most of the data items are relevant and only a small portion is irrelevant. However, in our large-scale search space reduction setting, a more tailored approach is needed. In this chapter, we propose interactive targeted topic modeling to steer the topics to discover the target-relevant topics and documents.

4.3 Interactive Search Space Reduction

In many practical cases in large-scale text analyses, users have specific aspects they want to focus on, which we will refer to as targets. Although there are many tools available

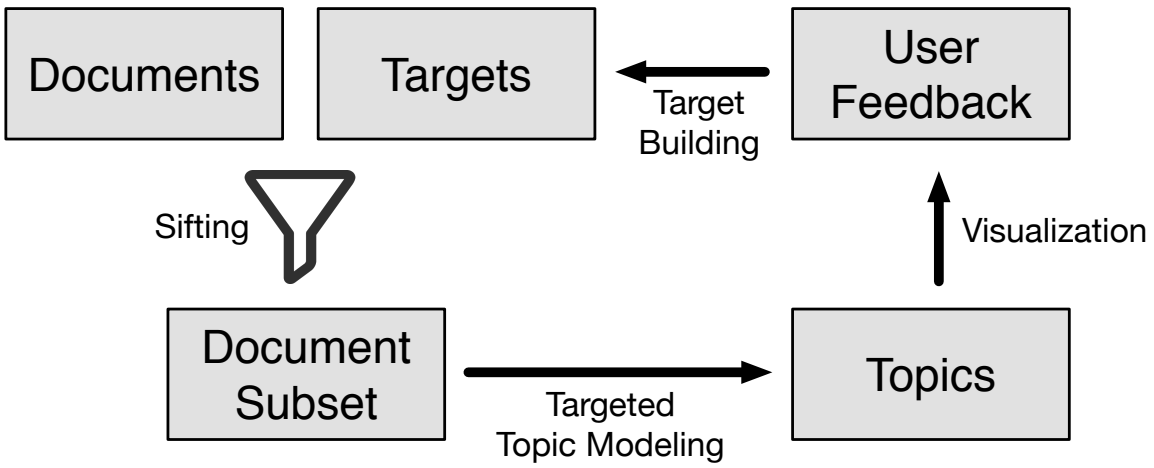


Figure 4.3: Our human-in-the-loop algorithm workflow for interactive search space reduction. A document subset is updated based on user feedback each iteration.

with powerful natural language processing and text mining features, they tend to lack the ability to concentrate on the targets. We define this problem of retrieving a subset of documents that are relevant to given targets from large-scale datasets with high recall as search space reduction. Our solution is to iteratively retrieve the relevant documents utilizing user feedback. Over multiple iterations, users inspect a topical summary of previously retrieved documents and give feedback, and our system updates targets to better reflect their mental model and retrieve relevant documents through sifting.

In this section, we first formulate the problem of interactive search space reduction, then describe our iterative workflow and algorithm.

4.3.1 Problem Formulation and Algorithm Workflow

Given a document collection $D = \{d_1, \cdots, d_n\}$ with n documents, our goal is to retrieve a subset $D^* \subseteq D$ of documents that are relevant with high recall. Note that we do not limit the number of retrieved documents $|D^*|$, as opposed to traditional information retrieval. Our iterative approach updates targets $G^{(t)}$ based on user feedbacks and retrieves documents $D^{(t)}$ over iterations $t = 1, \cdots, T$.

Our algorithm workflow is outlined in Fig. 4.3, with notation listed in Table 4.1. Each

Table 4.1: Key notations used in Chapter 4.

Notation	Description
D	Given document collection of n documents $\{d_1, \cdots, d_n\}$
W	Given keyword dictionary of m keywords $\{w_1, \cdots, w_m\}$
X	The $m \times n$ word-document matrix of D
t	Current iteration number
$D^{(t)}$	Set of n_t retrieved documents at the t-th iteration, $\{d_1^{(t)}, \cdots, d_{n_t}^{(t)}\}$
$s_r^{(t)}(\cdot)$	Relevance score of document/topic at the t-th iteration
$s_c^{(t)}(d)$	Topic score of document d at the t-th iteration
$G^{(t)}$	Set of r_t targets at the t-th iteration, $\{g_1^{(t)}, \cdots, g_{r_t}^{(t)}\}$
$\boldsymbol{G}^{(t)}$	Set of r_t target vectors for $G^{(t)}$, $\{\boldsymbol{g}_1^{(t)}, \cdots, \boldsymbol{g}_{r_t}^{(t)}\}$
$T^{(t)}$	Set of k_t topics at t-th iteration, $\{T_1^{(t)}, \cdots, T_{k_t}^{(t)}\}$
$w(T)(w(d))$	Set of top ten keywords of a topic T (or a document d)
$d(T)$	Set of documents that belong to a topic T
$W_+^{(t)}(W_-^{(t)})$	Set of good-to-have (bad-to-have) keywords by users at the t-th iteration
$T_+^{(t)}(T_-^{(t)})$	Set of upvoted (downvoted) topics by users at the t-th iteration
$D_+^{(t)}(D_-^{(t)})$	Set of upvoted (downvoted) documents by users at the t-th iteration
$X^{(t)}$	The word-document matrix of $D^{(t)}$, $X^{(t)} = X(:, D^{(t)})$
$V^{(t)}$	The word-topic matrix of $T^{(t)}$
$H^{(t)}$	The topic-document matrix of $T^{(t)}$
$\boldsymbol{x}_j^{(t)}$, $\boldsymbol{v}_j^{(t)}$, $\boldsymbol{h}_j^{(t)}$	j-th column of $X^{(t)}$, $V^{(t)}$, $H^{(t)}$, respectively
$\mathbb{R}_+$	The set of nonnegative real numbers
$\|\| \cdot \|\|_F$	The Frobenius norm
$\boldsymbol{e}_i$	The standard basis vector where $\boldsymbol{e}_i(j) = 1$ for $j = i$.
$A_{i\cdot}$	The i-th row of matrix A
$A_{\cdot j}$	The j-th column of matrix A
$argmax(\boldsymbol{a})$	The index of the largest element in vector $\boldsymbol{a}$

iteration consists of three computational steps: target building, sifting, and targeted topic modeling. An iteration starts with user feedback from its previous iteration. After exploration of previously retrieved documents and their topics, users can modify keyword queries and/or give positive or negative feedback on topics or documents. Based on the user input, the *interactive target building* step (Section 4.3.2) updates the targets $G^{(t-1)} \rightarrow G^{(t)}$. Next, the *sifting* step (Section 4.3.3) selects a new set of documents $D^{(t)}$ using the updated targets $G^{(t)}$. Finally, the *targeted topic modeling* step (Section 4.3.4) generates topics $T^{(t)}$ and the

system visualizes them. The users can repeat the iterative process until satisfied.

4.3.2 Interactive Target Building Based on User Feedback

We represent targets as a set of single keywords (e.g., "apple") or keyword compounds (e.g., "apple, orange"). The former looks for documents containing the single keyword and the latter looks for those containing all of the keywords in the keyword compound. In the search space reduction problem, users may not be familiar with their target domains [78]. Even for domain expert users, constructing a good static query is a challenging task without exploring and understanding given datasets in advance. Both cases can be solved with interactive target building. At each iteration, our interactive target building step updates targets based on user feedback.

Different from existing information retrieval approaches that use positive queries, we use negative as well as positive targets. This allows users to express their complicated mental target model. For example, the users may be interested in a target, but not interested in a similar concept (e.g., retrieve "apple, fruit" and ignore "orange, fruit"). Negative targets can also deal with multi-meaning words (e.g., retrieve "apple, iphone" and ignore "apple, fruit"). In detail, we allow the users to directly update the keyword sets including good-to-have keywords, bad-to-have keywords, stopwords to be ignored. Stopwords are the words that are not useful in text analysis including too frequent words such as articles, prepositions, and pronouns. In addition to the commonly used English stopwords, we allow the users to add custom stopwords that are data-specific or domain-specific. For example, when exploring medical records, ignoring common medical terms may increase the quality of topic modeling and sifting. Also, the users can indirectly update the target by giving item-level (documents) or group-level (topics) relevance feedback.

Our approach incorporates seven kinds of user relevance feedback into target building:

RF 1 *Edit good-to-have keywords*

RF 2 *Edit bad-to-have keywords*

RF 3 *Edit stopwords (words to be ignored)*

RF 4 *Confirm/upvote topics*

RF 5 *Reject/downvote topics*

RF 6 *Confirm/upvote documents*

RF 7 *Reject/downvote documents*

RF 1, **RF 4**, **RF 6** are positive relevance feedback indicating that the corresponding words, topics, or documents are relevant to the user's mental target G, respectively. On the contrary, **RF 2**, **RF 5**, **RF 7** are negative relevance feedback indicating that the corresponding words, topics, or documents are irrelevant to the user's mental target G, respectively. Lastly, **RF 3** modifies the set of stopwords, which affects the follow-up topic modeling process described in Section 4.3.4.

Given user relevance feedback, we model the targets and their representative vectors as follows:

TargetModel computes the targets $G^{(t)}$ and their vectors $\boldsymbol{G}^{(t)}$ at the t-th iteration using the user supplied input $(W_+^{(t)},\ W_-^{(t)},\ T_+^{(t)},\ T_-^{(t)},\ D_+^{(t)},\ D_-^{(t)})$. The target $G^{(t)}$ consists of positive/negative explicit/implicit parts. Users can change the explicit part $G_+^{(t)},\ G_-^{(t)}$ directly through keyword modification. For implicit part $\bar{G}_+^{(t)},\ \bar{G}_-^{(t)}$, using relevance feedback on a topic or a document, we extract its top keywords and add the keyword compound as an implicit target.

$[G^{(t)},\ \boldsymbol{G}^{(t)}] = \textbf{TargetModel}(W_+^{(t)},\ W_-^{(t)},\ T_+^{(t)},\ T_-^{(t)},\ D_+^{(t)},\ D_-^{(t)})$

$G_+^{(t)} = W_+^{(t)}; \boldsymbol{G}_+^{(t)} = \{\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2} : \boldsymbol{a} = \sum_{w \in g} \boldsymbol{e}_w,\ g \in G_+^{(t)}\}$

$G_-^{(t)} = W_-^{(t)}; \boldsymbol{G}_-^{(t)} = \{\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2} : \boldsymbol{a} = \sum_{w \in g} \boldsymbol{e}_w,\ g \in G_-^{(t)}\}$

$\bar{G}_+^{(t)} = \{w(T_j^{(t-1)}) : T_j^{(t-1)} \in T_+^{(t)}\} \cup \{w(d_j^{(t-1)}) : d_j^{(t-1)} \in D_+^{(t)}\}$

$\bar{\boldsymbol{G}}_+^{(t)} = \{\boldsymbol{v}_j^{(t-1)} : T_j^{(t-1)} \in T_+^{(t)}\} \cup \{\boldsymbol{x}_j^{(t-1)} : d_j^{(t-1)} \in D_+^{(t)}\}$

$\bar{G}_-^{(t)} = \{w(T_j^{(t-1)}) : T_j^{(t-1)} \in T_-^{(t)}\} \cup \{w(d_j^{(t-1)}) : d_j^{(t-1)} \in D_-^{(t)}\}$

$\bar{\boldsymbol{G}}_-^{(t)} = \{\boldsymbol{v}_j^{(t-1)} : T_j^{(t-1)} \in T_-^{(t)}\} \cup \{\boldsymbol{x}_j^{(t-1)} : d_j^{(t-1)} \in D_-^{(t)}\}$

$G^{(t)} = (G_+^{(t)},\ G_-^{(t)},\ \bar{G}_+^{(t)},\ \bar{G}_-^{(t)}); \boldsymbol{G}^{(t)} = (\boldsymbol{G}_+^{(t)},\ \boldsymbol{G}_-^{(t)},\ \bar{\boldsymbol{G}}_+^{(t)},\ \bar{\boldsymbol{G}}_-^{(t)})$

Keyword Suggestion

Manually entering keywords can be burdensome. To this end, we recommend candidates for the good-to-have and bad-to-have keyword sets in real time. Candidate recommendation is based on similarities with the current good-to-have and bad-to-have keyword sets. Similarities between words can be calculated by several distance measures. Among them, we adopt the vector-space model of word representation [79]. To learn word vectors, we use empirical pointwise mutual information (ePMI), which measures co-occurrence between word pairs. The ePMI score between the word pair (w_i, w_j) is defined as:

$$ePMI(w_i, w_j) = \log\Big(\frac{\#(w_i, w_j) \cdot N}{\#(w_i) \cdot \#(w_j)}\Big), \tag{4.1}$$

where N denotes the total number of the word co-occurring word pairs; and $\#(w_i, w_j)$ and $\#(w_i)$ denote the number of occurrences of the word pair (w_i, w_j) and the single word w_i, respectively. As suggested by [80], we first construct a matrix $P \in \mathbb{R}^{m \times m}$ where $P_{i,j} = ePMI(w_i, w_j)$, perform low-rank matrix factorization on P, and use the left factor as the vector representations of words after $l2$-normalization. We computed word vectors for each dataset to obtain dataset-specific word similarities, but pre-trained word vectors using word2vec [79] or Glove [81] can be used in our algorithm.

4.3.3 Sifting Documents and Words

After the target modeling step, we retrieve a new set of documents using the updated targets. We provide two retrieval options: hard filtering by target keywords and soft sifting.

HardSift throws out documents that contain one of the negative target elements or their nearest words and retrieves documents that contain one of the target elements or their nearest words. One of nearest words of a word w is denoted by $sim(w)$. Note that we apply negative feedback first and positive feedback later to take a conservative approach in filtering out documents.

$D^{(t)} = \textbf{HardSift}(G^{(t)},\ T_+^{(t)},\ T_-^{(t)},\ D_+^{(t)},\ D_-^{(t)},\ D^{(t-1)})$

$D^{(t)} = D^{(t-1)} \bigcup_{j=1}^{|G_+^{(t)}|} \{d_i \in D : \forall w \in g_j (\in G_+^{(t)}),\ d_i \text{ has } sim(w)\}$

$D^{(t)} = D^{(t)} \Big\backslash_{j=1}^{|G_-^{(t)}|} \{d_i \in D^{(t)} : \forall w \in g_j (\in G_-^{(t)}),\ d_i \text{ has } sim(w)\}$

$D^{(t)} = D^{(t)} \Big\backslash_{T_j^{(t-1)} \in T_-^{(t)}} d(T_j^{(t-1)})$

$D^{(t)} = D^{(t)} \Big\backslash_{d_j^{(t-1)} \in D_-^{(t)}} \{d_i \in D^{(t-1)} : (\boldsymbol{x}_i \cdot \boldsymbol{x}_j^{(t-1)}) > \delta\}$

$D^{(t)} = D^{(t)} \bigcup_{T_j^{(t-1)} \in T_+^{(t)}} \{d_i \in D : (\boldsymbol{x}_i \cdot \boldsymbol{v}_j^{(t-1)}) > \delta\}$

$D^{(t)} = D^{(t)} \bigcup_{d_j^{(t-1)} \in D_+^{(t)}} \{d_i \in D : (\boldsymbol{x}_i \cdot \boldsymbol{x}_j^{(t-1)}) > \delta\}$

SoftSift incorporates a relevance score model to rank documents by how similar they are to the explicit and implicit targets. The relevance score of a document with respect to a target g is calculated as cosine similarity between its target vector $\boldsymbol{g}$ and the document vector $\boldsymbol{x}$, $(\boldsymbol{x} \cdot \boldsymbol{g})$. All target vectors and document vectors are l2-normalized. To calculate the final relevance score of a document, we take a weighted average of its previous relevance score and its relevance scores with respect to positive and negative feedbacks at the current iteration. To put more emphasis on recall than precision, we use smaller weight for negative feedback score than positive feedback score, i.e. $\beta > \gamma$.

$D^{(t)} = \textbf{SoftSift}(\boldsymbol{G}^{(t)},\ D)$

$\alpha,\ \beta,\ \gamma$ is parameters for balancing previous scores, positive feedback, and negative feedback, respectively.

δ is the threshold for the soft mode.

for $d_i \in D$ **do**

$s_{r+}^{(t)}(d_i) = mean_{\boldsymbol{g}_+^{(t)} \in \boldsymbol{G}_+^{(t)} \cup \bar{\boldsymbol{G}}_+^{(t)}} (\boldsymbol{x}_i \cdot \boldsymbol{g}_+^{(t)})$

$s_{r-}^{(t)}(d_i) = mean_{\boldsymbol{g}_-^{(t)} \in \boldsymbol{G}_-^{(t)} \cup \bar{\boldsymbol{G}}_-^{(t)}} (\boldsymbol{x}_i \cdot \boldsymbol{g}_-^{(t)})$

$s_r^{(t)}(d_i) = \alpha s_r^{(t-1)}(d_i) + \beta s_{r+}^{(t)}(d_i) - \gamma s_{r-}^{(t)}(d_i)$

end for

$D^{(t)} = \{d_i \in D : s_r^{(t)}(d_i) > \delta\}$

4.3.4 Targeted Topic Modeling

The last step of an iteration is targeted topic modeling. Targeted topic modeling finds a target-specific topical summary of documents that are retrieved from the previous sifting step. The calculated topics and their representative documents are visualized to the users so that they can easily understand what kind of documents are retrieved at the current iteration and perform relevance feedback for the next iteration.

In this section, we explain nonnegative matrix factorization (NMF) [82] in the topic modeling context [7, 56] and our targeted topic modeling algorithm based on NMF with additional constraints.

Background: NMF for Topic Modeling

Given a nonnegative matrix $X \in \mathbb{R}_+^{m \times n}$, NMF approximates X as a product of nonnegative factor matrices $V \in \mathbb{R}_+^{m \times k}$ and $H \in \mathbb{R}_+^{k \times n}$, i.e., $X \approx VH$, with $k \ll \min(m, n)$. This can be solved by optimizing the following formula:

$$\min_{\{V,H\} \geq 0} ||X - VH||_F^2. \tag{4.2}$$

In the topic modeling context, X is a word-document matrix where $X_{\cdot j}$ (the j-th column vector of X) is a bag-of-words representation of j-th document over m keywords. X is based on TF-IDF representation of the document set and usually normalized with $l2$-norm. k is set to be the number of topics. Factor matrices V and H represent word-topic and topic-document relationships, respectively. $V_{\cdot i}$ represents the i-th topic as a distribution over words. Large values in $V_{\cdot i}$ indicate that the corresponding keywords are strongly associated with the i-th topic. $H_{\cdot j}$ represents the j-th document d_j as a weighted combination of topics. The j-th document d_j belongs to the i-th topic if the i-th element of $H_{\cdot j}$ is its maximum, i.e., $argmax(H_{\cdot j} = i)$. We denote the i-th topic as T_i and define it by its word distribution vector ($w(T_i) = V_{\cdot i}$) and the documents that belong to it ($d(T_i) = \{d_j | argmax(H_{\cdot j}\})$.

Targeted Topic Modeling using NMF

To reflect a target built by users into the topic modeling process, we introduce an additional constraint term to the standard NMF formula, Eqn. 4.2, as follows:

$$\min_{\{V,H\}\geq 0} ||X - VH||_F^2 + \rho||M \circ V - V_G||_F^2, \tag{4.3}$$

where $\circ$ is an elementwise multiplication. The additional term forces certain topics' word representation V to be similar to the corresponding target elements V_G with the help of masking coefficient matrix M. The parameter ρ controls the balance between the original term and the additional term. Bigger ρ results in stronger incorporation of the target in topic modeling. That is, the bigger the rho is, the closer the topics become to the targets at the expense of becoming less truthful representation of data. When $\rho = 0$, it is equivalent to the standard topic modeling. Also, ρ is inversely proportional to the number of positive targets. To compute M and V_G, for each positive target vector $\boldsymbol{g}_i \in \boldsymbol{G}_+ \cup \bar{\boldsymbol{G}}_+$, find its closest topic vector, which we define as $\boldsymbol{v}_i^* = argmax_{\boldsymbol{v}_j}(\boldsymbol{v}_j \cdot \boldsymbol{g}_i)$. We set $(V_G)_{j\cdot} = mean_{\boldsymbol{v}_i^* = \boldsymbol{v}_j}(\boldsymbol{g}_i)$ and $M_{j\cdot} = 1$ if $|\{\boldsymbol{g}_i : \boldsymbol{v}_i^* = \boldsymbol{v}_j\}| > 0$.

The detailed algorithm at the t-th iteration is as follows:

TargetedTopicModel applies a constrained NMF algorithm on the current document set $D^{(t)}$ and the current targets $G^{(t)}$ to compute k_t number of topics $T^{(t)}$. Additionally, we calculate each topic's relevance score with respect to the targets. Note that $rank(w, T_i)$ calculates the rank of a word w within the topic T_i's topic vector $\boldsymbol{v}_i$. For speedup, we use a fast rank-2 NMF [52] algorithm to initialize V and H in Eqn. 4.2.

$[T^{(t)},\ s_r^{(t)},\ s_c^{(t)}] =$ **TargetedTopicModel**$(X^{(t)},\ k_t,\ G^{(t)})$

Generate $V_G^{(t)}$, $M^{(t)}$ and solve

$\min_{\{V^{(t)},H^{(t)}\}\geq 0} ||X^{(t)} - V^{(t)}H^{(t)}||_F^2 + \rho||M^{(t)} \circ V^{(t)} - V_G^{(t)}||_F^2$

$s_r^{(t)}(T^{(t)}) = 1 - min_{g \in G_+^{(t)}}\big(mean_{w \in g}\dfrac{rank(w, T^{(t)})}{|W|}\big)$

$s_c^{(t)}(d_j^{(t)}) = max(\boldsymbol{h}_j)/sum(\boldsymbol{h}_j)$

4.4 System

In this section, we present TopicSifter, our interactive document search space reduction system. Our visualization system is tightly integrated with the underling algorithms described in Section 4.3 to support various user feedback interactions listed in Section 4.3.2.

TopicSifter is designed to meet these design goals:

1. **Given targets, retrieve relevant documents with high recall:** TopicSifter should retrieve documents that are relevant to targets.
2. **Show summary and details of sifted documents:** TopicSifter should provide a topical summary and details of retrieved documents to help users understand them.
3. **Support positive and negative feedback:** Users should be able to positive and negative relevance of both keywords, documents, and topics (Supporting **RF 1**-**RF 7**).
4. **Modify targets over iterations:** TopicSifter should allow users to update targets easily and iteratively.
5. **Observe changes between iterations:** TopicSifter should show differences in retrieved documents between iterations.
6. **Export results for further analysis:** TopicSifter is designed for one step of a complex text analysis workflow. Users should be able to export the retrieved documents for in-depth analyses.

TopicSifter consists of a web-based visualization interface using D3.js and a backend system in Python and MATLAB using the Django framework.

4.4.1 System Overview

TopicSifter consists of three panels: (1) the control panel, (2) the main view, and (3) the detail panel (Fig. 4.1). The control panel contains the keyword module to modify good-to-have words, bad-to-have words, and stopwords (supporting **RF 1**, **RF 2**, **RF 3**) and control

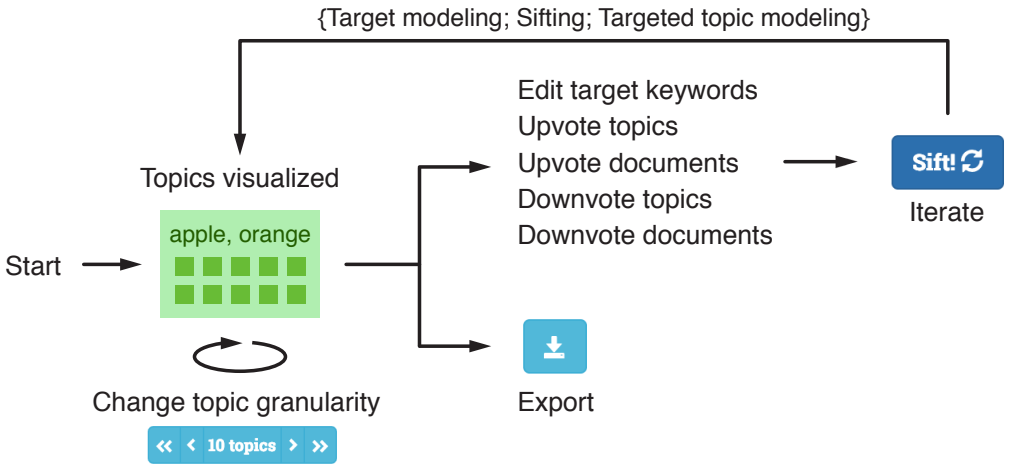


Figure 4.4: TopicSifter workflow. Users can provide feedback before clicking the blue button to move to the next iteration. Results can be saved by clicking the export button.

buttons to update the main view. The main view shows the sifting status and the topical overview of retrieved documents at the current iteration and allows the users to upvote or downvote topics and documents (supporting **RF 4**, **RF 5**, **RF 6**, **RF 7**). The relevance feedback on words, topics, and documents will be reflected on the next iteration (Fig. 4.4). Lastly, the detail panel has the document table to show additional detail of all documents and the history view to show historical trends over iterations. The width of each panel is adjustable by dragging the divider in order to allocate more or less space to the panel. The system design is shown in Fig. 4.1.

The users follow the workflow in Fig. 4.4. Each iteration starts with the users exploring the retrieved documents and their topics in the main view. To give relevance feedback, the users can modify keyword sets in the control panel or upvote/downvote topics and documents in the main view. They can export the results or move on to the next iteration using buttons in the control panel.

4.4.2 Control Panel

The users can utilize the control panel to update the main view. The control panel contains the keyword input module and the control buttons. The keyword input module shows cur-

rent set of good-to-have keywords $W_+^{(t)}$, bad-to-have keywords $W_-^{(t)}$, and stopwords and allows users to modify them (**RF 1**, **RF 2**, **RF 3**).

Keyword Input Module

In the keyword input module (Fig. 4.1(d)), the users can add new keywords using an input text box or see current keyword lists for good-to-have keywords, bad-to-have keywords, and stopwords. To add a keyword, users can enter the keyword in the input text box. While typing, possible matching keywords in the dictionary W is listed in the pop-up as shown in Fig. 4.5(a). The list is sorted by word frequency and updated as the user types more letters. After selecting one of the keywords in the pop-up list, the users can either enter the keyword as a single keyword (e.g., "visual") or build a keyword compound (e.g., "visual" AND "analyt(ic)" in Fig. 4.5(b)). By clicking one of the green, red, or gray buttons in Fig. 4.5(c), the entered keyword or keyword compound is added in the good-to-have keyword list $W_+^{(t)}$, the bad-to-have keyword list $W_-^{(t)}$, or the stopword list, respectively. Keywords or keyword compounds in the keyword list is visualized as word buttons inside the colored areas (good-to-have: green, bad-to-have: red, stopword: gray) as in Fig. 4.5(c). In order to remove a keyword or a keyword compound, the users can click the $\times$ icon on the keyword button.

As discussed in Section 4.3.2, our technique suggests additional keywords based on the current set of keywords. The keywords recommended for good-to-have or bad-to-have lists are visualized under the corresponding keyword list as keyword buttons with dashed borders with a + icon. The users can add one of suggested keywords by clicking the + icon. The recommended keywords are updated in real time as the users add or remove keywords to the keyword lists.

Changing the number of topics

Users can change the topic granularity by increasing or decreasing the number of topics using the button group in the control panel. When the generated topics are too fine-grained or too coarse-grained, giving relevance feedback can be problematic. For example, the

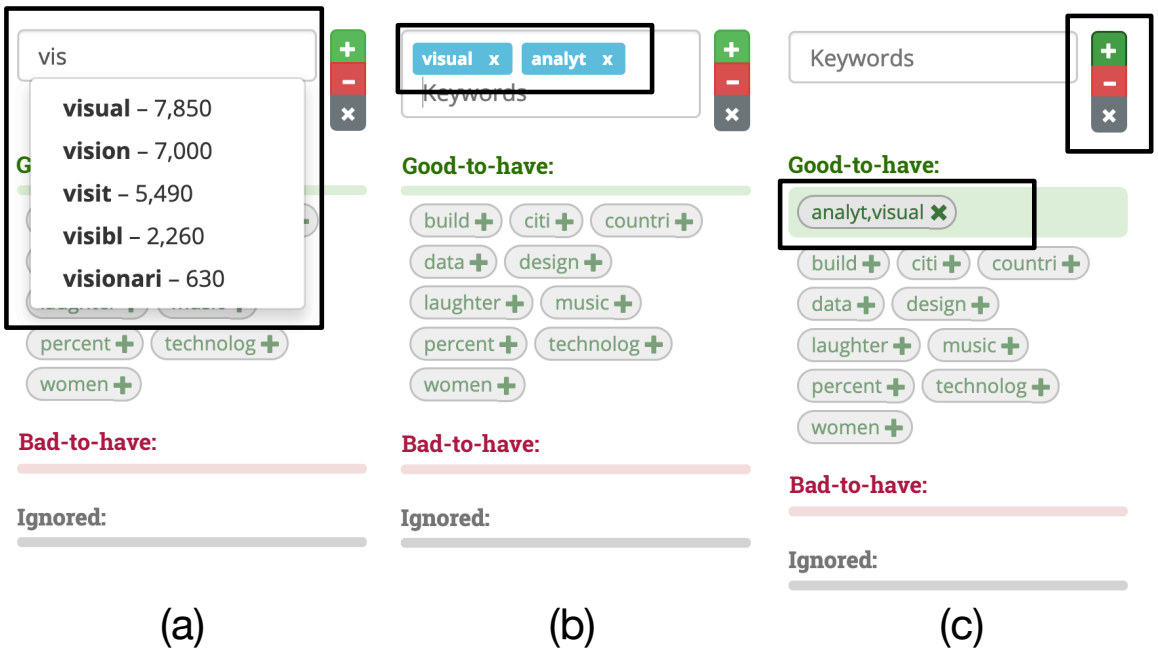


Figure 4.5: Users can add good-to-have words, bad-to-have words, and stopwords in the control panel. (a) While typing, partially matched keywords are ranked by frequency and shown in a pop-up list. (b) Multi-word compound is supported. (c) Clicking the green button adds the entered keyword compound into the good-to-have set.

user wants to give feedback on all "fruit" related topics, but there are too many fine-grained "fruit" related topics to interact with. On the other hand, the user may be interested in part of a topic (e.g., like "apple, mac" part from "apple, mac, fruit", but not "apple, fruit" part). The number of current topics is shown in the middle part of the button group. The users can click the buttons to decrease the number of topics by -5($\ll$), -1($<$), or increase it by +1($>$), +5($\gg$). Note that a new set of topics is generated using the same retrieved document subset. The visual update after changing the number of topics is fast since this happens within an iteration without triggering the target building, sifting, and targeted topic modeling processes (Fig. 4.4).

Sift Button Sift!

The users can run our backend algorithms by clicking the sift button. After modifying good-to-have keywords, bad-to-have keywords, and stopwords (**RF 1**, **RF 2**, **RF 3**) and upvoting or downvoting topics and documents (**RF 4**, **RF 5**, **RF 6**, **RF 7**), the users move on

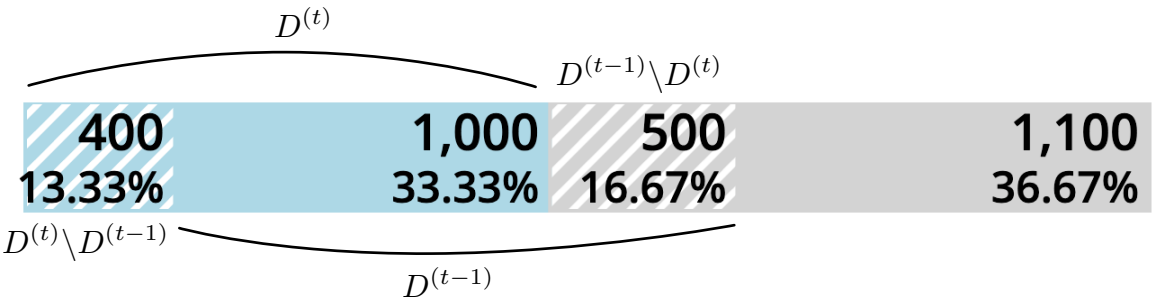


Figure 4.6: The status bar chart that shows sifting status at the current t-th iteration. Blue bars represent retrieved documents $D^{(t)}$ at the t-th iteration. Gray bars represent the rest (sifted out) documents in the corpus, i.e., $D \setminus D^{(t)}$. Patterned bars indicate changes from the $(t-1)$-th iteration.

to the next iteration. The sift button triggers the target building, sifting, and targeted topic modeling processes to retrieve a new set of documents and visualize their topic summary. This process is shown in Fig. 4.4.

Export Button

The users can export the results using the export button. When the users are satisfied with the retrieved documents after multiple iterations, our system provides an option to save the results. The results are saved as a JSON file including targets, topics, and IDs, topic membership, and relevance scores of retrieved documents.

4.4.3 Main View

The main view will visualize topic summary of retrieved documents at the current iteration along with the sifting status bar to show the difference between the current iteration and the previous iteration. In the topic visualization, the users can upvote or downvote topics and documents to indicate that they are relevant to targets or not (**RF 4**, **RF 5**, **RF 6**, **RF 7**).

Status Bar

Fig. 4.6 shows the status bar chart. The total length of all bars represents the total number of documents in the dataset. The total length of blue bars represents the number of retrieved

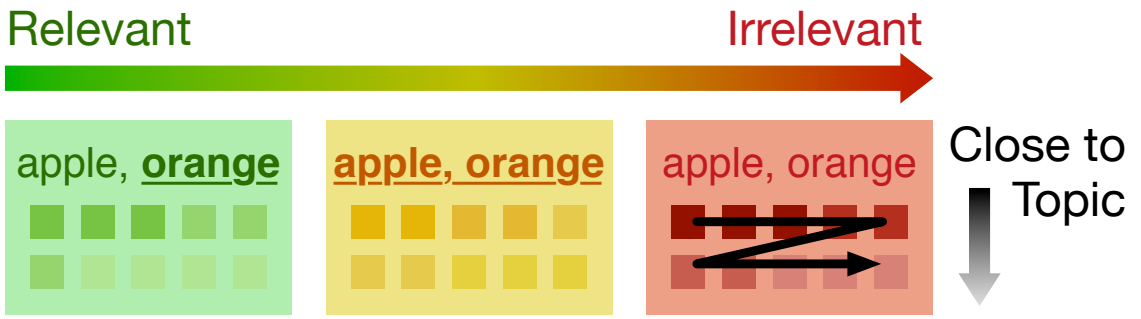


Figure 4.7: Visual encoding of topic cells and their representative documents. Target-relevancy of topics are encoded by their color hue (green to red). Topic-closeness of documents are encoded by their color lightness (dark to light) and positions (top-left to bottom-right). Topic change from the previous iteration is indicated by new keywords highlighted as bold and underlined.

documents at the current iteration, t, while the total length of gray bars represents the number of sifted out documents. Solid-colored bars represent documents that stay retrieved (solid blue) or stay sifted out (solid gray) between the previous $(t-1)$-th iteration and the current t-th iteration. Patterned bars represent document status changes from the previous iteration, $t-1$. In detail, the blue patterned bar represents incoming documents that were not retrieved at the $(t-1)$-th iteration but retrieved at the t-th iteration. The gray patterned bar represents outgoing documents that were retrieved at the $(t-1)$-th iteration but sifted out at the t-th iteration. Longer patterned bars indicate interactions at the t-th iteration have resulted in a larger change in retrieved documents.

Topic Visualization

Topics computed from the retrieved documents are visualized as rectangular cells (Fig. 4.1). On top of each topic cell, its top ten keywords are shown, and its representative documents are visualized as small squares. The sizes of cells are proportional to the number of retrieved documents that belong to each topic. The layout of cells is calculated by D3's built-in treemap algorithm. The color hues of topic cells represent how relevant each topic is to the target ($s_r^{(t)}(T)$) from green (relevant) to red (irrelevant) as in Fig. 4.7. The color hue of each topic is shared by its keywords and its documents. If a topic has changed from

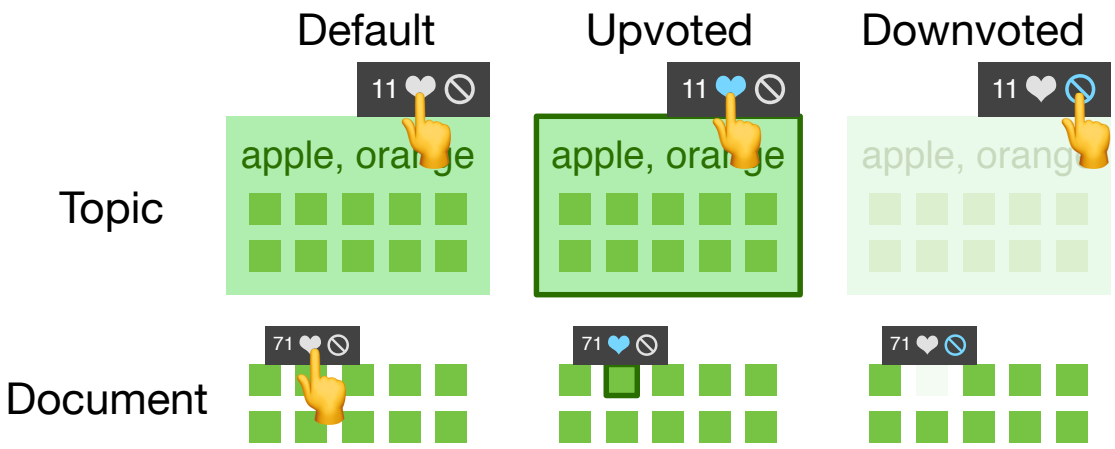


Figure 4.8: User interaction for positive and negative feedback. Users can click the upvote (or downvote) button in the pop-up menu of a topic or a document to indicate (ir-)relevancy. Highlight with border means upvoted and white out means downvoted.

the previous iteration, new representative keywords are highlighted as bold and underlined (The yellow topic in Fig. 4.7). For topic cells with narrow width, the users can hover over top keywords to see the full list of keywords. To give positive (or negative) feedback to topics, the users can click the menu button on the top right corner of each topic cells to open a pop-up menu with upvote and downvote button (Fig. 4.8).

The number of representative documents that are visualized as squares in a topic cell are determined by the size of the cell. Our system picks documents to be visualized by how close the documents are to its topic ($s_c^{(t)}(d)$) since they are more representative of the topic. The color lightness of document squares represents how close each document is to its topic from dark (close) to light (less close). The positions of document squares are also sorted by closeness to their topics from top-left to bottom-right (Fig. 4.7). To see the detail of a document, the users can hover over the square to see its document ID in a pop-up menu or click the square to see its detail in the document table in the detail panel. Users are able to give positive (or negative) feedback to documents to indicate that they are relevant (or irrelevant) to their mental targets by toggling the upvote (or downvote) button in the pop-up menu of each document square as in Fig. 4.8. Upvoted topics and documents are highlighted with border and downvoted topics and documents are whited out.

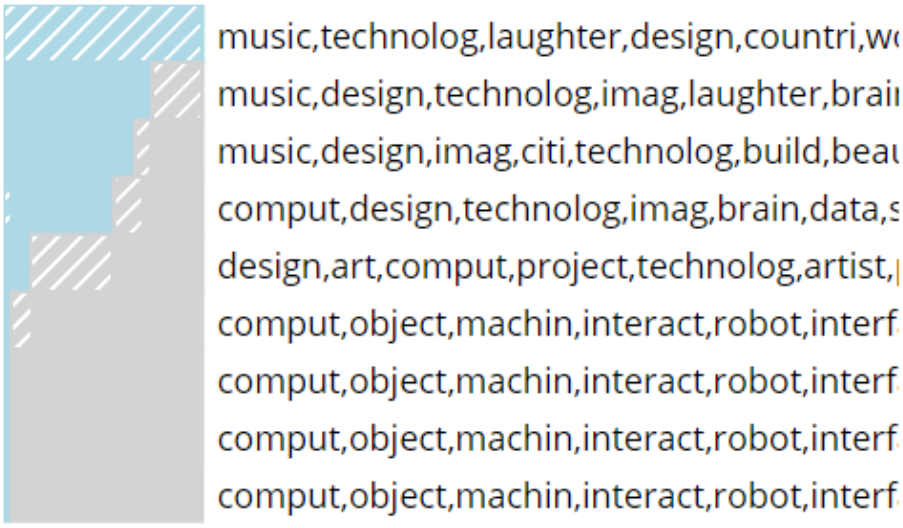


Figure 4.9: The history view. Stacked bars on the left show sifting status from top (old) to bottom (new). In each row, blue bars represent retrieved documents at an iteration and gray bars represent the rest (sifted out) documents in the corpus at the same iteration. Patterned bars indicate changes from the previous iteration. Keyword summary on the right shows the topical progression of retrieved documents over iterations.

4.4.4 Detail Panel

The detail panel has two tabs to toggle between the document table view and the history view. The document table view shows the list of all documents D and their raw text details. The history view shows the history of previous iterations to keep track of the iterative sifting process.

Document Table

The document table shows additional information of all documents in the dataset, i.e., D. Each row of the table shows document details such as document IDs, titles, raw texts, etc, along with their topic memberships and topic-relevance scores. The document table is linked with the topic visualization. Hovering over a document square highlights the corresponding table row, and vice versa. Column fields may vary depending on datasets used. The raw texts can be long, so our system does not show them by default, but a row can be expanded to show the raw text when clicked. One challenge is that rendering all doc-

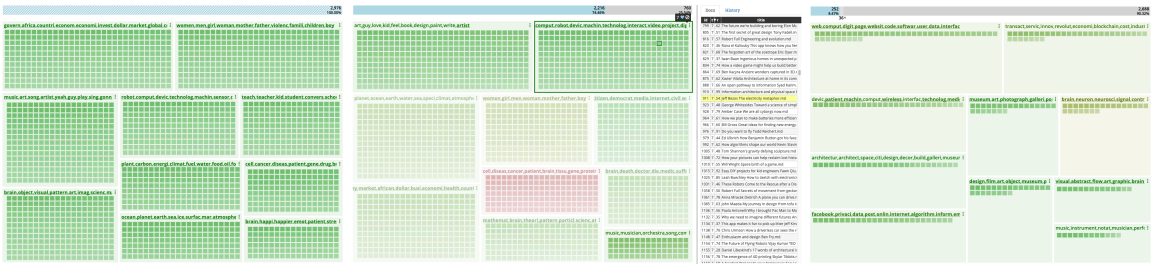


Figure 4.10: Exploring the TED dataset. The initial iteration (left) shows topic summary of all documents. After adding "art, technology" to the good-to-have list, the user upvotes an interesting document and downvotes another (middle). Documents are further sifted (right).

ument rows are impractical in our large-scale text analytics setting. To solve this, we use Clusterize.js[1] library to render currently visible rows only and reuse those HTML elements when the table is scrolled. Another challenge is navigating and scrolling through tens of thousands of rows. For easy navigation, when a document square in the main view is right-clicked, the document table automatically scrolls to the corresponding row.

History View

Fig. 4.9 shows the history view, which contains a stacked bar chart (left) and the keyword summary history (right). The stacked bar chart shows all the visualized status bars from previous iterations. It can reveal changes per iteration and if the sifting results became stable. The keyword summary history shows top keywords for retrieved documents at each previous iteration. Users can observe whether their interactions have resulted as expected.

4.5 Evaluation

In this section, we provide quantitative evaluation utilizing simulated user feedback with a labeled dataset. Also, we show use cases to illustrate the usefulness of TopicSifter for search space reduction using two datasets: a TED dataset and Twitter dataset.

[1]Available at: https://clusterize.js.org/

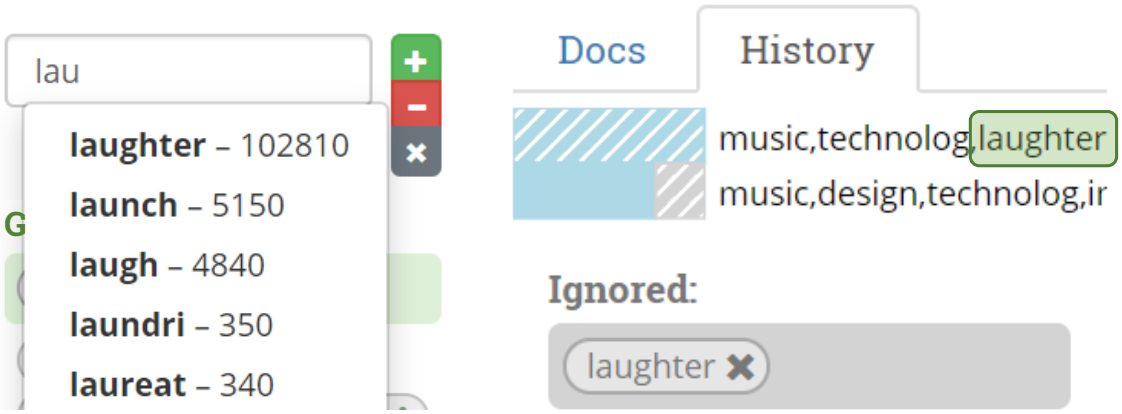


Figure 4.11: A stopword is detected from the history view.

4.5.1 Dataset Description

The 20 Newsgroup dataset[2] is a collection of 19.8K newsgroup documents partitioned into 20 categories. The size of dictionary is 128K. The TED talk transcript dataset[3] contains 2,896 documents that are transcribed from the English TED talk videos. The talks are about various topics including technology, education, etc. The size of dictionary is 18,275. The contents of the documents are spoken languages in a subtitle-like style. The twitter dataset[4] was originally explored by [83]. We use part of the data containing 500K tweets. After removing the documents with less than five words, we are left with 300K documents and 32.3K words. We applied the Porter stemming algorithm [84] for pre-processing and built the TF-IDF matrices for the datasets.

4.5.2 Quantitative Evaluation

In this section, we present results of a study simulating user input to test the effectiveness of our technique.

[2]Source: http://qwone.com/~jason/20Newsgroups
[3]Source: https://github.com/saranyan/TED-Talks
[4]Source: https://archive.org/details/twitter_cikm_2010

Experiment Setup

To simulate user relevance feedback, we used the 20 Newsgroup dataset which has category labels. Among 20 categories, we chose two labels "rec.sport.baseball" (989 documents) and "rec.sport.hockey" (993 documents) as relevant/true labels, which is about 10% of the entire dataset.

First, we entered "game, team, player, play", which were four most representative keywords from documents from the two categories, as initial target words. At each iteration, we select two documents or topics to give relevance feedback on (upvote or downvote based on the true label). We compared six strategies: 1) upvote two true documents ($+_d$), 2) upvote two true topics ($+_T$), 3) downvote two false documents ($-_d$), 4) downvote two false topics ($-_T$), 5) upvote a true document and downvote a false document ($\pm_d$), 6) upvote a true topic and downvote a false topic ($\pm_T$).

Results

Table 4.2 summarizes the performance of different feedback strategies at the 10-th iteration, averaged over three runs. We used four measures: precision, recall, F1-score (the harmonic mean of precision and recall), and PRES [85], which is a recall-oriented measure. For each strategy, we tried parameters from $\alpha \in \{0.4, 0.5, 0.6, 0.7\}$, $\beta \in \{0.4, 0.5, 0.6\}$, $\gamma \in \{0, 0.1, 0.2\}$ where $\alpha + \beta - \gamma = 1$ and chose the combination with best F1 score. For the sifting threshold, we used $\delta = 0.04$. All strategies converged after 4-6 iterations.

Simulating positive feedback showed higher recall and lower precision than negative feedback. Performing both positive and negative feedback showed better or comparable scores than performing only positive feedbacks, which advocates our novel negative targeting. In addition, positive topic-level feedbacks ($+_T$, $\pm_T$) outperformed the others in F1 and PRES scores. This validates that our topic-level relevance feedback is beneficial in search space reduction.

Table 4.2: Retrieval performance of relevance feedback strategies with their parameter settings. Scores are averaged over three runs. Best scores are highlighted.

	$+_d$	$+_T$	$-_d$	$-_T$	$\pm_d$	$\pm_T$
Precision	0.798	0.670	0.825	**0.880**	0.794	0.808
Recall	0.667	**0.827**	0.596	0.415	0.669	0.754
F1	0.727	0.740	0.692	0.564	0.726	**0.780**
PRES [85]	-0.095	0.680	-0.362	-2.527	-0.082	**0.703**
α	0.4	0.6	0.7	0.5	0.6	0.7
β	0.6	0.4	0.5	0.6	0.6	0.5
γ	0	0	0.2	0.1	0.2	0.2
δ	0.04	0.04	0.04	0.04	0.04	0.04

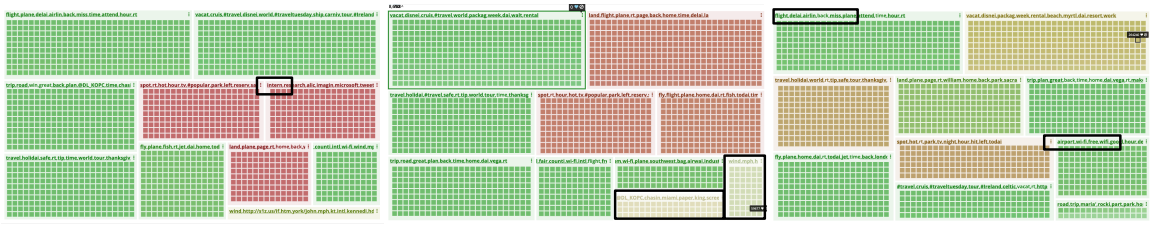


Figure 4.12: Exploring the Twitter dataset. After the initial iteration (left), some travel-related topics are found. After adding "intern" to the stopword list, irrelevant topics are still included (middle). Tweets are further sifted (right).

4.5.3 Use Case 1: Exploring Scraped Data

Jim is an art-major student who is also interested in technology. He is looking for technology areas where he can incorporate his artistic sense, and uses TopicSifter to retrieve talks related to his interest.

His visual exploration starts with an initial topic modeling that shows ten topics of all documents of the TED transcript dataset. From the main view (shown in Fig. 4.10(left)), he observes that a variety of topics are covered in the TED dataset, thus, he decides to focus on his interest, art and technology. He adds the keyword compound "art, technology" to the good-to-have keyword list and run the TopicSifter. He discovers some topics that are not interesting, such as biology/medicine or economy related ones, and downvotes them by clicking the topic cells. During the process, he finds out that a keyword "laughter" was the third-most frequent word in the TED dataset as he sees the history view from the

Figure 4.13: Initial user-input good-to-have keyword "#travel" and the keywords recommended by TopicSifter (left). The recommended keywords are also incorporated into the good-to-have list.

detail panel (shown in the upper-right of Fig. 4.11). He reminds that the TED dataset is based on the scripts of the talks, and the keyword "laughter" is usually used to describe audience's reaction in scripts. He adds the keyword to the list of stopwords so that it cannot influence the sifting process (Left of Fig. 4.11). As he proceeds, he sees a topic with keywords "market, africa, dollar" and downvotes it since it looks unrelated. Here, TopicSifter does not simply removes all document in the downvoted topic. It still retrieves target-relevant documents that was in the downvoted topic, accomplishing high recall. For example, documents titled "The surprising seeds of a big data revolution in healthcare", and "Tim Brown urges designers to think big" are both highly related to art and technology field, and were assigned to the "money" topic. They were survived by TopicSifter by taking account into overall relevancy. He finds "comput, robot" topic inteseting, inspects its documents in the table view interesting, and upvotes it (Fig. 4.10(middle)). In this topic, Jim finds out interesting topics and corresponding documents that contain contents about 3-D printer or human-computer interaction. Finally, he continues iterations until he is satisfied with his target documents about art and technology (Fig. 4.10(right)).

4.5.4 Use Case 2: Exploring Social Media Data

Now, we will follow the case of a marketer in a travel agency, Pam, who uses the proposed technique to sort out consumers' interests in travel experiences. Pam starts by loading the twitter dataset and looking at the initial topics.

Since Twitter is a social media platform, many tweets are about everyday life and emotions. For example, Pam sees that some topics include top keywords such as: "rt", "home", "day", and "today". She adds the keyword "#travel" to the good-to-have list to observe users' behavior using hashtag keywords about traveling on Twitter. As recommended good-to-have words pop up around the selected keywords, she selects relevant keywords among them such as "airline", "plane", "travel" and "vacation" to see broader user interests about traveling (Fig. 4.13). After a single sifting phase, she observes that a red (and thus less relevant) topic includes the keyword "intern" (rectangle in Fig. 4.12(left)). Many tweets included in it are comments about "internship" such as "Why are like 80% of the Poker-Road intern applicants from Canada? [...]". She finds it strange that a topic about internship is retrieved for travel related targets. As it turns out, the word "international", which is relevant to the targets, is stemmed to "intern", so tweets about internships are incorrectly identified as relevant. Pam adds "intern" to the stopword list to avoid this issue. After one iteration, the "intern" topic is removed (Fig. 4.12(middle)). She spots an unusual topic "wind,mph". Tweets in this topic are mostly automatically generated from a weather bot twitter account such as "HD: Light Rain and Breezy and 52 F at New York/John F. Kennedy [...]". Another topic "@DL_KOPC,chasin,miami" contains various spam messages such as advertisement for a trip to Miami. She downvoted these two topics to remove additional spamming tweets (black rectangles in Fig. 4.12(middle)). At the next iteration, there are many casual tweets such as "Family, food, games, and football. That's Thanksgiving." or "Just chatted w/ Jane Lindskold & husband Jim here at the airport. Very cool people.". Pam continues exploration to find out more specific tweets that represent customers' interests related to traveling (shown in Fig. 4.12(right)). One big travel-related concern is "flight

delay" as shown in the top-right topic in Fig. 4.12(right). Another interest is "free Wi-Fi" as shown in the bottom-right topic in Fig. 4.12(right). She starts designing travel packages that includes free WiFi options and flight delay insurances. The application helped her realize customer concerns and customize the agency's products.

4.6 Discussion

Iterative methods are computational methods that update approximate solutions over iterations. In general, iterative methods have some stopping criteria or stopping rules to terminate the methods, based on their objective functions or evaluation measures, e.g., when a score converges to a local minimum. Likewise, many visual analytics systems that adopt interactive machine learning or optimization methods utilize some form of measures to evaluate their tasks and application. These measures can be kept internally for monitoring; or can be shown to the users as charts (e.g., [32]) or some form of visual encodings (e.g., [86]) to inform users about the status of the current iteration. In our case, the relevancy scores of documents can be used as a measure. Unfortunately, our iterative retrieval approach not only updates the solution (which is the retrieved set of relevant documents), but also updates the target by which we measure the relevance scores of the documents. For this reason, comparing the relevance scores between iterations are meaningless if the target has been changed. That is, a higher relevance score in an iteration does not necessarily mean a better solution than a lower relevance score in another iteration. One naïve walkaround would be to compute the relevance scores of previously retrieved documents against the current target. However, this walkaround requires the system to store all historical results and calculate the relevance scores again at every iteration, which is not practical. Instead, for the TopicSifter prototype system, we decided to show retrieval status changes similar to membership changes in clustering. As explained in Fig. 4.9, the history view in the detail panel shows changes in retrieved documents the over iterations in the stacked bar chart. In addition, we use a colored triangle mark to indicate if a topic has changed much

from the previous iteration as in Fig. 4.7. These kinds of visual cues can guide the users' decision on when to stop the iteration (e.g., limited change between iterations)

4.7 Conclusion

In this chapter, we proposed a novel sifting technique to solve search space reduction problem interactively and iteratively. Our technique combined interactive target building and targeted topic modeling to sift through document collections and retrieve relevant document as many as possible. As a proof of concept, we built an interactive search space reduction system which offers tight integration between the visualization and the underlying algorithms.

CHAPTER 5
EMBIVIS: INTERACTIVE EXPLORATION AND DEBIASING OF WORD EMBEDDING

In this chapter, we present an interactive way to examine and mitigate biases in word embeddings. Word embeddings are shown to reflect human biases such as gender or racial stereotypes. Our interactive approach allows users to form interpretable attribute axes to examine biases within word embeddings and de-bias them if needed. This chapter is adapted from our ongoing work.

5.1 Introduction

As machine learning algorithms are deployed to more decision making systems for wider applications and domains, fairness and bias have become important issues in the artificial intelligence (AI) community. Many cases of biases in deployed systems have been reported and studied over the past decades. For instance, a widely used criminal risk prediction system, which is used for parole or sentencing decisions, is shown to put higher risk on African American defendants compared to Caucasian defendants with the same record [87]. Another example is Google Image Search, where gender proportions are exaggerated in the search results (i.e., male-dominated occupations tend to have even more men than expected) [88]. This is not surprising because AI systems are inherently data-driven, and thus, they reflect stereotypes and biases embedded within the real-world data. To this end, there have been many works that try to directly deal with the bias for fair and socially responsible AI. These techniques aim to either remove bias by transforming the training data before training (pre-processing), modifying algorithms to consider bias and fairness during the training (in-processing), or debiasing the results after the training (post-processing).

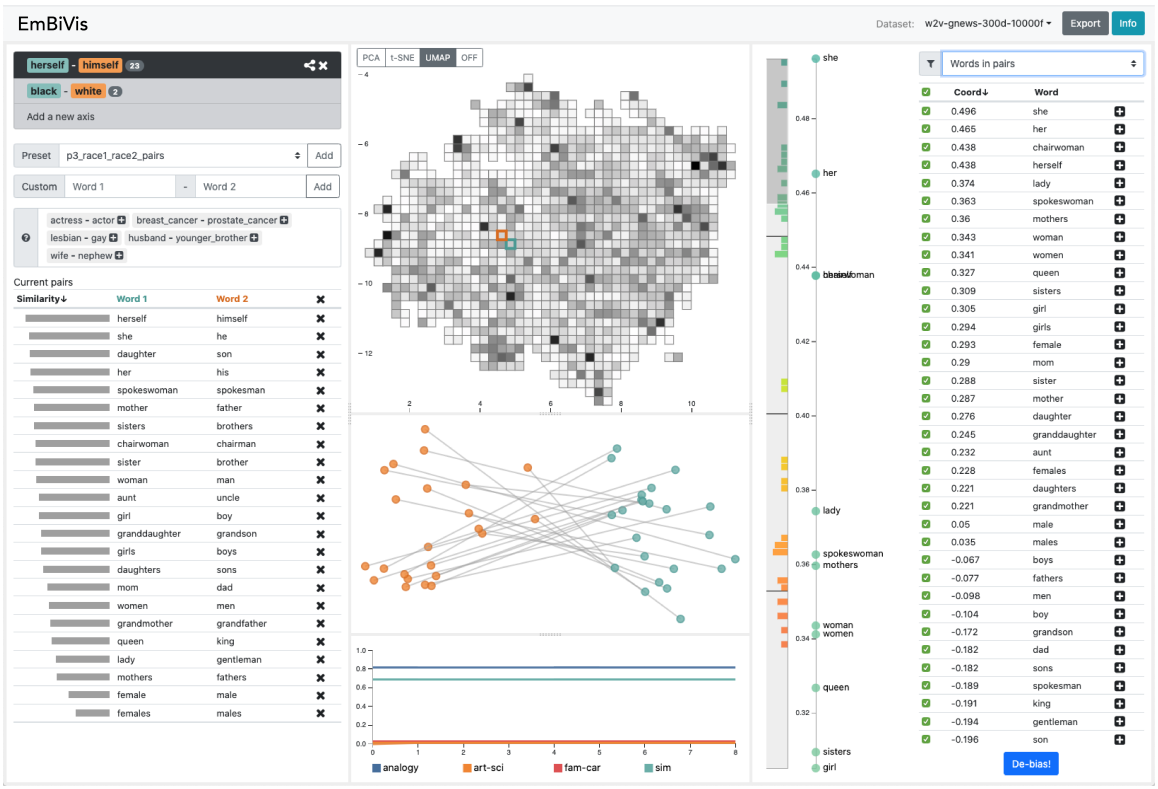


Figure 5.1: Our system has (1) the control bar (top), (2) the attribute panel (left), (3) the visualization panel (middle), and (4) the bias panel (right). The control bar on the top contains the dataset dropdown menu, the export button, and the tutorial button. The attribute panel has the attribute axis list, the axis module to update an axis, and the table showing analogy pairs for the selected attribute axis. The visualization panel contains three views: the global view, the pair view, and the quality monitoring view. The bias panel consists of the axis projection view, the word table, and the debias button. Views are connected via brushing and linking.

Natural language processing (NLP) tasks are no exception to such bias problems. Huge text corpora that many state-of-the-art NLP techniques are trained on inevitably contain human biases. Machine translation task is an example. When translating sentences from gender-neutral languages such as Hungarian to English, Google Translate tends to prefer male nouns, specifically for sentences about STEM (Science, Technology, Engineering and Mathematics) jobs [89]. For example, the same Hungarian pronoun is translated differently: '*he* is an engineer' vs. '*she* is a nurse'. Another example is coreference resolution, which is the task to identify mentions referring to the same entity in a text. In a recent work [90], gendered pronouns are more likely to be linked to pro-stereotypical entities. For

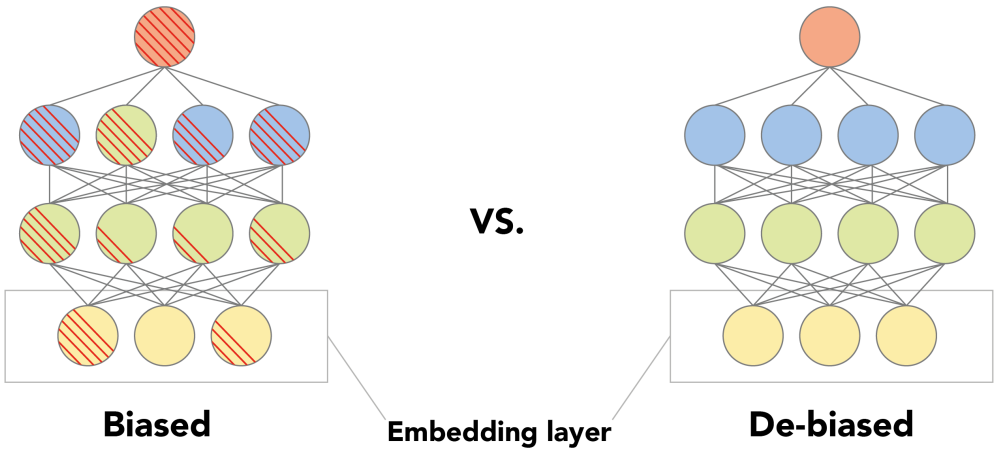


Figure 5.2: Mitigating the biases in word embeddings reduces the risk of propagating biases into downstream applications that utilize trained word embeddings.

instance, for the sentence 'The physician hired the secretary because *he* was [...]', 'he' is more likely to reference 'physician'. On the other hand, if 'he' is replaced with 'she' in the same sentence, 'secretary' will be more likely to be associated with 'she'.

Among many NLP tasks, we focus on biases in neural word embeddings. Word embeddings are widely used in NLP applications and can be considered as the building blocks for numerous neural NLP techniques. Generally, the first layer (from the bottom) of deep neural networks for NLP is an embedding layer which transforms a discrete token (such as a word or a character) to a continuous vector. In many cases, a pre-trained word embedding is used as the static first layer or as an initialization for the first layer in deep neural networks. Therefore, by mitigating the biases in word embeddings, we can reduce the risk of propagating biases into downstream applications that utilize trained word embeddings (Fig. 5.2). For instance, it has been shown that using a debiased word embedding in a coreference resolution algorithm reduces bias in the coreference resolution task [90].

Most of the algorithms for debiasing word embeddings are based on the seminal work of Bolukbasi et al. [91]. These algorithms can be largely described as a two-step process. The first step is to identify attribute (or bias) subspace (e.g., a gender subspace). The

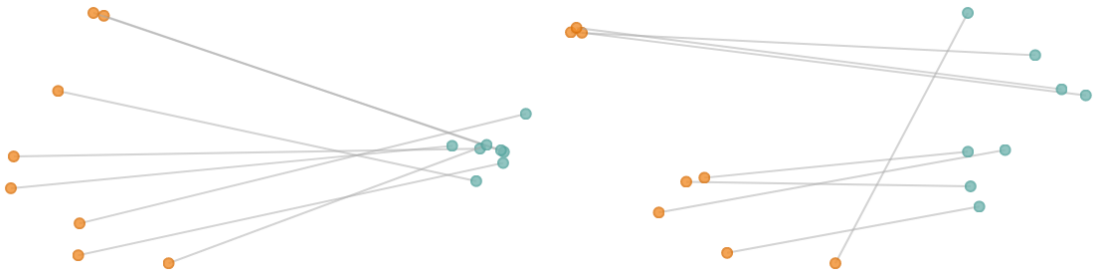


Figure 5.3: Eight pairs of words defining the gender attribute visualized using embeddings trained on Wikipedia (left) and Google News (right) corpora. Green dots represent female-related words and orange dots represent corresponding male-related words. Gender-defining pairs from [91] are used.

subspace is usually represented as a single vector, which is computed based on pairs of words that can define the attribute (e.g., 'woman-man', 'female-male', 'she-he', and so on). The second step is to transform word vectors using the identified subspace so that the amount of bias in the resulting embedding is reduced. For example, remove gender portion from word vectors for gender-neutral words, making the word vector orthogonal to the bias subspace. The main differences between debiasing algorithms come from how to compute bias subspaces, which attribute-defining pairs to choose (from step 1), how to transform the word vectors, and which words to transform (from step 2).

We argue that human-in-the-loop approaches can be beneficial in choosing attribute-defining pairs and words-to-be-transformed. Fig. 5.3 shows the same set of 8 pairs of words visualized using different embeddings trained on Wikipedia (left) and Google News (right). While the pair difference vectors are similar to each other in the Wikipedia embedding, one pair ('female-male') is vastly different from the other pairs in the Google News embedding. In this case, removing the different pairs from the defining set may result in finding a better gender subspace. This illustrates the importance and the need for an interactive approach to solve the debiasing problem. Instead of blindly applying a state-of-the-art debiasing algorithm, we advocate an interactive and exploratory approach where users' downstream applications and domain contexts can be incorporated into the analytic process.

In this chapter, we propose a novel human-in-the-loop framework to debias word em-

beddings. Our approach allows users to construct attribute axes, visually examine attribute biases within word embeddings, and iteratively debias them, realizing highly customized and user-tailored debiasing. To test our approach, we build EmBiVis, a web-based visualization prototype system, which will be open-sourced later.

The primary contributions of this work are:

- A novel human-in-the-loop technique to formulate attribute axes, assess attribute biases, and debias word embeddings, iteratively;
- A visual analytic prototype system, EmBiVis, that supports interactive exploration of biases in word embeddings and guides interactive debiasing; and
- A use case and experiments that demonstrate the effectiveness of our approach.

5.2 Related Work

5.2.1 Debiasing Word Embedding

The most widely known work for word embedding debiasing is Bolukbasi et al. [91]. In this work, the authors find that even the state-of-the-art embeddings exhibit gender stereotypes. For example, 'man' is to 'computer programmer' as 'woman' is to 'homemaker'. To mitigate this, they propose a post-processing approach to debias word embeddings. To debias word embedding for gender bias, they first 1) identify gender subspace using a list of gender-specific words and then 2) remove the bias subspace from gender-neutral words (hard) or apply linear transformation to all words to minimize bias in gender-neutral words (soft). Manzini et al. [92] extend the work of Bolukbasi et al. [91] to include other attribute biases such as religion and race, separately. Gonen and Goldberg [93] criticize that this line of work does not entirely eliminate the gender bias from the embeddings, but rather hides the bias. There are other extensions such as [94] which support joint debiasing of multiple attributes and [95, 96, 97]. More recently, Karve et al. [98] propose the conceptor debiasing

method, which is the current state-of-the-art. This method transforms word vectors using a debiasing conceptor matrix and supports multiple attributes simultaneously.

Contrary to the above post-processing methods that debias word embeddings after they are already trained, Zhao et al. [99] try to learn attribute-neutral word embeddings (in-processing approach). They introduce a gender-neutral variant of GloVE [81] called GN-GloVE that isolates gender information in certain dimensions while keeping the other dimensions neutral to gender. As a result, the gender dimensions in the GN-GloVE embedding becomes interpretable and can be used for gender-specific tasks. Another direction is transforming the training corpus to reduce bias (pre-processing). Brunet et al. [100] trace the origins of bias in training documents, which are then either perturbed or removed before training. In our work, we take a post-processing approach to debias word embeddings. While in-processing and pre-processing approaches have their own merits, they are not easily usable or applicable to non-NLP experts. Rather, we target layusers who utilize pre-trained word embeddings in their domain applications and want to easily debias them according to their own needs.

There have been attempts to study biases in contextualized word embeddings [101, 102] such as ELMo [103] and BERT [104]. For example, Zhao et al. [101] mitigates gender bias in ELMo by augmenting training corpus by swapping gender-specific entities (pre-processing) and neutralizing word vectors (post-processing).

In terms of attributes for debiasing word embeddings, gender [105, 90, 106, 107, 99, 100, 101, 108, 109, 93, 97, 96, 94, 95, 98] has been most frequently studied, and there are a number of works on race [105, 92, 96, 94, 98], religion [92, 94], nationality [96], and age [96]. We also categorize related works by downstream applications tasks: document classification [108], coreference resolution [90, 99, 101, 102] language model [106, 109], machine translation [107], and sentence representation learning [105]. Note that we only include debiasing works that use word embeddings in their algorithms. For a more general review of bias literature in NLP or AI, refer to [110, 111, 112].

5.2.2 Assessing Biases in Word Embedding

Accurate quantification of biases is an important task for fairness in AI, especially for the evaluation of debiasing techniques. The most widely used metric for biases in word embedding is Caliskan et al. [113]. This work introduces a new method for evaluating bias: the Word Embedding Association Test (WEAT), which replicates Implicit Association Test (IAT) [114] using word embedding relationships. To simply put, WEAT compares two target sets (attribute-defining, e.g., male or female-related words) with respect to two attribute sets of words (family or career-related words). Their result shows that females are more associated with family (or art) and males with career (or science) (gender bias). They also find that European American names such as 'Brad, Brendan, and Geoffrey' are more likely to be closer to pleasant than to 'unpleasant' than African American names such as 'Darnell, Hakim, and Jermaine' (racial bias). More recently, Ethayarajh et al. [95] argue that WEAT overestimates bias and can be manipulated by changing the attribute word sets. To mitigate this, they propose a new metric called Relational Inner Product Association (RIPA).

Garg et al. [115] compare average distances from occupation words (and selected adjectives) to female words vs. to male words as well as distances from occupation words (and selected adjectives) to hispanic vs. asian vs. white last names. Gonen and Goldberg [93] suggests an implicit bias test using k-means clustering and SVM classifier.

5.2.3 Interactive Visual Analysis on Word Embedding

There are a number of visual analytics systems for exploring word embeddings [116, 117, 118, 119, 120]. Most of these tools project word embeddings onto 2D or 3D scatterplots using dimension reduction techniques such as Principal Component Analysis (PCA) [121], t-Distributed Stochastic Neighbor Embedding (t-SNE) [122], and Uniform Manifold Approximation and Projection (UMAP) [123]. Embedding projector [120], which is deployed in Tensorflow, supports local neighborhood lookup. Parallax [118] utilizes algebraic formula (e.g., $sum(\overrightarrow{he}, \overrightarrow{his})$: sum of two-word vectors for words 'he' and 'his') to form se-

mantic axes and project word vectors along with them for improved interpretability. Liu et al. [116] extend this line of work to general embeddings to better interpret latent spaces. Alternatively, Heimerl and Gleicher [117] propose a novel 1D visualization and a heatmap visualization that is tailored to specific tasks. More closely to our work, Word Embedding Visual Explorer [119] visualizes semantic relationships such as word pairs, forms an axis based on the pairs, and shows words that are at the extreme ends of the axis.

While these works focus on the general exploration of word embeddings, there are two use cases that visualize bias to some extent. Parallax [118] visualizes profession words used in [91] onto a male x-axis (computed as $avg(\overrightarrow{he}, \overrightarrow{his})$), and a female y-axis (computed as $avg(\overrightarrow{she}, \overrightarrow{her})$). As a result, 'nurse, dancer, maid' are projected closer to the female axis, while 'boss, captain, commander' are closer to the male axis (gender bias in occupation words). Similarly, in Liu et al. [116], 'pink, wedding' are located near female names and 'director, mayor, victory' near male names when projected onto a gender attribute vector. In addition to visualizing biases, our system supports interactive refinement of bias subspace, quantified bias assessment over iterations, and debiasing.

5.3 Interactive Debiasing of Word Embedding

Word embeddings are shown to have undesirable associations with respect to sensitive attributes such as gender or race [113, 91]. To solve this, many automatic algorithms are introduced to debias word embeddings in recent years [91, 92, 94, 98]. These automatic techniques require pre-selected sets of attribute-defining word pairs and attribute-neutral words as their input, and the quality of debiasing is highly dependent on these selections. However, the selections should be customized according to the needs of the end-users and should vary depending on the used corpus, the application tasks, or the context of the domain for better results. Therefore, we introduce an interactive and iterative approach for human-in-the-loop debiasing of word embeddings. In this section, we formulate the problem of debiasing word embedding, explain our interactive workflow to solve it, and

Table 5.1: Key notations used in Chapter 5.

Notation	Description
m	The dimension of the given word embedding
n	The number of words
k	The number of attribute-defining word pairs
w_i	The i-th word, $1 \leq i \leq n$
V	The set of words, $V = \{w_1, \cdots, w_n\}$
V_S	The set of attribute-specific words, $V_S \subset V$
V_N	The set of attribute-neutral words, $V_N \subset V$
P	The set of attribute-defining word pairs, $P = \{(w_{1_1}, w_{1_2}), \cdots, (w_{k_1}, w_{k_2})\}$
$\boldsymbol{w}_i$	The word vector of the i-th word $\in \mathbb{R}^{m \times 1}$
$\boldsymbol{W}$	The word embedding matrix $\in \mathbb{R}^{m \times n}$, i.e., $\boldsymbol{W}_{.i} = \boldsymbol{w}_i$
$\boldsymbol{N}$	The matrix of attribute-neutral words $\in \mathbb{R}^{m \times \lvert V_N \rvert}$
$\boldsymbol{B}$	The bias subspace defined by P, $\boldsymbol{B} = span(\{\boldsymbol{w}_i - \boldsymbol{w}_j \lvert (w_i, w_j) \in P\})$
$\lvert\lvert \cdot \rvert\rvert_F$	The Frobenius norm
$\boldsymbol{A}_{i\cdot}$	The i-th row of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{\cdot i}$	The i-th column of matrix $\boldsymbol{A}$
$argmax(\boldsymbol{a})$	The index of the largest element in vector $\boldsymbol{a}$

then describe our algorithms.

5.3.1 Problem Formulation and Algorithm Workflow

Given a pre-trained word embedding $\boldsymbol{W}$, the goal is to obtain a new word embedding $\boldsymbol{W^*}$ with less bias and comparable quality. That is, we want the new embedding $\boldsymbol{W^*}$ such that the word vectors of attribute-neutral words are unbiased, i.e., $\forall w \in V_N,\ \boldsymbol{w} \perp \boldsymbol{B}$, where V_N is the set of attribute-neutral words, $\boldsymbol{w}$ is the corresponding word vector of the word w, and B is the bias subspace. Notations used in the chapter are summarized in Table 5.1.

Our approach is to update $\boldsymbol{W}^{(t)}$ over iterations ($t = 1, \cdots, T$) based on user input of attribute-defining pairs $P^{(t)}$ and attribute-neutral words $V_N^{(t)}$. At each iteration, we first compute attribute subspace $\boldsymbol{B}^{(t)}$ using the attribute-defining pairs $P^{(t)}$. Then we debias the embedding $\boldsymbol{W}^{(t)}$ with respect to $\boldsymbol{B}^{(t)}$, based on the selection of attribute-neutral words $V_N^{(t)}$ and attribute-specific words $V_S^{(t)}$. Fig. 5.4 outlines our workflow. We describe each step in

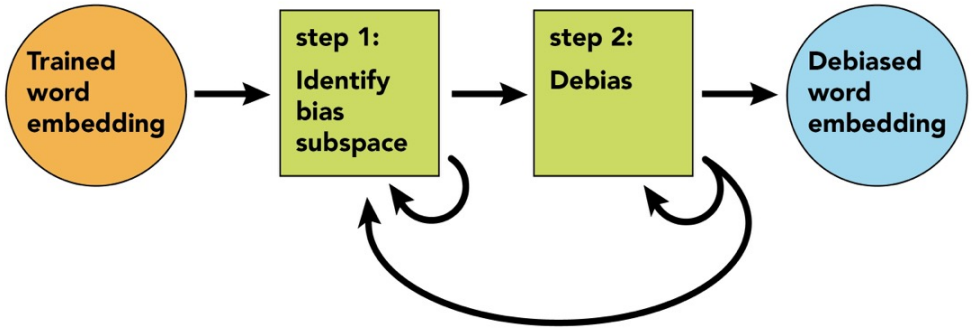


Figure 5.4: A diagram outlining our algorithm workflow. Given a trained word embedding, our algorithm identifies bias subspace (step 1) and debias the embedding (step 2) over iterations. Each step can be repeated upon user feedback.

detail in the following.

5.3.2 Step 1: Formulate Attribute Subspace

We compute the attribute subspace $\boldsymbol{B}^{(t)}$ at the t-th iteration based on the attribute-defining word pairs $P^{(t)}$ entered by the users. For each pair $p_i = (w_{i_1}, w_{i_2}) \in P^{(t)}$, the difference vector $\boldsymbol{w_{i_1}} - \boldsymbol{w_{i_2}}$ constructs the attribute subspace $\boldsymbol{B}$. That is,

$$\boldsymbol{B} = span(\left[\boldsymbol{w_{1_1}} - \boldsymbol{w_{1_2}}, \quad \cdots, \quad \boldsymbol{w_{k_1}} - \boldsymbol{w_{k_2}}\right]).$$

We perform singular value decomposition on the stacked matrix $\hat{\boldsymbol{B}}$ to get $\hat{\boldsymbol{B}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. For practicality, we set the attribute subspace as the span of the first r columns of the left singular matrix $\boldsymbol{U}$.

Pair Recommendation After the subspace is computed, we provide pair recommendation for users to help them populate and construct the pair set for next iteration, $P^{(t+1)}$. To do this, we first calculate the representative vector $\boldsymbol{b}$ of the subspace by equation $\boldsymbol{b} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i$ where σ_i denotes the i-th singular value, and $\boldsymbol{u}_i$ denotes the i-th left singular vector. We recommend pairs whose difference vectors are closest to $\boldsymbol{b}$. In other words, we recommend

top word pairs satisfying $(w_{i'}, w_{j'})$ where $i', j' = \arg\min_{i,j} dist(\boldsymbol{w}_i - \boldsymbol{w}_j, \boldsymbol{b})$. We use cosine similarity as distance measure $dist(\cdot)$. Calculating the desired pair (i', j') takes $\mathcal{O}(n^2)$ time because all pair-wise distance should be calculated. However, this is undesirable for our real-time interaction system. For efficiency in real-time interaction, we filter candidate words. We consider only words that are close to currently selected pairs. We set threshold $\delta = 1$ and filter out words which have larger distance than δ from words in currently selected pairs.

5.3.3 Step 2: Debias

Given the user's selection of attribute-specific words $V_S^{(t)}$ and attribute-neutral words $V_N^{(t)}$, the debiasing step transforms word vectors to reduce bias with respect to the bias subspace computed at step 1. Our system adopts two debiasing algorithms from previous works [91, 98]. However, any post-processing debiasing algorithm can be used as a replacement if needed.

The first option is *HardDebias*, which was proposed in [91].

HardDebias at iteration t:

$$\boldsymbol{w}^{(t)} = \frac{\boldsymbol{w}^{(t-1)} - \boldsymbol{b}}{||\boldsymbol{w}^{(t-1)} - \boldsymbol{b}^{(t)}||} \text{ for } \forall w \in V_N^{(t)}$$

$$\boldsymbol{w}^{(t)} = (\boldsymbol{\mu} - \boldsymbol{\mu_B}) + \sqrt{1 - ||\boldsymbol{\mu} - \boldsymbol{\mu_B}||^2} \frac{\boldsymbol{w}^{(t-1)} - \boldsymbol{\mu_B}}{||\boldsymbol{w}^{(t-1)} - \boldsymbol{\mu_B}||} \text{ for } \forall w \in V_S^{(t)}$$

where $\boldsymbol{\mu}$ is the average vector of word w and its counterpart and $\boldsymbol{\mu_B}$ is $\boldsymbol{\mu}$'s component in the subspace.

The second option is *ConceptorDebias*, which was proposed from [98].

ConceptorDebias:

A conceptor matrix, C, is defined as an identity transformation that minimizes

$$\|W - CW\|_F^2 + \alpha^{-2}\|C\|_F^2. \tag{5.1}$$

Choosing G, which is the negating conceptor, that minimizes

$$\|(G(X-Y))^T G(A-B)\|_F = \|(X-Y)^T G^T G(A-B)\|_F$$

will result in mitigating the most prominent bias direction.

5.4 System

To demonstrate the effectiveness of our interactive debiasing approach, we build a visual analytic prototype system, *EmBiVis*. EmBiVis consists of a web-based visualization interface using D3.js, a backend module that processes all embedding computation in Python, and the Flask framework and the Flask-SocketIO server connecting the two. In this section, we describe the design and functionality of EmBiVis.

5.4.1 Design Goals, Challenges, and Tasks

Our design goal is to help users assess various attribute biases in word embeddings and debias them for later use. Our target user group are those who are not necessarily NLP experts but want to use pre-trained word embeddings in their downstream NLP applications in which certain biases may negatively impact the outcome. For these users, there are many challenges in utilizing previous debiasing techniques. First, most of the existing techniques require users to provide word pairs that define an attribute (e.g., 'woman-man' and 'female-male' for the 'gender' attribute) beforehand. Although some works offer predetermined sets of word pairs for a few attributes such as gender and race, selecting a good set of attribute-defining word pairs is difficult without exploration or prior knowledge on the training corpus or the used preprocessing method. Similarly, most of the existing techniques require users to select words to be debiased, e.g., attribute-neutral words such as 'nurse' and 'surgeon' for the 'gender' attribute. However, users may want to explore which words are biased and how biased these words are before debiasing them. Also, the selection may depend on the users' applications or domains. Finally, many works focus on debiasing

a single attribute. But they may want to remove biases for multiple attributes. In addition to supporting multiple attributes, we need to ensure that debiasing on an attribute does not make the embedding more biased on another attribute.

To support users in the overarching task of debiasing an embedding, EmBiVis supports the following tasks:

Task 1 *Show global distribution of words*

Task 2 *Show local relationships between words*

Task 3 *Identify a subspace (or an axis) describing an attribute*

Task 4 *Allow users to edit an attribute axis*

Task 5 *Show relationships between attribute-defining word pairs*

Task 6 *Project words on an attribute axis*

Task 7 *Allow users to select words for debiasing*

Task 8 *Monitor the quality of embedding over iterations*

Task 9 *Support multiple attributes*

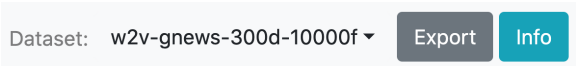
Tasks 1-2 support data understanding through an interactive exploration of word vectors (both globally and locally). Tasks 3-5 and Tasks 6-8 support Step 1: Formulate Attribute Subspace and Step 2: Debias (of Section 3.3), respectively. Lastly, Task 9 supports debiasing by multiple attributes.

5.4.2 System Overview

The system design is shown in Fig. 5.1. EmBiVis has (1) the control bar, (2) the attribute panel, (3) the visualization panel, and (4) the bias panel. The control bar on the top contains overall controls for the tool. The attribute panel supports the attribute axis formulation (Section 5.3.2). The visualization panel shows various views to support the exploration of word vectors and constructed axes. Lastly, debiasing is done in the bias panel (Section 5.3.3).

For all views, we color one side of an attribute as green and the other as orange. The width and height of each view or panel are adjustable by dragging the divider to allocate more or less space to it.

5.4.3 Control Bar



The control bar on the top contains a dropdown menu to select word embeddings, the export button to save the resulted embedding after debiasing, and the information button to show a pop-up window with tutorial instructions. At the start, a user chooses an embedding to debias in the dataset dropdown. After the debiasing process is done, the user can export embedding for later use by clicking the export button.

5.4.4 Attribute Panel

The key functionality of the attribute panel is Step 1: formulating axis subspace (Section 5.3.2). The attribute panel has the attribute axis list, the axis module to update an axis, and the table showing analogy pairs for the selected attribute axis (supporting Tasks 3-5, 9).

Attribute Axis List

To support multiple attribute debiasing (Task 9), the users can add, modify, or delete attribute axes using the attribute axis list. In the list, each item represents an axis with its most representative word pair and the number of attribute-defining word pairs shown. When an axis is selected, it is highlighted with a black background. The selected axis can be dupli-

cated or deleted using the buttons on the right side of the list item. To formulate a new axis, the user can click the 'Add a new axis' button.

Axis Update Module

The axis update module offers three ways to add attribute-defining pairs (Task 4). First, we collected attribute-defining pairs from previous works covered in Section 3.2. The user can select the pair set from the dropdown menu (top). Second, the user can manually enter word pairs (middle). Third, based on currently added attribute-defining pairs and the recommendation algorithm in Section 5.3.2, EmBiVis suggests the top five word pair candidates (bottom). Additionally, when the user enters a word in the custom input (middle), EmBiVis recommends additional word pairs containing the entered word. Hovering over a suggested pair highlights the pair with green and orange colors and updates the global view and the pair view in the visualization panel (Task 5). This allows the user to learn the suggested pair's global position as well as its relationships with the current set of attribute-defining pairs. The user can add a suggested pair using its plus button.

Pair Table

The pair table shows the attribute-defining word pairs for the currently selected attribute axis (Task 5). Each row shows a word pair (w_i, w_j) and the cosine similarity between the pair difference vector $(\boldsymbol{w}_i - \boldsymbol{w}_j)$ and the axis vector $\boldsymbol{b}$. A higher similarity means that the pair is more representative of the selected axis. The user can remove an attribute-defining pair by clicking the 'x' button on the corresponding row (Task 4). Rows are sortable by similarity or words. Similar to the recommended pairs, hovering over a row highlights

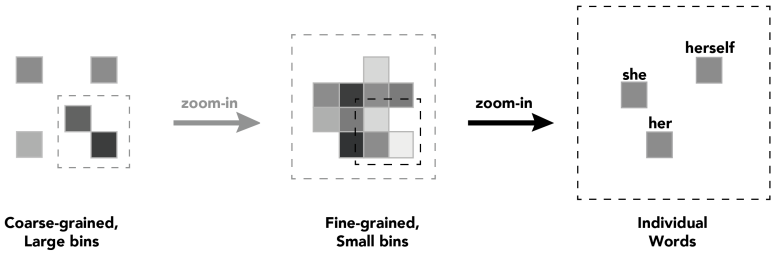


Figure 5.5: Zooming-in functionality of the binned scatterplot in the global view. Incrementally zooming in on a region increases the details (words) shown.

the pair with green and orange colors; highlights the corresponding in words the global view and the pair view in the visualization panel (Task 5). This allows the user to learn the selected pair's global position as well as its relationships with other attribute-defining pairs.

5.4.5 Visualization Panel

The visualization panel offers various visualizations that help the exploration of word embeddings, examination of pair vectors, and embedding quality monitoring. The visualization panel contains three views: the global view, the pair view, and quality monitoring view (supporting Tasks 1-2, 5, 8).

Global View

The global view visualizes words in a binned scatterplot to show the global distribution of words and a word's relationship with other words (Tasks 1-2). While the other views in the system focus on the aspects that are directly related to attribute subspace, this view shows an overview of the words using the high dimensional relationships. Because the size of the dictionary is not practical to visualize individual words in the scatterplot, we use a binned scatterplot with zoom-in. In Fig. 5.5 (left), we start with large bins with coarse granularity where each bin is represented as a square. Zooming-in will show smaller bins with finer granularity, which is still represented as the same-sized squares (Fig. 5.5 (middle)). At the

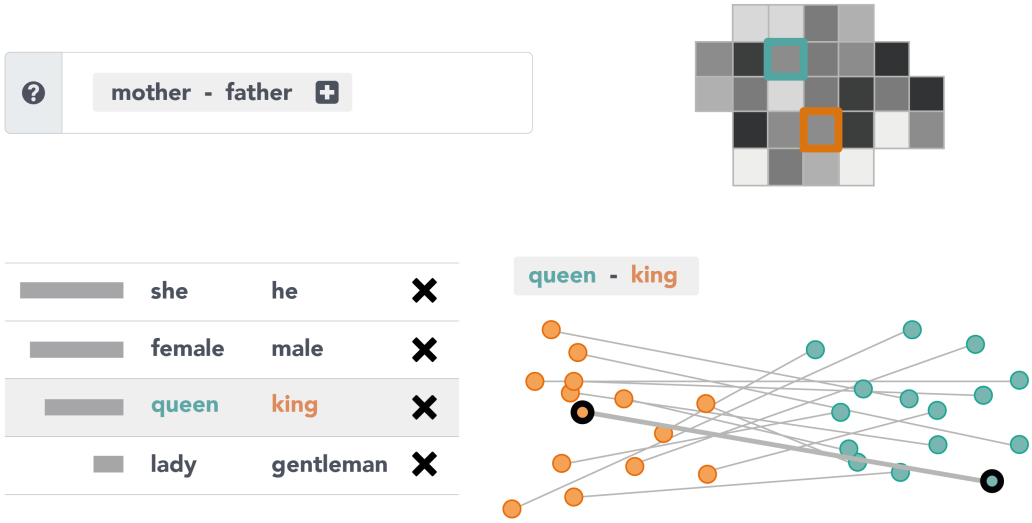


Figure 5.6: Illustrating the brushing and linking functionality of our system. Hovering over a word pair in the table shows the corresponding pair in the 2D projection.

finest level of zooming in, we show individual words with labels (Fig. 5.5 (right)). We utilize the quadtree algorithm for latency. We provide three dimension reduction options: PCA, t-SNE, and UMAP.

Pair View

The pair view shows the 2D projection of attribute-defining pairs of the selected axis (Task 5). One advantage of our interactive approach over automated debiasing algorithms is that the user can visually examine the saliency of the added attribute-defining pairs as illustrated in Fig. 5.3. The user can examine the pair view and discover any sub-trends within the pairs. For example, family related gender pairs (e.g., mom-dad, daughter-son) and the others (e.g., female-male, he-she) tend to form different clusters. Also, the pair view, the pair table, the pair recommendation list, and the global view are connected via brushing and linking. Utilizing the brushing and linking, the user can discover interesting discovery such that pairs that have similar difference vectors are not necessarily near each other in the original space.

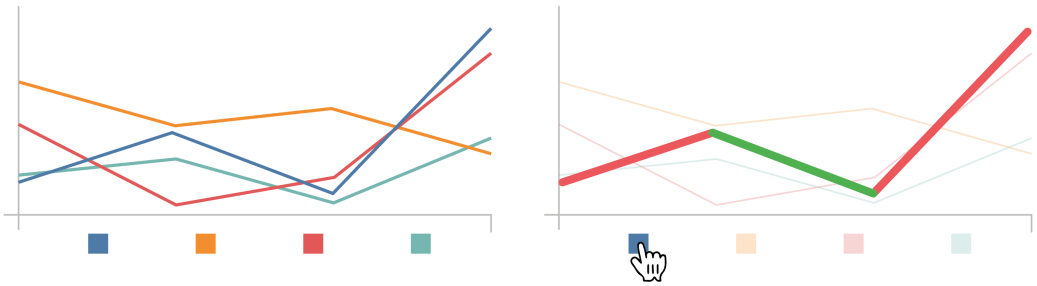


Figure 5.7: The quality monitoring view (left). Hovering over a line or a legend highlights the metric using green (improvement) and red (deterioration) colors (right).

Quality Monitoring View

The quality monitoring view shows a temporal line chart of embedding quality measures and bias scores (Task 8). For the general embedding quality, we utilize word analogy measure and similarity ranking measure, which are widely used in the NLP community. For the bias metric, we adopt [95] using attribute sets from Caliskan et al. [113]. In Fig. 5.7, hovering over a line or its legend highlights the selected measure and grays out the others. Each segment (between iterations) in the highlighted line is colored as green (or red) if the metric is improved (or not) at the iteration.

5.4.6 Bias Panel

The key functionality of the bias panel is Step 2: debiasing (Section 5.3.3). The bias panel consists of the axis projection view, the word table, and the debias button (supporting Tasks 6-7).

Axis Projection View

The axis projection view visualizes the coordinates of the word vectors projected onto the selected bias axis (Task 6). Fig. 5.8 (left) shows the axis projection view, which consists of a bar chart and a 1D plot. For both the bar chart and the 1D plot, top is the green extreme

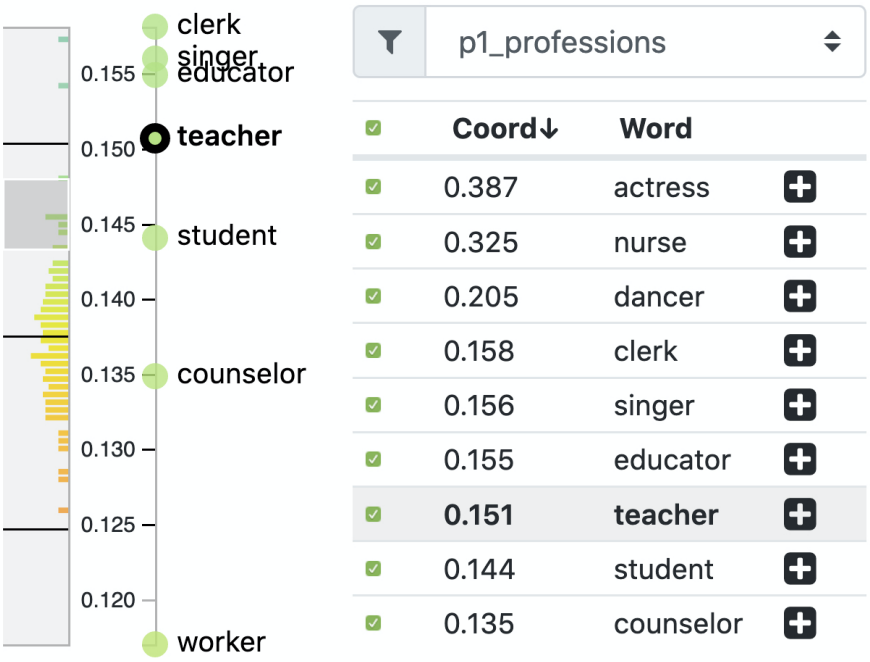


Figure 5.8: The axis projection view (left), the word filter (top-right), and the word table (bottom-right) in the bias panel.

and the bottom side is the orange extreme of the selected bias axis. We use a focus+context approach to show the distribution of projected words in a histogram. Upon brushing the bar chart, the selected section will be drawn in the 1D plot. For example, the focused region in Fig. 5.8 shows that 'clerk, singer, educator' are biased toward female extreme. Hovering over a word in the axis projection view highlights the corresponding word row in the word table.

Word Table

The word table allows the user to choose which words to debias. (Task 7) Each row represents a word, and its coordinate on the bias axis is also shown. The user inspects the words in the extreme. If they are attribute specific (e.g., actress), the user can uncheck it not to debiase it and consider adding it to the attribute-defining pair with the opposite word (e.g., 'actor'). If there are biased attribute-neutral words, the user can use the checkbox to debias it. On top of the word table, there is a filter with selected word groups. We offer Wordnet

based categories (e.g., nouns, nouns about people, adjectives, etc.) as well as word groups used in the previous works (e.g., professions, family words, etc). When a filter is selected, the word table and the axis projection view are updated with only the words that are in the group. Hovering over a word row highlights the corresponding word in the axis projection view.

5.5 Use Case

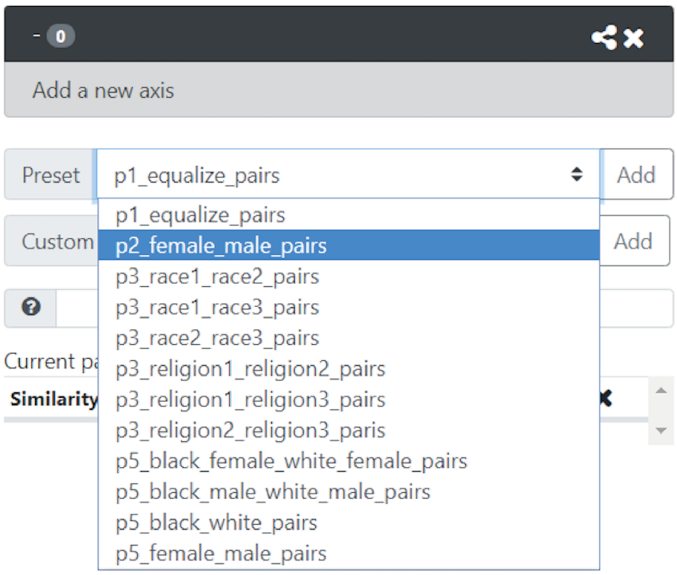


Figure 5.9: A user selects a set of pre-defined pairs.

We describe our system's functionalities with an example scenario. A user is interested in the implicit gender bias in the word embedding model, *word2vec*. The user's visual explorations start with initial empty views. The user selects a set of pre-selected pairs that defines gender attribute (Fig. 5.9). Among the added preset, the user finds out that the *maiden-bachelor* pair does not align well with other added pairs (Fig. 5.11a). After removing the *maiden-bachelor* pair with other less interesting pairs, the user examines recommended pairs. The user likes the pair *aunt-uncle*, and add it to to refine the gender subspace (Fig. 5.11b). In the bias panel (Fig. 5.10), the user selects *verb.emotion* category from the word filter to see how implicit gender bias is associated with emotional verbs.

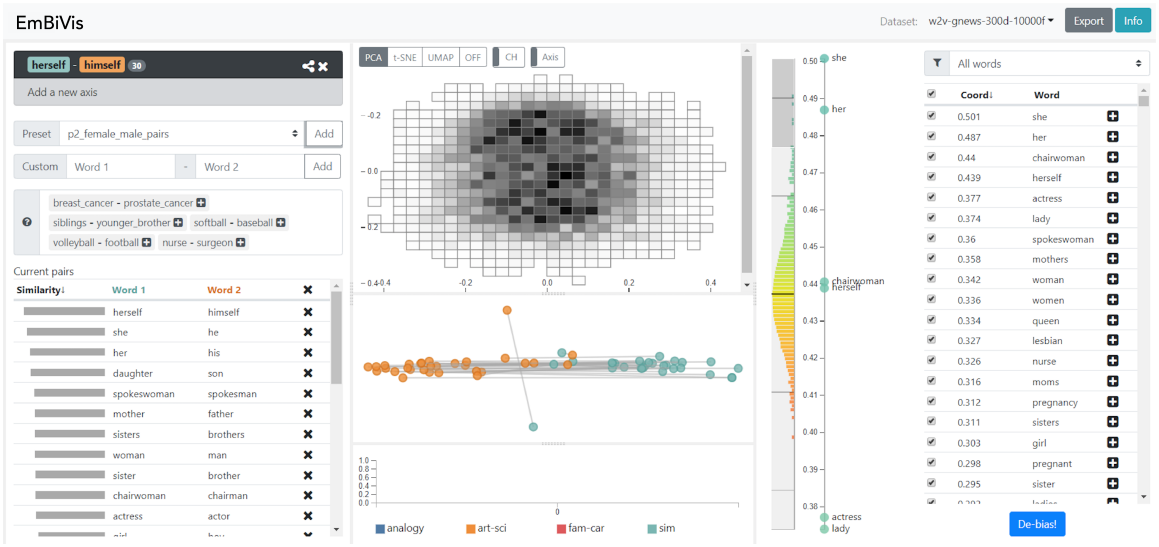


Figure 5.10: Initial view after selecting a preset of (*female-male*) pairs.

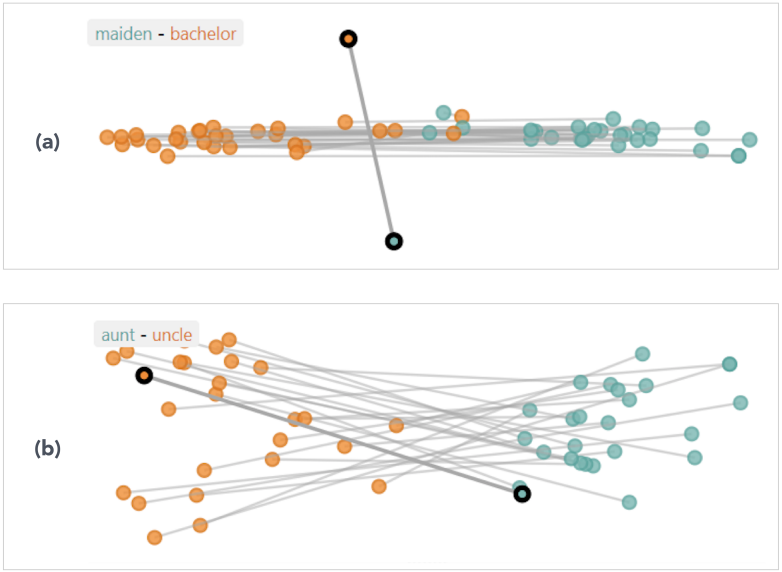


Figure 5.11: (a) A user hovers over and removes an unrelated pair (*maiden*, *bachelor*) from selected pairs set. (b) A user adds a recommended pair (*aunt*, *uncle*).

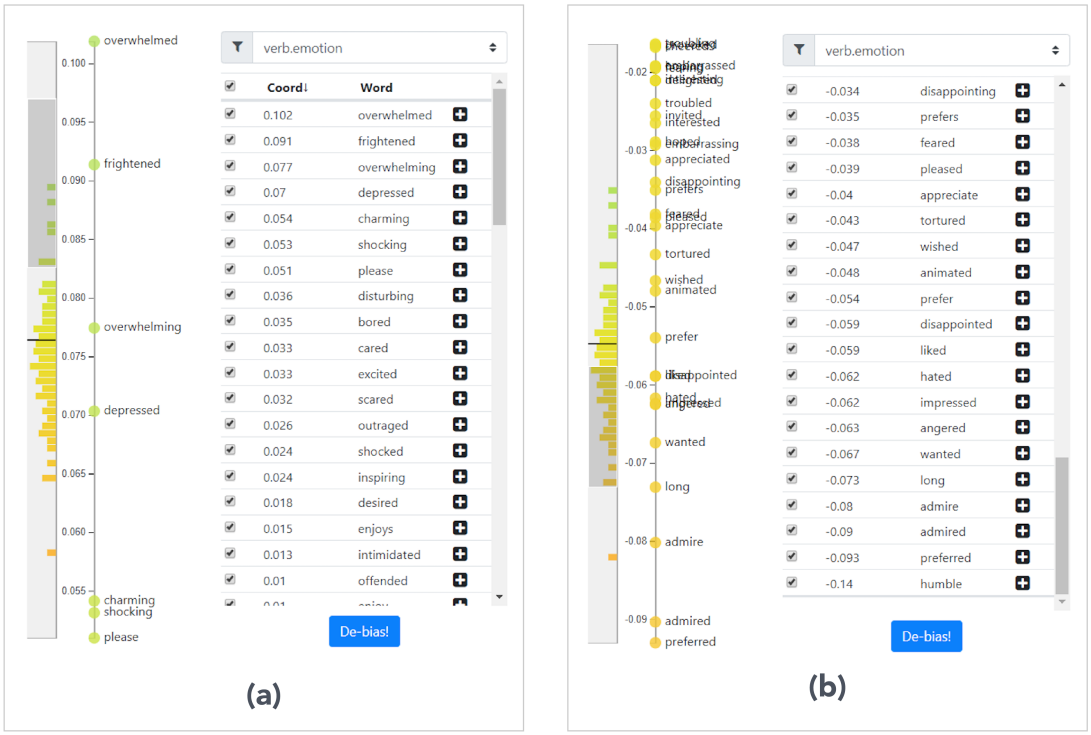


Figure 5.12: Emotional verbs located at the two ends, (a) *female*, and (b) *male*. Words corresponding to the *female* direction are mostly negative, and words corresponding to the *male* direction are mostly positive.

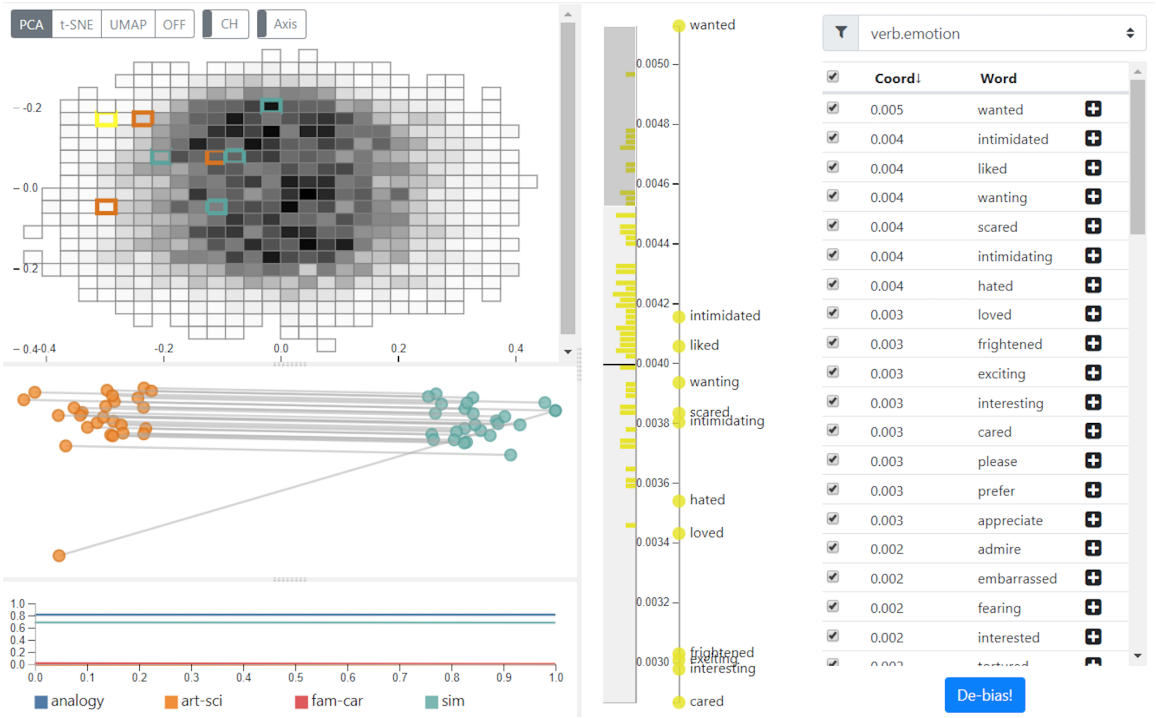


Figure 5.13: After debiasing, attribute pairs are more parallel and emotional verbs are well distributed along the attribute direction.

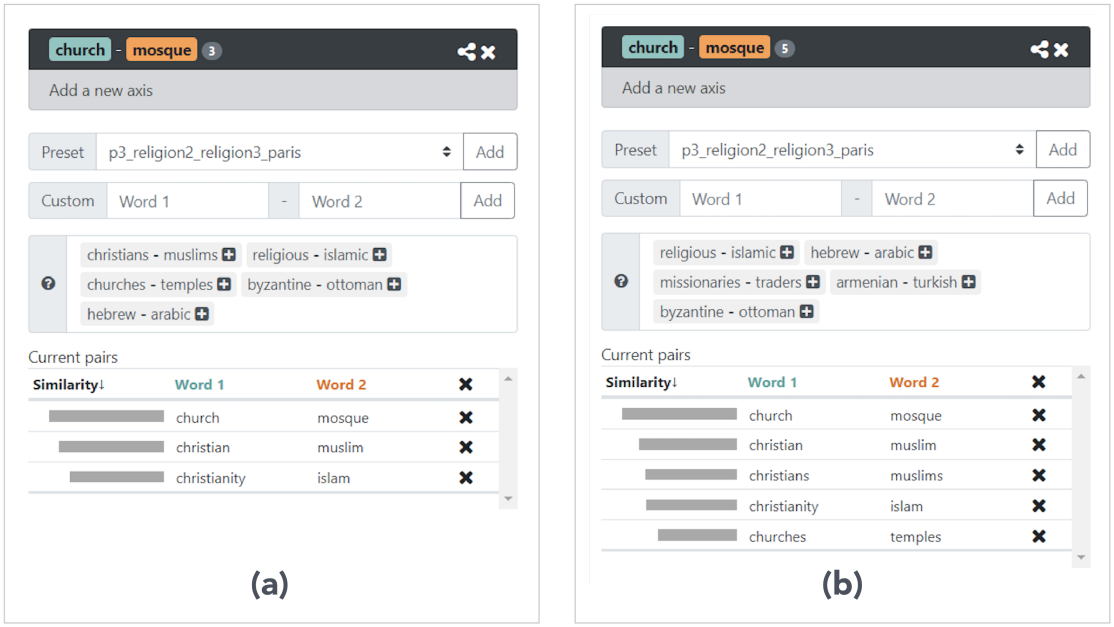


Figure 5.14: A user adds recommended pairs to the pairs set.

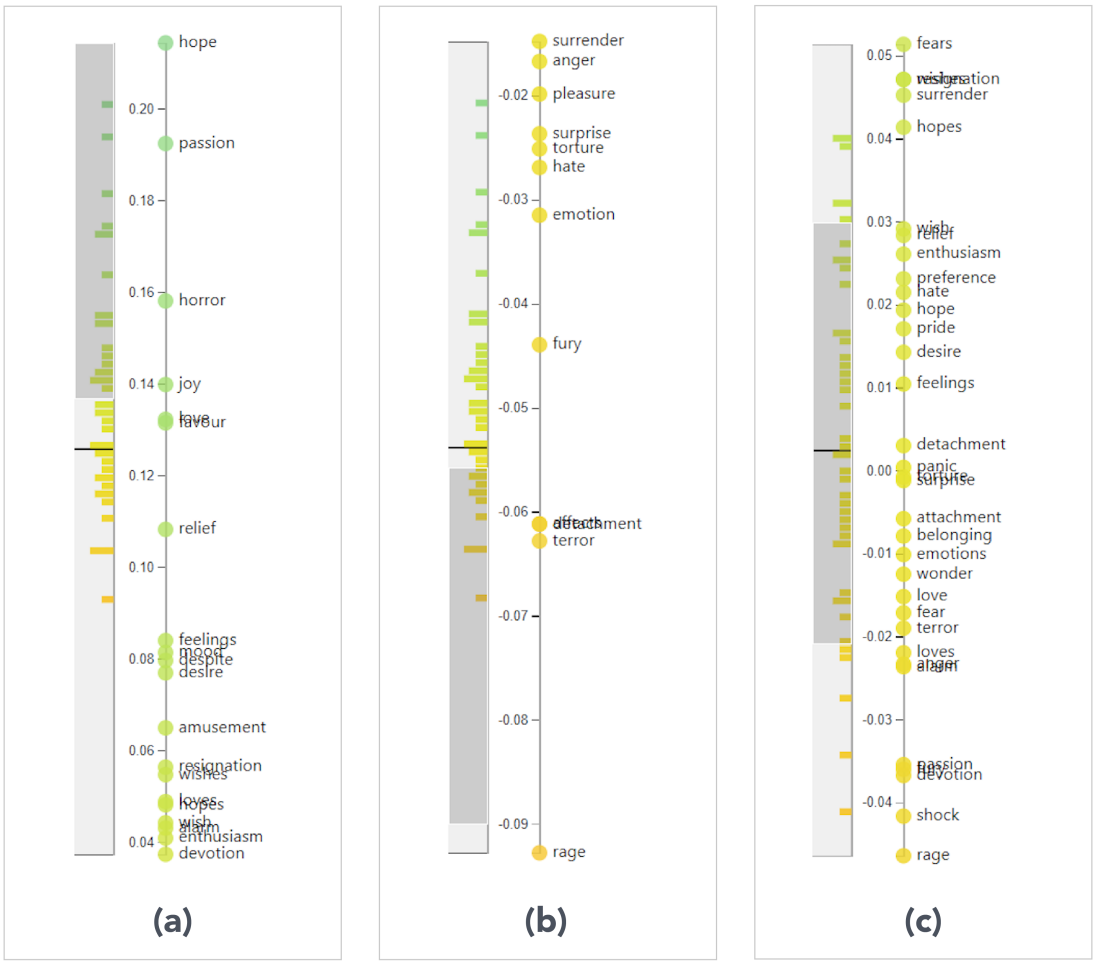


Figure 5.15: Nouns associated with the religion attribute direction before and after debiasing.

The user notices that most words at the extreme of the *female* direction are negative words such as *overwhelmed*, *frightened*, and *depressed* (Fig. 5.12a). On the other hand, the words at the end of the *he* extreme are mostly positive words such as *humble*, *preferred*, and *admired* (Fig. 5.12b). Since these associations are undesirable, the user selects them and debias the embedding by clicking the *Debias!* button. After debiasing, as shown in Fig. 5.13 , attribute pairs are more parallel then before (meaning most of their difference vectors can be explained by the bias vector), and the emotion verbs are much more well-distributed along the gender attribute direction. The user moves on to explore the implicit religious bias. The user selects *christian-muslim* pre-defined pairs (Fig. 5.14a) and add more pairs using the custom input menu. After populating more attribute-defining pairs using the recommendation made by our system (5.14b), the user sees the similarity distribution of words categorized as *noun.feeling* are highly biased (Fig. 5.15 a, b). For instance, words such as 'rage, terror' are on the islam extreme, while on the christianity extreme 'hope, passion'. Interestingly, 'horror' is also associated with christianity. After debiasing the embedding, words are equally distributed along the axis (Fig. 5.15 c).

5.6 Experiment

In this section, we present experimental results to test the effectiveness of our interactive approach against automated/static debiasing algorithms for the task of sentiment prediction.

5.6.1 Experiment Setup

To observe the effect of different debiasing methods for NLP downstream tasks, we select a sentiment analysis in the context of movie reviews. Following the experimental settings from [94], we measure the accuracy of sentiment prediction utilizing word embeddings that are debiased by different debiasing approaches. We compare our method with two static debiasing methods: HardDebias by [91] and ConceptorDebias by [124] since our method is based on these two static methods. Non-debiased, original word embedding is also used

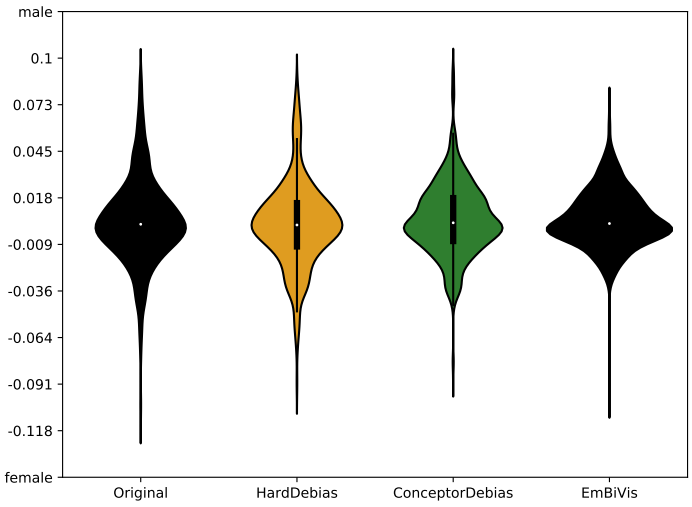


Figure 5.16: Classifier bias on the (female, male) attribute.

as a baseline. We use the IMDB movie review data set [125] with labels, where 1 and 0 each represents positive and negative sentiment, respectively. We add randomly chosen word from opposing set pairs (e.g., a typical female term or a male term) to the end of each test sentence, and calculate the difference in the sentiment prediction (i.e., the polarity score), made by a simple neural network that takes a pre-trained word embeddings as its embedding layer.

5.6.2 Results

Fig. 5.16 shows the *polarity score* for the (*female*, *male*) attribute. *polarity score* is defined as follows. For a sentence s_i, a label is tagged where l_i is either 1 (positive sentiment), or 0 (negative sentiment). After the model is trained on a train set, we use the model to output two prediction results for two modified sentences for each s_j. First prediction l_{j_1} is made for s_j followed by a male word, and the second prediction l_{j_2} is made for s_j followed by a female word. *polarity score* = $l_{j_1} - l_{j_2}$. The largest variance of the non-debiased embedding indicates that biased words affects the sentimental analysis to some extent. Among the compared methods, our approach shows the smallest variance, which

means that due to our debiasing approach, biased words had less effect on the sentiment classifier. In addition, for the non-debiased, original embedding, there is no clear trend that the embedding influences the average polarity score of our sentiment classifier.

5.7 Discussion

In this section, we discuss limitations and opportunities of our interactive debiasing approach.

Customizing Bias Assessment Metric. Bias evaluation metrics such as the Word Embedding Association Test (WEAT) [113] are highly dependent on the selected sets of attribute words ('art', 'science') that the target sets ('he', 'she') are compared with. For example, if we choose different sets of words describing 'art' and words describing 'science', we may get different conclusion whether a specific gender is more close to art or science. Furthermore, the meaning and usage of a word changes over time. And the dictionary of word embeddings may vary depending on the corpus that the given word embedding is trained on (e.g., word embedding trained on Twitter data vs. word embedding trained on news articles). Therefore, adopting the evaluation sets as-is may not be the best option to evaluate word embedding for different downstream tasks in different domains. We provide an option to easily change the list of evaluation words via a json file.

Dealing with Non-binary Attributes. Attributes can have multiple classes. Unfortunately, most existing literature treats multi-class attributes as binary. A common case is to pick a pair of classes (e.g., black vs. white from race). Another common case is use all pairwise classes (e.g., christianity vs. islam, islam vs atheism, atheism vs. christianity from religion). Similarly, EmBiVis only supports binary attributes. So users need to construct pairwise axis (e.g., male-female, black-white, etc) by either picking one pair or creating pairwise classes from a multi-class attribute. To deal with this limitation, we plan to extend our approach to incorporate multi-class attributes as future work. One challenge is visual-

izing attribute-defining word sets. In our current system, we visualize vector differences of attribute-defining pairs, which cannot be easily translated to triplets or quadruples. Another challenge is projecting words onto multiple axes. For example, the axis projection view, which shows two extremes, can be replaced with a radar chart or parallel coordinates to visualize multiple axes.

Simultaneous Debiasing of Multiple Attributes. Our approach adopts single attribute debiasing per iteration (Fig. 5.4). That is, users have to perform debiasing one attribute by one if they want to debias multiple attributes. However, debiasing by one attribute may aggravate another attribute bias in the updated embedding if there is an underlying association between the two attributes in the original corpus. In addition, debiasing multiple times for all attributes can lead to information loss because a debiasing operation distorts the original high-dimensional relationships among words. One solution can be joint/simultaneous debiasing [98, 94]. This will likely require a new visualization design that is tailored to support additional tasks such as attribute axes comparison, word projection on multiple axes, and so on.

Debiasing via Post-processing. As mentioned in Section 3.1, there are three types of debiasing techniques: pre-processing, in-processing, and post-processing. Our approach falls into the third category, i.e., a *post-hoc* debiasing of word embeddings. Although our current prototype supports two debiasing algorithm options, any other post-processing debiasing techniques can be applied to our human-in-the-loop framework. In addition, we envision that human input (even from layusers who are domain experts) can also be beneficial for pre-processing debiasing. Pre-processing debiasing techniques transform training data by augmenting data, neutralizing data, removing biased subsets, and so on. Users' domain knowledge can be incorporated into these algorithms in a semi-supervised manner. For example, which words to substitute, with which words to substitute, and what kind of documents to remove.

5.8 Conclusion

Debiasing word embedding is an important NLP task that can benefit many downstream applications. In this chapter, we propose a new human-in-the-loop debiasing framework designed for layusers without expert knowledge on NLP techniques. Our approach allows users to formulate attribute axes, assess attribute biases, and debias word embeddings via interactive visualization. We demonstrate the effectiveness of our approach using use cases and experiments. As future work, we would like to extend our framework to support multi-class debiasing.

CHAPTER 6
CONCLUSION

The goal of this dissertation was to achieve tight integration among algorithms, visualizations, and user interactions in human-in-the-loop machine learning systems for improved interactivity, scalability, and interpretability. We designed and developed three interactive visual text analytics systems for (1) interactive hierarchical topic modeling, (2) interactive search space reduction, and (3) interactive exploration and mitigation of biases in word embedding. From these works, we suggest the following design guidelines for tightly integrated interactive systems:

1. **Co-development of algorithms, visualizations, and user interactions that are grounded in human-in-the-loop interactivity.** State-of-the-art automated algorithms are not necessarily best for visual analytics. Interactions designed around algorithmic convenience can lead to analytic process not tailored to user needs. Thus, algorithms, visualizations, and interactions should be simultaneously developed during the deign process so that interaction can be directly linked to model change and fast algorithm update.
2. **Adaptive update for real-time interactions.** Many existing human-in-the-loop machine learning systems suffer from the interaction latency problem due to slow computational result updates, which hinders active user interactions. To solve this, we recommend tailoring the underlying algorithm to allow incremental, timely, and responsive updates instead of recomputing the entire solution after interaction [86].
3. **Bidirectional guidance between the system and the user.** Many prior works have focused on incorporating user feedback into the underlying model (human-to-machine guidance), but limited work has been done regarding the systems providing feedback to users to guide their analysis process (machine-to-human guidance). For efficient

completion of tasks, we advocate two-way guidance between the system and the user to leverage the strengths of both sides.

Building on the works in this dissertation, there are several directions for future work. First, we would like to explore interactions on various degrees of information. With our systems, users can interact with keywords, documents, and topics. Providing more flexible and diverse interaction units such as phrases, sentences, and paragraphs can be helpful in expressing users' mental models. For example, highlighting a phrase can be a form of user interaction. Another promising research direction is active learning for machine-to-human guidance. In Chapters 3-5, we offer visual guides such as interaction assistant in Archi-Text or keyword recommendation in TopicSifter and EmBiVis, which are mostly based on similarities. Applying active learning framework using various performance measures and explicit and implicit user feedback can lead to efficient analytic processes. Additionally, providing rationale behind guidance can build trust between the system and the users.

REFERENCES

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[2] A. Endert, P. Fiaux, and C. North, "Semantic interaction for visual text analytics," in *Proc. the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2012, pp. 473–482.

[3] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. the ACM SIGIR conference on Research and Development in Rnformation Retrieval*, ACM, 1999, pp. 50–57.

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

[5] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[6] J. Kim and H. Park, "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM Journal of Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, Nov. 2011.

[7] R. Du, D. Kuang, B. Drake, and H. Park, "DC-NMF: Nonnegative matrix factorization based on divide-and-conquer for fast clustering and topic modeling," *Journal of Global Optimization*, vol. 68, no. 4, pp. 777–798, Aug. 2017.

[8] J. Kim, Y. He, and H. Park, "Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework," *Journal of Global Optimization*, vol. 58, no. 2, pp. 285–319, 2014.

[9] N. E. Miller, P. C. Wong, M. Brewster, and H. Foote, "Topic islands-a wavelet-based text visualization system," in *Proc. Visualization*, Oct. 1998, pp. 189–196.

[10] E. Hetzler and A. Turner, "Analysis experiences using information visualization," *IEEE Computer Graphics and Applications*, vol. 24, no. 5, pp. 22–26, Sep. 2004.

[11] N. Cao, J. Sun, Y.-R. Lin, D. Gotz, S. Liu, and H. Qu, "FacetAtlas: Multifaceted visualization for rich text corpora," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1172–1181, Nov. 2010.

[12] J. Choo, C. Lee, C. K. Reddy, and H. Park, "UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 1992–2001, Dec. 2013.

[13] H. Lee, J. Kihm, J. Choo, J. Stasko, and H. Park, "iVisClustering: An interactive visual document clustering via topic modeling," *Computer Graphics Forum*, vol. 31, no. 3pt3, pp. 1155–1164, 2012.

[14] S. Liu, X. Wang, J. Chen, J. Zhu, and B. Guo, "Topicpanorama: A full picture of relevant topics," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2014, pp. 183–192.

[15] M. Kim, K. Kang, D. Park, J. Choo, and N. Elmqvist, "TopicLens: Efficient multi-level visual topic exploration of large-scale document collections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 151–160, Jan. 2017.

[16] Y.-R. Lin, J. Sun, N. Cao, and S. Liu, "Contextour: Contextual contour visual analysis on dynamic multi-relational clustering," in *Proc. the SIAM International Conference on Data Mining*, SIAM, 2010, pp. 418–429.

[17] N. Cao, D. Gotz, J. Sun, Y.-R. Lin, and H. Qu, "Solarmap: Multifaceted visual analytics for topic exploration," in *IEEE International Conference on Data Mining (ICDM)*, IEEE, 2011, pp. 101–110.

[18] D. Herr, Q. Han, S. Lohmann, and T. Ertl, "Hierarchy-based projection of high-dimensional labeled data to reduce visual clutter," *Computers & Graphics*, vol. 62, pp. 28–40, 2017.

[19] C. North, A. Endert, and P. Fiaux, "Semantic interaction for sensemaking: Inferring analytical reasoning for model steering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 2879–2888, 2012.

[20] W. Cui, S. Liu, Z. Wu, and H. Wei, "How hierarchical topics evolve in large text corpora," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2281–2290, Dec. 2014.

[21] F. V. Paulovich and R. Minghim, "HiPP: A novel hierarchical point placement strategy and its application to the exploration of document collections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1229–1236, Nov. 2008.

[22] 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 Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2271–2280, Dec. 2014.

[23] W. Dou, L. Yu, X. Wang, Z. Man, and W. Ribarsky, "HierarchicalTopics: Visually exploring large text collections using topic hierarchies," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2002–2011, Dec. 2013.

[24] E. Hoque and G. Carenini, "Interactive topic hierarchy revision for exploring a collection of online conversations," *Information Visualization*, 2018. eprint: https://doi.org/10.1177/1473871618757228.

[25] J. Stasko and E. Zhang, "Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *IEEE Symposium on Information Visualization*, IEEE, 2000, pp. 57–65.

[26] A. Smith, T. Hawes, and M. Myers, "Hiearchie: Visualization for hierarchical topic models," in *Proc. the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 2014, pp. 71–78.

[27] Y. Yang, Q. Yao, and H. Qu, "Vistopic: A visual analytics system for making sense of large document collections using hierarchical topic modeling," *Visual Informatics*, vol. 1, no. 1, pp. 40–47, 2017.

[28] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. the Neural Information Processing Systems*, 2000, pp. 556–562.

[29] Y. Hu, J. Boyd-Graber, B. Satinoff, and A. Smith, "Interactive topic modeling," *Machine Learning*, vol. 95, no. 3, pp. 423–469, Jun. 2014.

[30] T. Y. Lee, A. Smith, K. Seppi, N. Elmqvist, J. Boyd-Graber, and L. Findlater, "The human touch: How non-expert users perceive, interpret, and fix topic models," *International Journal of Human-Computer Studies*, vol. 105, pp. 28–42, 2017.

[31] W. A. Pike, J. Stasko, R. Chang, and T. A. O'connell, "The science of interaction," *Information Visualization*, vol. 8, no. 4, pp. 263–274, 2009.

[32] M. El-Assady, F. Sperrle, O. Deussen, D. A. Keim, and C. Collins, "Visual Analytics for Topic Model Optimization based on User-Steerable Speculative Execution," *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[33] A. Smith, V. Kumar, J. Boyd-Graber, K. Seppi, and L. Findlater, "Closing the loop: User-centered design and evaluation of a human-in-the-loop topic modeling system," in *Proc. the International Conference on Intelligent User Interfaces*, Tokyo, Japan: ACM, 2018, pp. 293–304, ISBN: 978-1-4503-4945-1.

[34] C. Musialek, "Distill-ery: Iterative topic modeling to improve the content analysis process,"

[35] J. Chuang, Y. Hu, A. Jin, J. D. Wilkerson, D. A. McFarland, C. D. Manning, and J. Heer, "Document exploration with topic modeling: Designing interactive visualizations to support effective analysis workflows," in *NIPS workshopon Topic Models: Computation, Application, and Evaluation*, 2013.

[36] M. El-Assady, R. Sevastjanova, F. Sperrle, D. Keim, and C. Collins, "Progressive learning of topic modeling parameters: A visual analytics framework," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[37] E. Hoque and G. Carenini, "ConVisIT: Interactive topic modeling for exploring asynchronous online conversations," in *Proc. the Conference on Intelligent User Interfaces*, Atlanta, Georgia, USA: ACM, 2015, pp. 169–180, ISBN: 978-1-4503-3306-1.

[38] N. Cao and W. Cui, *Introduction to text visualization*. Springer.

[39] S. Liu, J. Yin, X. Wang, W. Cui, K. Cao, and J. Pei, "Online visual analytics of text streams," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 11, pp. 2451–2466, Nov. 2016.

[40] X. Wang, S. Liu, J. Liu, J. Chen, J. Zhu, and B. Guo, "Topicpanorama: A full picture of relevant topics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 12, pp. 2508–2521, Dec. 2016.

[41] E. Horvitz, "Principles of mixed-initiative user interfaces," in *Proc. the SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, USA: ACM, 1999, pp. 159–166, ISBN: 0-201-48559-1.

[42] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang, "Dis-function: Learning distance functions interactively," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2012, pp. 83–92.

[43] A. Endert, C. Han, D. Maiti, L. House, and C. North, "Observation-level Interaction with Statistical Models for Visual Analytics," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2011, pp. 121–130.

[44] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert, "Podium: Ranking data using mixed-initiative visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 288–297, 2018.

[45] A. Endert, *Semantic Interaction for Visual Analytics: Inferring Analytical Reasoning for Model Steering*, 2. Morgan & Claypool Publishers, 2016, vol. 4, pp. 1–99.

[46] H. Kim, J. Choo, H. Park, and A. Endert, "Interaxis: Steering scatterplot axes via observation-level interaction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 131–140, 2016.

[47] B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert, "Axisketcher: Interactive nonlinear axis mapping of visualizations through user drawings," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 221–230, 2017.

[48] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit, "Opening the black box: Strategies for increased user involvement in existing algorithm implementations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1643–1652, Dec. 2014.

[49] Å. Björck, H. Park, and L. Eldén, "Accurate downdating of least squares solutions," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 2, pp. 549–568, 1994. eprint: https://doi.org/10.1137/S089547989222895X.

[50] H. Park and L. Eldén, "Downdating the rank-revealing urv decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 1, pp. 138–155, 1995. eprint: https://doi.org/10.1137/S0895479892241913.

[51] K. Yoo and H. Park, "Accurate downdating of a modified gram-schmidt qr decomposition," *BIT Numerical Mathematics*, vol. 36, no. 1, pp. 166–181, Mar. 1996.

[52] D. Kuang and H. Park, "Fast rank-2 nonnegative matrix factorization for hierarchical document clustering," in *Proc. the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2013, pp. 739–747, ISBN: 978-1-4503-2174-7.

[53] N. Gillis, D. Kuang, and H. Park, "Hierarchical clustering of hyperspectral images using rank-two nonnegative matrix factorization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 2066–2078, 2015.

[54] A. Smith, T. Y. Lee, F. Poursabzi-Sangdeh, J. Boyd-Graber, N. Elmqvist, and L. Findlater, "Evaluating visual representations for topic understanding and their effects on manually generated topic labels," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 1–16, 2017.

[55] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin, "Automatic evaluation of topic coherence," in *Proc. the Human Language Technologies: NAACL-HLT*, Los Angeles, California: Association for Computational Linguistics, 2010, pp. 100–108, ISBN: 1-932432-65-5.

[56] B. Drake, T. Huang, A. Beavers, R. Du, and H. Park, "Event detection based on nonnegative matrix factorization: Ceasefire violation, environmental, and malware events," in *Advances in Human Factors in Cybersecurity*, D. Nicholson, Ed., New York City, USA: Springer International Publishing, 2018, pp. 158–169, ISBN: 978-3-319-60585-2.

[57] S. Wang, Z. Chen, G. Fei, B. Liu, and S. Emery, "Targeted topic modeling for focused analysis," in *Proc. the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1235–1244.

[58] C. Li, Y. Wang, P. Resnick, and Q. Mei, "Req-rec: High recall retrieval with query pooling and interactive classification," in *Proc. the International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2014, pp. 163–172.

[59] M. A. Hearst, "TileBars: Visualization of term distribution information in full text information access," in *Proc. the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press/Addison-Wesley Publishing Co., 1995, pp. 59–66.

[60] H. Reiterer, G. Tullius, and T. M. Mann, "Insyder: A content-based visual-information-seeking system for the web," *International Journal on Digital Libraries*, vol. 5, no. 1, pp. 25–41, Mar. 2005.

[61] O. Hoeber and X. D. Yang, "A comparative user study of web search interfaces: Hotmap, concept highlighter, and google," in *Proc. the IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE, 2006, pp. 866–874.

[62] K. Andrews, W. Kienreich, V. Sabol, J. Becker, G. Droschl, F. Kappe, M. Granitzer, P. Auer, and K. Tochtermann, "The infoSky visual explorer: Exploiting hierarchical structure and document similarities," *Information Visualization*, vol. 1, no. 3-4, pp. 166–181, 2002. eprint: http://ivi.sagepub.com/content/1/3-4/166.full.pdf+html.

[63] E. Gomez-Nieto, F. San Roman, P. Pagliosa, W. Casaca, E. S. Helou, M. C. F. de Oliveira, and L. G. Nonato, "Similarity preserving snippet-based visualization of web search results," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 3, pp. 457–470, 2014.

[64] G. Smith, M. Czerwinski, B. Meyers, D. Robbins, G. Robertson, and D. S. Tan, "FacetMap: A scalable search and browse visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 797–804, 2006.

[65] E. Clarkson, K. Desai, and J. Foley, "Resultmaps: Visualization for search interfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1057–1064, 2009.

[66] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[67] I. Ruthven and M. Lalmas, "A survey on the use of relevance feedback for information access systems," *The Knowledge Engineering Review*, vol. 18, no. 2, pp. 95–145, 2003.

[68] D. Harman, "Towards interactive query expansion," in *Proc. the International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 1988, pp. 321–331.

[69] R. H. Fowler, W. A. Fowler, and B. A. Wilson, "Integrating query thesaurus, and documents through a common visual representation," in *Proc. the International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 1991, pp. 142–151.

[70] O. Hoeber, X.-D. Yang, and Y. Yao, "Visualization support for interactive query refinement," in *Proc. the IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE, 2005, pp. 657–665.

[71] S. Havre, E. Hetzler, K. Perrine, E. Jurrus, and N. Miller, "Interactive visualization of multiple query results," in *Proc. the IEEE Symposium on Information Visualization*, Washington, DC, USA: IEEE Computer Society, 2001, pp. 105–, ISBN: 0-7695-1342-5.

[72] J. Choo, H. Kim, E. Clarkson, Z. Liu, C. Lee, F. Li, H. Lee, R. Kannan, C. D. Stolper, J. Stasko, and H. Park, "VisIRR: A visual analytics system for information retrieval and recommendation for large-scale document data," *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 1, 8:1–8:20, Jan. 2018.

[73] T. Ruotsalo, J. Peltonen, M. Eugster, D. Głowacka, K. Konyushkova, K. Athukorala, I. Kosunen, A. Reijonen, P. Myllymäki, G. Jacucci, *et al.*, "Directing exploratory search with interactive intent modeling," in *Proc. the ACM International Conference on Information and Knowledge Management*, ACM, 2013, pp. 1759–1764.

[74] T. Ruotsalo, G. Jacucci, P. Myllymäki, and S. Kaski, "Interactive intent modeling: Information discovery beyond search.," *Communications of the ACM*, vol. 58, no. 1, pp. 86–92, 2015.

[75] S. Wang, M. Zhou, S. Mazumder, B. Liu, and Y. Chang, "Disentangling aspect and opinion words in target-based sentiment analysis using lifelong learning," *arXiv preprint arXiv:1802.05818*, 2018.

[76] V. Rakesh, W. Ding, A. Ahuja, N. Rao, Y. Sun, and C. K. Reddy, "A sparse topic model for extracting aspect-specific summaries from online reviews," in *Proc. the World Wide Web Conference*, International World Wide Web Conferences Steering Committee, 2018, pp. 1573–1582.

[77] X. Zheng, A. Sun, S. Wang, and J. Han, "Semi-supervised event-related tweet identification with dynamic keyword generation," in *Proc. of the ACM on Conference on Information and Knowledge Management*, ACM, 2017, pp. 1619–1628.

[78] R. White and R. Roth, *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool, 2013, ISBN: 9781598297843.

[79] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[80] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proc. the International Conference on Neural Information Processing Systems - Volume 2*, Montreal, Canada: MIT Press, 2014, pp. 2177–2185.

[81] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. the Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[82] D. Kuang, S. Yun, and H. Park, "SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering," *Journal of Global Optimization*, vol. 62, no. 3, pp. 545–574, Nov. 2014.

[83] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: A content-based approach to geo-locating twitter users," in *Proc. the ACM International Conference on Information and Knowledge Management*, ACM, 2010, pp. 759–768.

[84] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[85] W. Magdy and G. J. Jones, "PRES: A score metric for evaluating recall-oriented information retrieval applications," in *Proc. the International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2010, pp. 611–618.

[86] H. Kim, J. Choo, C. Lee, H. Lee, C. K. Reddy, and H. Park, "PIVE: Per-iteration visualization environment for real-time interactions with dimension reduction and clustering," in *Proc. the AAAI Conference on Artificial Intelligence*, 2017.

[87] J. Dressel and H. Farid, "The accuracy, fairness, and limits of predicting recidivism," *Science Advances*, vol. 4, no. 1, eaao5580, 2018.

[88] M. Kay, C. Matuszek, and S. A. Munson, "Unequal representation and gender stereotypes in image search results for occupations," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 3819–3828.

[89] M. O. Prates, P. H. Avelar, and L. C. Lamb, "Assessing gender bias in machine translation: A case study with google translate," *Neural Computing and Applications*, pp. 1–19, 2019.

[90] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang, "Gender bias in coreference resolution: Evaluation and debiasing methods," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 15–20.

[91] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," in *Advances in neural information processing systems*, 2016, pp. 4349–4357.

[92] T. Manzini, Y. C. Lim, Y. Tsvetkov, and A. W. Black, "Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings," *arXiv preprint arXiv:1904.04047*, 2019.

[93] H. Gonen and Y. Goldberg, "Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them," *arXiv preprint arXiv:1903.03862*, 2019.

[94] R. Popović, F. Lemmerich, and M. Strohmaier, "Joint multiclass debiasing of word embeddings," in *25th International Symposium on Intelligent Systems (ISMIS 2020)*, 2020. arXiv: 2003.11520 [cs.CL].

[95] K. Ethayarajh, D. Duvenaud, and G. Hirst, "Understanding undesirable word embedding associations," *arXiv preprint arXiv:1908.06361*, 2019.

[96] S. Dev and J. M. Phillips, "Attenuating bias in word vectors," *CoRR*, vol. abs/1901.07656, 2019. arXiv: 1901.07656.

[97] A. Lauscher, G. Glavaš, S. P. Ponzetto, and I. Vulić, "A general framework for implicit and explicit debiasing of distributional word vector spaces," *AAAI*, 2020. arXiv: 1909.06092 [cs.CL].

[98] S. Karve, L. Ungar, and J. Sedoc, "Conceptor debiasing of word representations evaluated on WEAT," in *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 40–48.

[99] J. Zhao, Y. Zhou, Z. Li, W. Wang, and K.-W. Chang, "Learning gender-neutral word embeddings," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 4847–4853.

[100] M.-E. Brunet, C. Alkalay-Houlihan, A. Anderson, and R. Zemel, "Understanding the origins of bias in word embeddings," in *International Conference on Machine Learning*, 2019, pp. 803–811.

[101] J. Zhao, T. Wang, M. Yatskar, R. Cotterell, V. Ordonez, and K.-W. Chang, "Gender bias in contextualized word embeddings," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 629–634.

[102] K. Kurita, N. Vyas, A. Pareek, A. W. Black, and Y. Tsvetkov, "Measuring bias in contextualized word representations," *arXiv preprint arXiv:1906.07337*, 2019.

[103] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[104] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[105] C. May, A. Wang, S. Bordia, S. R. Bowman, and R. Rudinger, "On measuring social biases in sentence encoders," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 622–628.

[106] S. Bordia and S. R. Bowman, "Identifying and reducing gender bias in word-level language models," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 7–15.

[107] J. E. Font and M. R. Costa-jussà, "Equalizing gender biases in neural machine translation with word embeddings techniques," Jan. 10, 2019. arXiv: 1901.03116v2 [cs.CL].

[108] F. Prost, N. Thain, and T. Bolukbasi, "Debiasing embeddings for reduced gender bias in text classification," in *Proceedings of the First Workshop on Gender Bias*

in Natural Language Processing, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 69–75.

[109] Y. Qian, U. Muaz, B. Zhang, and J. W. Hyun, "Reducing gender bias in word-level language models with a gender-equalizing loss function," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, 2019, pp. 223–228.

[110] T. Sun, A. Gaut, S. Tang, Y. Huang, M. ElSherief, J. Zhao, D. Mirza, E. Belding, K.-W. Chang, and W. Y. Wang, "Mitigating gender bias in natural language processing: Literature review," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1630–1640.

[111] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *arXiv preprint arXiv:1908.09635*, 2019.

[112] P. Sen and D. Ganguly, "Towards socially responsible ai: Cognitive bias-aware multi-objective learning," *AAAI*, 2020.

[113] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases," *Science*, vol. 356, no. 6334, pp. 183–186, 2017.

[114] A. G. Greenwald, D. E. McGhee, and J. L. Schwartz, "Measuring individual differences in implicit cognition: The implicit association test.," *Journal of personality and social psychology*, vol. 74, no. 6, p. 1464, 1998.

[115] N. Garg, L. Schiebinger, D. Jurafsky, and J. Zou, "Word embeddings quantify 100 years of gender and ethnic stereotypes," *Proceedings of the National Academy of Sciences*, vol. 115, no. 16, E3635–E3644, 2018.

[116] Y. Liu, E. Jun, Q. Li, and J. Heer, "Latent space cartography: Visual analysis of vector space embeddings," *Computer Graphics Forum (Proc. EuroVis)*, vol. 38, no. 3, pp. 67–78, 2019.

[117] F. Heimerl and M. Gleicher, "Interactive analysis of word vector embeddings," in *Computer Graphics Forum*, Wiley Online Library, vol. 37, 2018, pp. 253–265.

[118] J. Z. Piero Molino Yang Wang, "Parallax: Visualizing and understanding the semantics of embedding spaces via algebraic formulae," in *ACL*, 2019.

[119] S. Liu, P.-T. Bremer, J. J. Thiagarajan, V. Srikumar, B. Wang, Y. Livnat, and V. Pascucci, "Visual exploration of semantic relationships in neural word embeddings,"

IEEE transactions on visualization and computer graphics, vol. 24, no. 1, pp. 553–562, 2017.

[120] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, "Embedding projector: Interactive visualization and interpretation of embeddings," in *NIPS Workshop on Interpretable ML in Complex Systems*, 2016. arXiv: 1611.05469 [stat.ML].

[121] I. T. Jolliffe, "Principal components in regression analysis," in *Principal component analysis*, Springer, 1986, pp. 129–155.

[122] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9: 2579–2605, Nov 2008.

[123] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *ArXiv e-prints*, Feb. 2018. arXiv: 1802.03426 [stat.ML].

[124] S. Karve, L. Ungar, and J. Sedoc, "Conceptor debiasing of word representations evaluated on weat," *arXiv preprint arXiv:1906.05993*, 2019.

[125] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, Association for Computational Linguistics, 2011, pp. 142–150.