



DI (FH) Lukas Mayr

PaperViz: A visual document collection exploration tool to support scientific paper writing

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.tech. Vedran Sabol

Knowledge Technologies Institute

Co-Supervisor: MSc Cecilia Di Sciascio

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

.....

date

.....

(signature)

Abstract

Knowledge work such as writing scientific papers typically requires the extraction of useful information from scientific literature. Nowadays the primary information artifact used by researchers are electronic documents available on the web, which can be found by general and academic search engines such as Google Scholar or IEEE Xplore. But retrieving only the most relevant results may prove to be hard due to the vast amount of web sites and papers. As a consequence, researchers are often confronted with loads of low-quality or irrelevant content. To address this issue, we introduce a novel system, which combines a rich, interactive web-based user interface and different visualization approaches. This system, named PaperViz, enables researchers to use their visual skills to quickly spot potentially relevant literature within large document collections. The core component of PaperViz is a graph-based visualization, which is using key phrase extraction algorithms to map textual to graphic representations. This visualization approach relies on an interactive user-driven model that allows researchers to shape the visualization by formulating a search query consisting of a set of key phrases and their positions within a spatial layout. Based on this search query, the graph-based visualization computes and presents information about document collections using different visual features and mini visualizations enabling researchers to gain rapid knowledge about the content and the relevance of the inspected document repositories. In addition, it helps users to identify key phrases that best describe the kind of papers they are looking for. These key phrases can be used to optimize the search query, which usually results in a smaller but more relevant result list. Furthermore, PaperViz provides additional features to support researchers during the whole process of writing scientific papers, such as citation management, document metadata extraction or a web based text editor. This paper introduces the design and components of this system and describes several use scenarios, in which PaperViz supports rapid sensemaking on document collections. Moreover, a case study, where experts and non-experts had to accomplish different tasks using PaperViz, was conducted in order to examine its usability.

Keywords. search engines, key phrases, document collection exploration, graph-based visualization approaches

Kurzfassung

Wissensarbeit, wie beispielsweise das Verfassen von wissenschaftlichen Fachartikeln, erfordert zumeist eine umfangreiche Recherche bestehender Literatur um das Themenfeld zu erschließen. Heutzutage nutzen Wissenschaftler primär elektronische Dokumente als Informationsquelle, die im Web zum Download bereitgestellt werden und mittels akademischer Suchmaschinen wie Google Scholar oder IEEE Xplore gefunden werden können. Es hat sich allerdings herausgestellt, dass es aufgrund der riesigen Mengen an Webseiten und elektronischen Artikeln oft schwierig ist eine Suchanfrage so zu formulieren, dass die Suchmaschinen nur potentiell relevante Resultate liefern. Meist beinhalten die Ergebnislisten eine Vielzahl von irrelevanten oder minderwertigen Artikeln. Das Identifizieren von nützlichen Informationen in solch großen Dokumentensammlungen ist üblicherweise mit viel Aufwand und Zeit verbunden. Aus diesem Grund haben wir ein innovatives, web-basiertes System namens PaperViz entwickelt, welches es Wissenschaftlern ermöglicht, ihre visuellen Fähigkeiten gezielt dazu einsetzen, um rasch potentiell relevante Informationen in großen Dokumentensammlungen zu identifizieren. Die Hauptkomponente von PaperViz ist eine graphen-basierte Visualisierung welche Textextrahierungsalgorithmen verwendet um textuelle Information grafisch abzubilden. Dieser Visualisierungsansatz basiert auf einem benutzergesteuerten Modell, welches Wissenschaftlern erlaubt die Visualisierung aufgrund ihrer Interessen zu modellieren. Basierend auf der Definition einer Suchanfrage, welche aus Schlüsselwörtern und deren Positionen innerhalb eines Koordinatensystems besteht, berechnet das System verschiedene Eigenschaften der untersuchten Dokumentensammlungen und stellt diese Anhand verschiedener visueller Merkmale (Farben, Positionen, Größen, etc.) und Minivisualisierungen dar. Dadurch erlangen Wissenschaftler schnell Kenntnis über den Inhalt und der Relevanz von Dokumenten und Dokumentensammlungen. Des Weiteren hilft dieses System den Benutzern bei der Definition von Schlüsselwörtern, welche zur Optimierung einer Suchanfrage genützt werden können. Das hat wiederum den Vorteil, dass die Ergebnislisten kürzer aber auch qualitativ hochwertiger werden. Zusätzlich bietet PaperViz Funktionen an, welche den Wissenschaftler während des gesamten Prozesses zur Erstellung des Fachartikels unterstützen, wie beispielsweise eine Zitierfunktion, automatisches Extrahieren von Dokument-Metadaten oder einem web-

basierten Texteditor. Die vorliegende Arbeit stellt das Design und die Komponenten dieses Systems vor und beschreibt verschiedene Anwendungsszenarien von PaperViz. Zusätzlich wurde eine Evaluierung durchgeführt mit dem Ziel, Stärken und Schwächen des Systems zu identifizieren sowie dessen Benutzerfreundlichkeit zu messen.

Schlüsselwörter. Suchmaschinen, Schlüsselwörter, Dokumentensammlung, Graphenbasierte Visualisierungsansätze

Acknowledgments

First, I would like to thank my thesis advisors Cecilia di Sciascio and Vedran Sabol for their guidance in creating this thesis.

I also want to thank my closest friends and colleagues for providing me with unfailing support throughout my years of study and through the process of researching and writing this thesis.

Finally, I must express my very profound gratitude to my family, especially my parents, for having enabled my studies, their confidence in me, and their interest in my work.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Structure	4
2	Related Work	5
2.1	Approaches of document collection visualization	6
2.1.0.1	ThemeRiver and TIARA	6
2.1.0.2	ParallelTopics	7
2.1.0.3	Tag cloud	8
2.1.0.4	TopicNets	9
2.1.0.5	FaceAtlas	10
2.1.0.6	InfoSky	11
2.1.0.7	ThemeScape	12
2.1.0.8	VIBE	13
2.1.0.9	Jigsaw	14
2.1.0.10	Aduna Cluster Maps	15
2.1.0.11	TopicViz	17
2.1.0.12	Apolo	18
2.1.0.13	TileBars	19
2.1.0.14	WordTree	20
2.1.0.15	PhraseNet	21
2.1.0.16	Document Cards	22
2.2	Document Collection Management Systems	23
2.3	Contributions	24
3	Design	25
3.1	Requirements	25
3.2	GUI components	27
3.2.1	User management	31
3.2.2	Navigation	32
3.2.2.1	Tree view	33

3.2.2.2	Breadcrumbs	33
3.2.2.3	Actions	33
3.2.3	Metadata information area	34
3.2.4	Bookmarks and annotations	35
3.2.4.1	List of bookmarks	35
3.2.4.2	Text editor	36
3.2.5	Visualization	36
3.2.5.1	Tag cloud	37
3.2.5.2	Key phrase box	38
3.3	Graph-based visualization	39
3.3.1	Layout	39
3.3.2	Visual encoding	41
3.3.2.1	Shape	42
3.3.2.2	Color	42
3.3.2.3	Intensity	42
3.3.2.4	Position	43
3.3.2.5	Size	48
3.3.3	Interactions and Details-on-demand	48
3.3.3.1	Bar chart	49
3.3.3.2	Connecting lines	51
3.3.3.3	TileBars	52
3.3.4	Additional controls	53
3.3.4.1	Gravity slider	53
3.3.4.2	Buttons	54
3.3.5	Usage scenarios	54
3.3.5.1	Scenario 1	54
3.3.5.2	Scenario 2	59
3.3.5.3	Scenario 3	60
3.3.5.4	Conclusion	62
3.4	Workflow	63
3.5	Summary	64
4	Implementation	66
4.1	Information flow	67
4.2	Backend processing service	68
4.2.1	Frameworks and libraries	69
4.2.1.1	Web API 2	69
4.2.1.2	OAuth 2.0	70
4.2.1.3	Entity Framework	70
4.2.1.4	RestSharp	71
4.2.1.5	iTextSharp	71

4.2.2	Web APIs	71
4.2.2.1	Connection to Mendeley	71
4.2.2.2	Connection to Sensium	74
4.2.2.3	Connection to the PaperViz front end	76
4.2.3	Data processing and preperation	82
4.2.4	Data storing	83
4.3	Web front end	86
4.3.1	Frameworks, libraries and design patterns	87
4.3.1.1	jQuery	87
4.3.1.2	Bootstrap	87
4.3.1.3	Knockout	88
4.3.1.4	JavaScript Module Pattern	89
4.3.1.5	D3.js	90
4.3.2	Summary	90
5	Evaluation	91
5.1	Methodology	91
5.1.1	Evaluation preparation	92
5.1.2	Tasks	93
5.1.2.1	Training Task	93
5.1.2.2	Task 1	93
5.1.2.3	Task 2	94
5.1.2.4	Task 3	94
5.1.2.5	Task 4	94
5.1.3	Questionnaires	94
5.1.3.1	Questionnaire 1	95
5.1.3.2	Questionnaire 2	95
5.1.3.3	Questionnaire 3	95
5.1.4	Data sets	95
5.2	Participants	96
5.3	Procedure and Conditions	96
5.3.1	Introduction	96
5.3.2	Training	96
5.3.3	Feedback	97
5.4	Results	97
5.5	Conclusion	102
6	Conclusion & Future Work	103
A	Questionnaires	106
B	Acronyms	108

Bibliography

112

List of Figures

1.1	A screenshot of PaperViz	1
2.1	ThemeRiver Visualization	6
2.2	ParallelTopics Visualization	7
2.3	Tag cloud Visualization	8
2.4	TopicNets Visualization	9
2.5	FaceAtlas Visualization	10
2.6	InfoSky Visualization	11
2.7	ThemeScape Visualization	12
2.8	VIBE Visualization	13
2.9	JigSaw List View	14
2.10	JigSaw Graph View	15
2.11	Social Bookmarking Visualization	15
2.12	TopicViz Visualization	17
2.13	Apolo Visualization	18
2.14	TileBar Visualization	19
2.15	WordTree Visualization	20
2.16	PhraseNet Visualization	21
2.17	Document Cards Visualization	22
3.1	First mockup of the GUI	27
3.2	Details-on-demand mockup	28
3.3	Final layout of PaperViz	30
3.4	Login dialog of PaperViz	31
3.5	The user ribbon of PaperViz	31
3.6	The sync ribbon of PaperViz	32
3.7	The tree view component of PaperViz	33
3.8	The breadcrumbs component of PaperViz	33
3.9	The metadata information area of PaperViz	34
3.10	The list of bookmarks	35
3.11	The text editor of PaperViz	36

3.12	The tag cloud of PaperViz	37
3.13	The key phrase box of PaperViz	38
3.14	Dragging and dropping key phrases	38
3.15	The graph-based visualization of PaperViz	39
3.16	Shapes: triangles and circles	41
3.17	Relevance is visualized by intensity	42
3.18	Key phrases —intensity changed	43
3.19	Positioning interpretation	44
3.20	Example: badly placed key phrases	46
3.21	Example: overlapping visualization elements	46
3.22	Bar chart for collections	49
3.23	Bar chart for key phrases	50
3.24	Connecting Lines	51
3.25	Connecting lines —overlaps are avoided	51
3.26	TileBars	52
3.27	PDF page parameter	53
3.28	Gravity slider	53
3.29	Scenario 1: PaperViz visualization	55
3.30	Scenario 1: PaperViz visualization (2)	56
3.31	Scenario 1: PaperViz visualization (3)	57
3.32	Scenario 1: TileBar visualization	57
3.33	Senario 1: identified document section	58
3.34	Senario 1: metadata information area	58
3.35	Senario 2: PaperViz visualization	59
3.36	Senario 3: PaperViz visualization	61
3.37	Senario 3: PaperViz visualization (2)	62
3.38	Workflow	63
4.1	The information flow between PaperViz and external systems	67
4.2	OAuth 2.0 login credential flow	70
4.3	The settings dialog of PaperViz	73
4.4	Sensium Workflow	74
4.5	Synchronization options of PaperViz	79
4.6	Loading animation of PaperViz	79
4.7	Flow diagram of the loading process	80
4.8	Class diagram of the document collection response object	81
4.9	Loading and saving the application state	82
4.10	Database diagram	84
4.11	Front end architecture	86
4.12	Bootstrap template ‘SB Admin 2’	88

5.1	System Usability Scale Grading Graph	97
5.2	Scores of the questions about the functions and features of PaperViz	98
5.3	The results of the personal feelings about the success in accomplishing a task (performance), the effort to accomplish their level of performance (effort) and the difficulty-level of the task (frustration).	100

List of Tables

2.1	Comparison of document collection management systems used by researches	23
3.1	Confrontation of requirements and features	65
4.1	Consumed Mendeley web API functions	74
4.2	Exposed web API functions of the PaperViz' backend processing service . .	77
4.3	Description of the database tables	85
5.1	PaperViz' System Usability Scale scores	97
5.2	Log statistics of user activities traced in the course of the evaluation	101
A.1	Questionnaire about workload and task difficulty for PaperViz	106
A.2	Questionnaire about the functions and components of PaperViz	107
A.3	The System Usability Scale questions adapted from [40]	107

Chapter 1

Introduction

Contents

1.1 Motivation	2
1.2 Structure	4

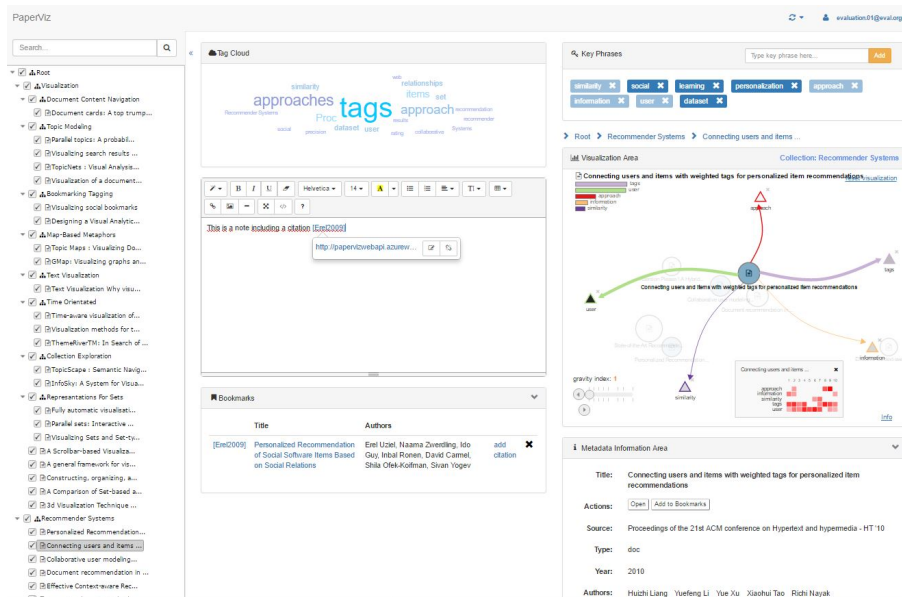


Figure 1.1: A screenshot of PaperViz

This thesis introduces a novel interactive tool called PaperViz, which is designed to support researchers in writing scientific papers. PaperViz combines different visualization techniques, which are based on key phrase extraction algorithms and are interactively linked to each other. By using their visual skills, researchers are able to quickly examine the content of large document collections. A short glance at the visualizations is usually

sufficient to determine whether a document is likely to be relevant or not. Thus, the system enables the user to quickly spot sections of documents that might be worth inspecting in depth without reading a single paragraph. As a result, this reduces the amount of reading and results in a faster process of finding relevant information. Furthermore, additional issues are addressed that typically arise when writing scientific papers, such as managing relevant references or finding additional key words for refining the search process. PaperViz consists of two main components:

- **The backend processing service** is used for data storage, data preparation and the user management. In addition, it implements the interfaces to two external systems. The first system is a text mining engine designed to extract key phrases from text blocks. The second one is a reference management system, which is used as the document collection source for PaperViz.
- **The Web front end** is a single page application (SPA) that complies with MCV principles and presents the data prepared by the backend processing service in understandable patterns. Moreover, it includes some features, which aim to speed up the process of exploring, understanding and managing large collections of documents (see figure 1.1).

1.1 Motivation

Nowadays the web contains a tremendous amount of digital data and its growth rate increases rapidly. These characteristics hold great potential for different areas of life. Researchers for instance profit from the vast scientific materials they have access to. Due to the billions of digital scientific articles, papers, journals and other documents, it is usually not hard to find literature related to a specific research field. Furthermore, numerous search engines, such as Google Scholar, IEEE Xplore or Pubmed exist, which are especially designed for the purpose of academic research. But despite these benefits, identifying valuable scientific literature is still a very challenging task. To point out some common issues researchers face nowadays, it is first of all important to take a look at the process of finding and evaluating research materials. After defining a problem and selecting a research field, the following steps have to be carried out (adapted from [3, 41]):

1. Define search expressions
2. Browse the web by using general or academic search engines
3. Review and evaluate the literature found by the search engines

4. Derive new search expressions (key phrases, authors, journals, etc.) from these documents
5. Start over at step 2 –try different combinations of search expressions

If researchers can instantly formulate a well-defined search query and only need a small set of documents, not many iterations will be needed to perform this process. But in practice this is rarely the case [36]. When starting knowledge work on an unfamiliar field, researchers may not have a strategy of what to search for and what facets should be covered in their papers. As a consequence they use search expressions that are too general which results in a huge list of low-quality and irrelevant content. Spotting valuable material within this list is usually a very time consuming task [3]. When using academic search engines the result of the search task is typically provided as a list of abstracts, which researchers are required to read in order to determine the relevance of a document related to their science project. Furthermore, they need to identify suitable key phrases that best describe the kind of papers they are interested in. This step is crucial in order to optimize the search results in further iterations. Thus, finding relevant information is not only a simple lookup task. Researchers should also be able to learn from search results and develop a search strategy which is different from trial-and-error tactics in order to find the information they are looking for. In literature searching to learn or searching to investigate is often referred as exploratory search. Exploratory search involves much more tasks than simply formulating a search query. It typically requires the information seeker to spend time viewing, comparing and making qualitative judgments about sets of objects [28]. In addition, the management of found resources for later detailed exploration is also a task researchers have to cope with. In short, researchers nowadays face the following challenges:

- How to quickly identify relevant resources within a large collection of documents?
- How to find suitable keywords and search expressions related to the field of interest?
- How to find further interesting resources?
- How to organize the list of documents for a later exploration?

Currently no system addresses all these issues, and therefore PaperViz was designed. Thus, the goal of the system can be defined as speeding up the process of finding, evaluating and organizing scientific resources.

1.2 Structure

Since this master thesis introduces a system, which core component is the visualization of document collections, chapter 2 describes some popular existing document collection visualization systems and approaches. Moreover, this chapter compares some widely used document collection management systems in order to evaluate, which of them is the most suitable candidate to act as the document collection source system for PaperViz.

Chapter 3 introduces the design of PaperViz. At the beginning the most important requirements, which should be fulfilled by the system are listed, followed by a detailed explanation of the graphical layout and the supported functions. In addition, a detailed description of PaperViz' graph-based visualization is given. Chapter 3 also includes a section outlining three typical usage scenarios of how this visualization can be used to examine document collections. The chapter closes with a confrontation of the requirements and the implemented features.

Chapter 4 focuses on the architecture and the implementation of PaperViz. It describes the purposes and the used technologies of both PaperViz components, the backend processing service and the web front end. Furthermore, it outlines the interfaces between these components and the external systems, Mendeley and Sensium.

Chapter 5 contains the evaluation of PaperViz, and finally chapter 6 concludes the thesis and looks out on future work.

Chapter 2

Related Work

Contents

2.1	Approaches of document collection visualization	6
2.2	Document Collection Management Systems	23
2.3	Contributions	24

Since this thesis is about visualizing the content of document collections, it is first of all necessary to take a closer look at existing document visualization approaches and systems. Today numerous of these approaches and systems exists, but they vary regarding type and granularity of information provided about collections. For instance, some techniques focus on visualizing associations such as citations and other document references between documents. Other approaches were designed to reveal patterns of themes and topics within a document. Due to the wide range of possible applications of document visualization techniques, it is quite difficult to categorize them. One popular way of classifying visualization approaches is by dividing them into three groups according to the granularity of information they provide about the document repositories [12]:

1. **Collection level:** to provide an overview about the content of collections and sub collections. The goal of collection level visualizations is to provide a general overview of the collection content and characteristics like which topics, themes or keywords does it cover.
2. **Document level:** to visualize similarities and differences between documents. Document level visualizations are used to visualize attributes of documents, such as their relations and linkage to other documents, topics or themes. These are mostly interactive techniques providing a visual search interface that enables users to query (query-focused) or explore (browsing-focused) their collections.

3. **Intra-document level:** to illustrate the internal structure of a document. Intra-document level approaches visualize the internal structure of a document like the frequency and distribution of key words.

Many of the visualization described in the following section cannot be strictly assigned to one of this three categories as they provide visualizations for different granularity levels (ParallelTopics, FaceAtlas) or can be applied on both, documents and document collections (WordTree, PhraseNet).

Moreover, this chapter also includes a section about some widely used document collection management systems. As most of these system are very mature and enjoy great popularity, it is not necessary to integrate a document collection management functionality into PaperViz. Instead, one of these systems should act as a source for PaperViz.

2.1 Approaches of document collection visualization

In the following section, some widley used document collection visualization approaches and systems are described.

2.1.0.1 ThemeRiver and TIARA

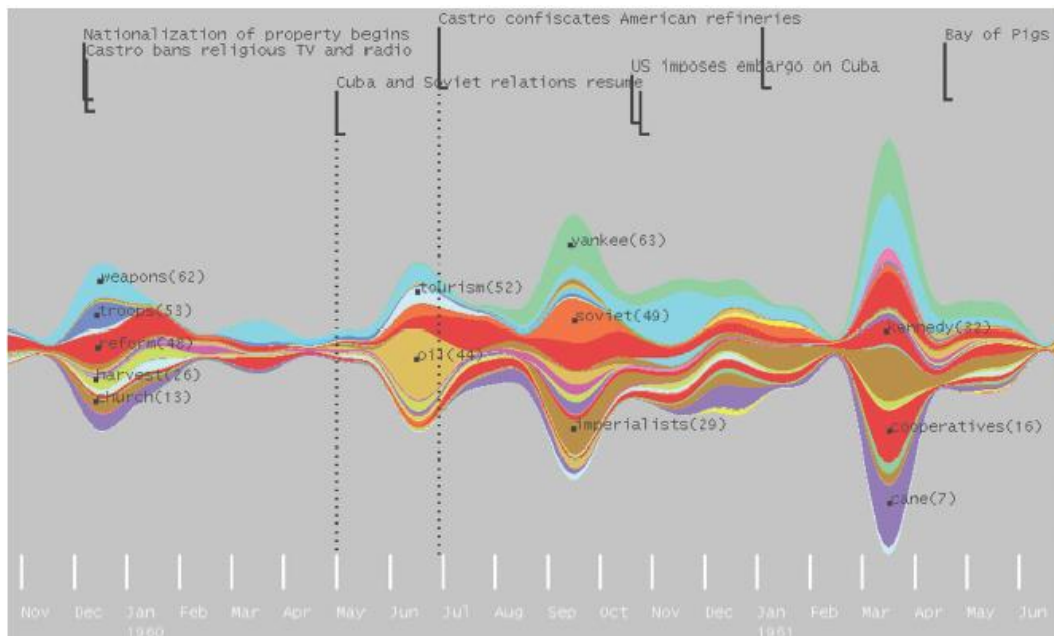


Figure 2.1: ThemeRiver Visualization [19]

ThemeRiver is an approach that visualizes thematic changes over a serial dimension within a collection of documents. It aims to identify novel and unexpected patterns of themes and topics. The serial dimension is often expressed as a time scale so that the “river” flows through time. Topics or themes are visualized as colored currents and the widths of these currents indicate the strength of a specific topic at a given time. Furthermore, the visualization can be enriched with external events. Figure 2.1 illustrates such a ThemeRiver visualization for news articles related to Fidel Castro from November 1959 to June 1961. It shows that Castro started to talk about the Soviet Union in May 1960, where Cuba and the Soviet Union resumed diplomatic relations [19].

A very similar but more sophisticated time-aware visualization technique is TIARA (Text Insight via Automated Responsive Analytics), which uses Latent Dirichlet Allocation (LDA) to automatically extract time-sensitive keywords on collections of articles and emails [1]. LDA is a generative probabilistic model for extracting latent themes within a collection of documents. It is a quite simple and efficient topic model and therefore perhaps the most common one currently in use [4]. Using this time-aware visualization approaches an analyst can easily identify the major topics in the document collection and how they evolved over time. But if they want to identify relationships between topics and documents, other visualization techniques need to be used such as ParallelTopics.

2.1.0.2 ParallelTopics

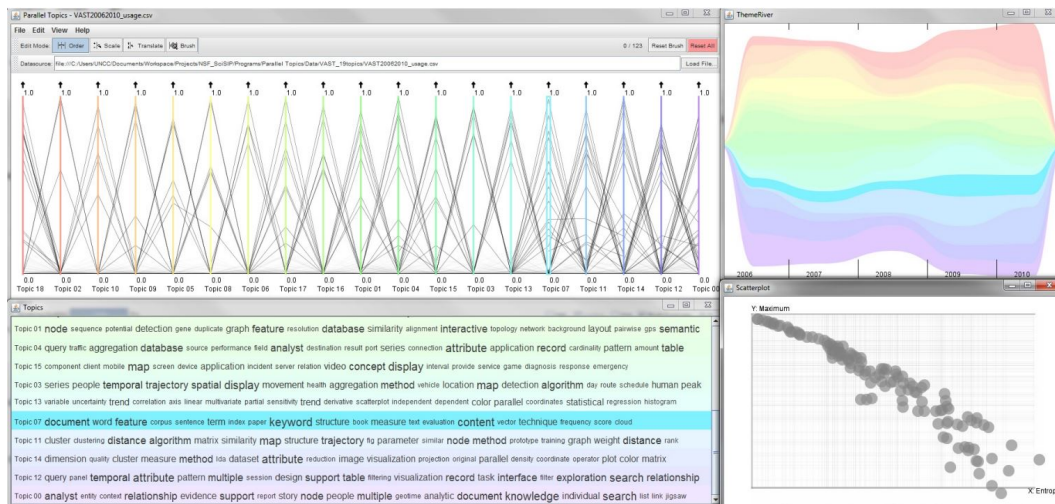


Figure 2.2: ParallelTopics Visualization [11]

ParallelTopics is a visualization technique that uses the parallel coordinate metaphor to present a document distribution across several topics. This enables an analyst to easily identify the strength of the relationship between topics and documents. Furthermore, it

similar topic. Nowadays tag clouds are widely used on blogs and websites such as Flickr, Technorati or Del.icio.us. In the World Wide Web they are also used for navigation purposes and thus, also, play a role for search engine optimization. There are also a lot of very similar techniques to tag clouds like tag graphs, spark clouds or data clouds [2, 23].

2.1.0.4 TopicNets

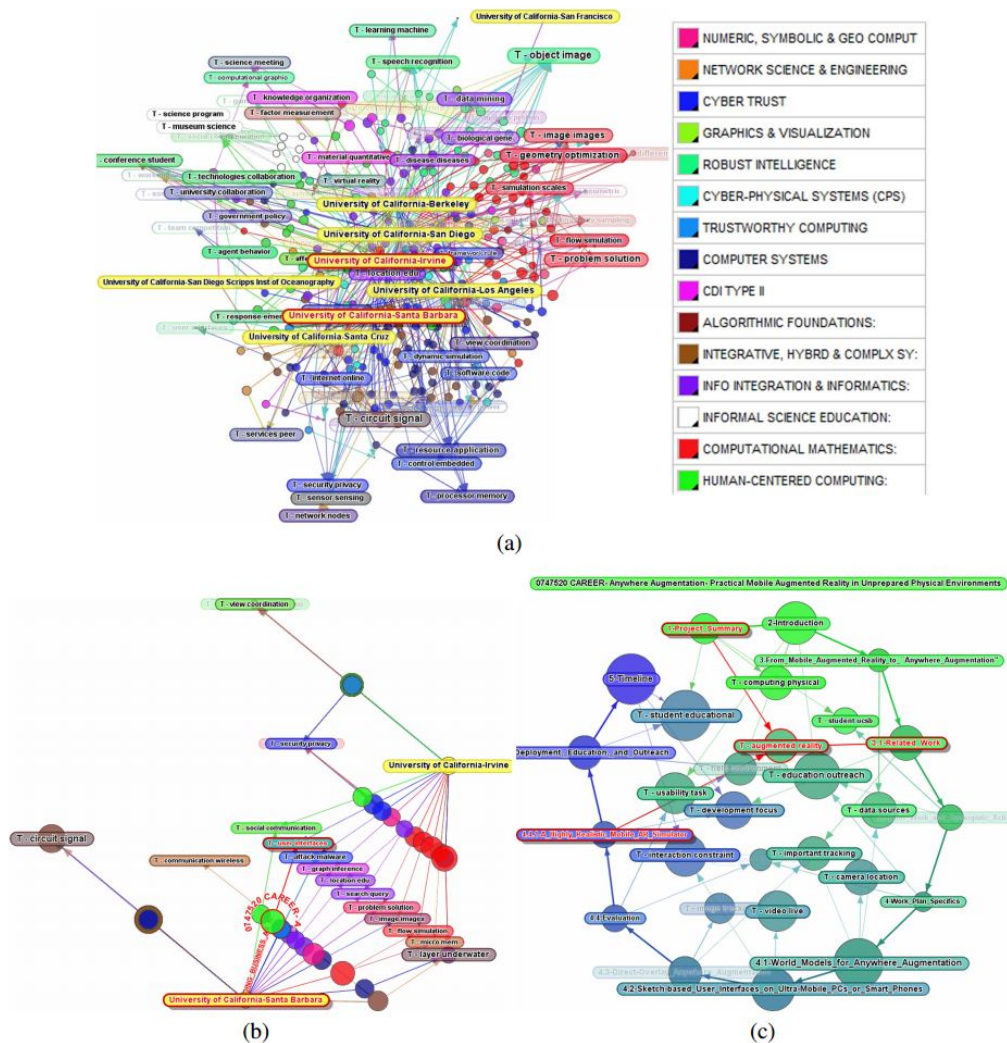


Figure 2.4: TopicNets Visualization [18]

TopicNets is another web-based topic visualization system for large sets of documents that is based on LDA topic models. In contrast to the other systems described before it also includes a visualization of topic similarity over different sections within a single document. Figure 2.4 shows how a user can explore large document collections with TopicNets. The

first image (a) visualizes computer science articles from several Californian universities along with their related topics. The yellow nodes represent the universities and based on their position a user can easily identify which topics are popular for each campus. The color of topic nodes, which are labeled with ‘T’ at the beginning, correspond to their connected documents. As the universities UCSB and UCI are selected and highlighted in the first image, (b) illustrates the filtered visualization, where articles not related to this two universities are removed. The user can now select a certain document in this visualization and gets an intra-document graph (c). The sections of the article are positioned in a circle starting at the top left and the topics are arranged corresponding to their strength within each section [18].

2.1.0.5 FaceAtlas

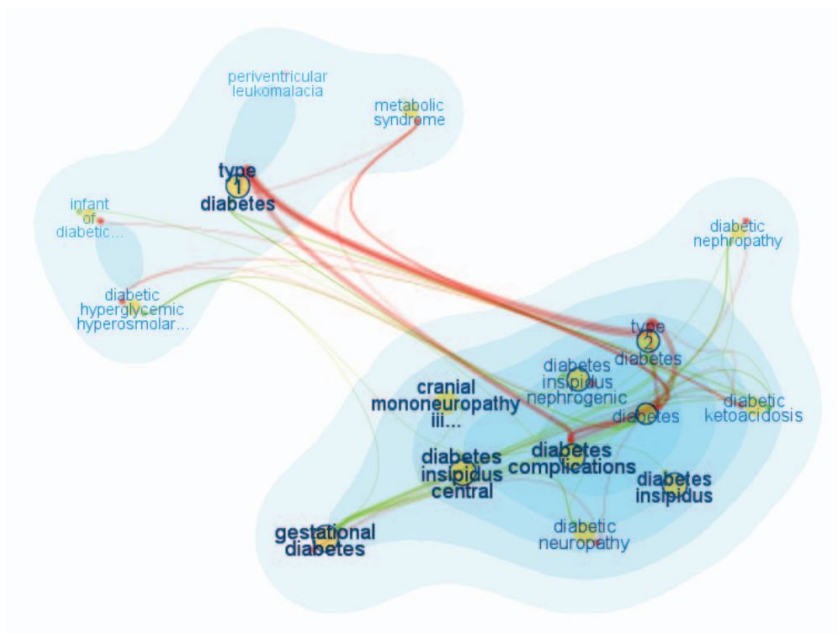


Figure 2.5: FaceAtlas Visualization [6]

Document collections often contain a wealth of multifaceted interconnected data. For example a medical article could contain facets of diagnosis, symptoms, treatments and preventions. The FaceAtlas project aims to visualize the relations of documents in a multi-relational graph based on such facets. By combining a search interface and an interactive visualization the researcher can easily identify relations of documents within a rich text corpora. Figure 2.5 illustrates a FaceAtlas visualization generated by searching for the word ‘diabetes’ in a collection of medical documents. It shows two clusters of the disease, corresponding to type-1 and type-2 diabetes and the links between the facets

express their relations [6].

2.1.0.6 InfoSky

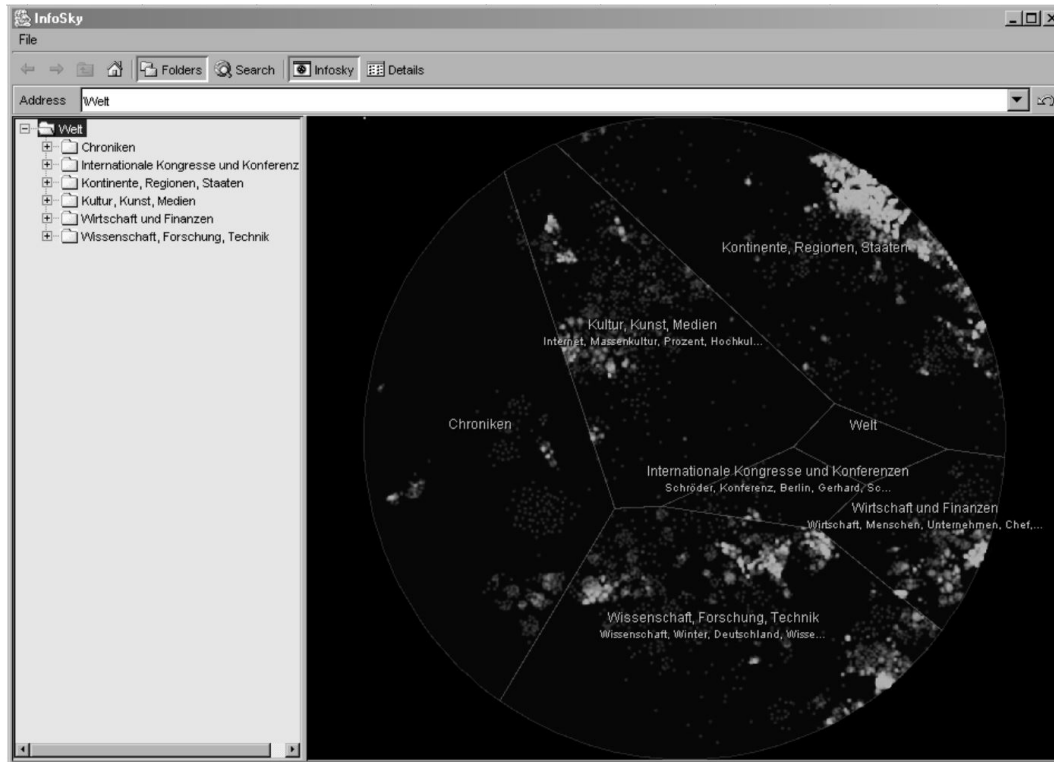


Figure 2.6: InfoSky Visualization [24]

Infosky is an interactive visualization technique that clusters documents of large hierarchically organized document collections and enables users to intuitively search and navigate in deep, complex hierarchies. It uses a telescope metaphor, where documents are represented as stars, collection as constellations and the whole repository as the galaxy. The hierarchy is shown as nested Voronoi areas and the distance represents the topical similarity of documents. Furthermore, color-coding is used to express different document properties. Figure 2.6 shows the InfoSky Visual Explorer, which galaxy is derived from a collection of approximately 109.000 articles from the Sueddeutsche Zeitung. The left side of the explorer window lists the hierarchically structured collections of these articles, classified thematically by the editorial staff of the newspaper. Interactive functions such as zooming and panning reveal details of clusters and stars [24].

2.1.0.7 ThemeScape

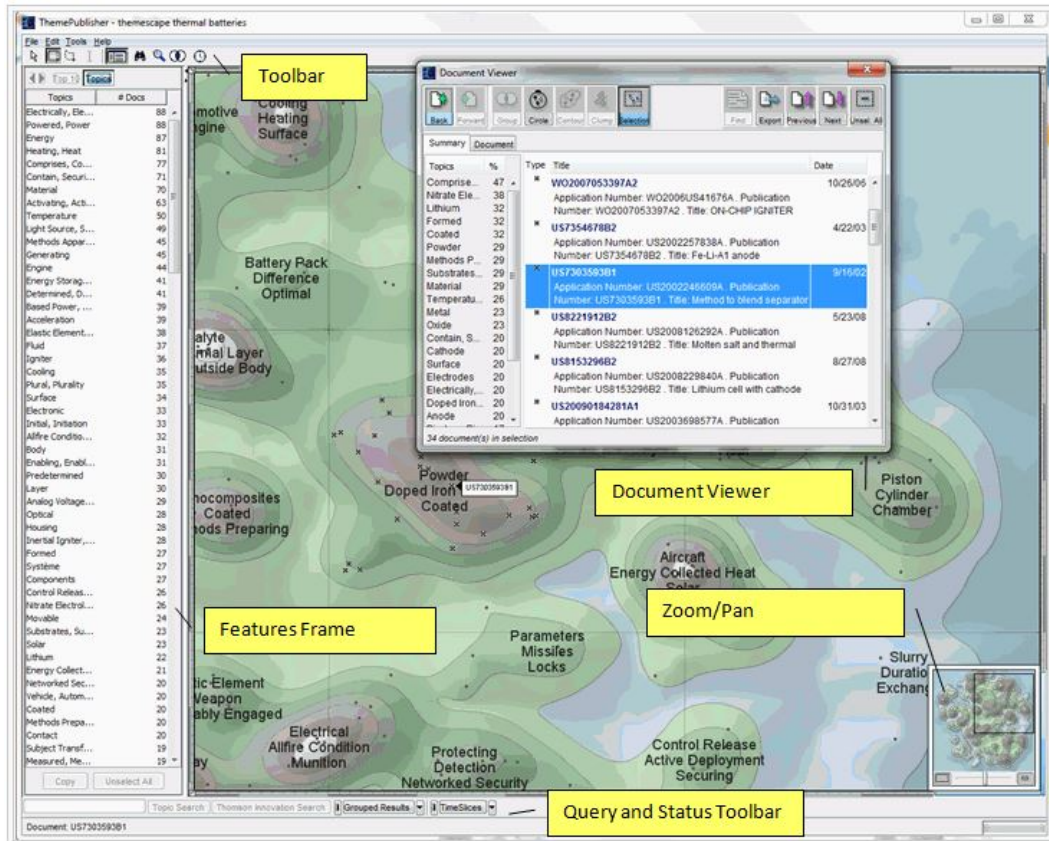


Figure 2.7: ThemeScape Visualization [44]

ThemeScape is another data visualization tool that is designed to handle very large document collections. It uses so called “content maps” to create topographic map-style outputs. The presence of themes in such document collections are visualized as mountains in a relief map, whereas the height of a peak illustrates the relevance of theme and the distance between two peaks on the map correspond to the strength of their relationship. Figure 2.7 shows an example of an output from ThemeScape. The small crosses in the visualization represent the documents and the contour lines illustrate the relative document density of a certain theme. Beside of a zooming functionality that shows details-on-demand, the user can click on a cross to open the corresponding document. This functionality enables researchers to quickly find the most relevant themes and their related documents within a large text corpus and then explore further areas of interests [8].

2.1.0.8 VIBE

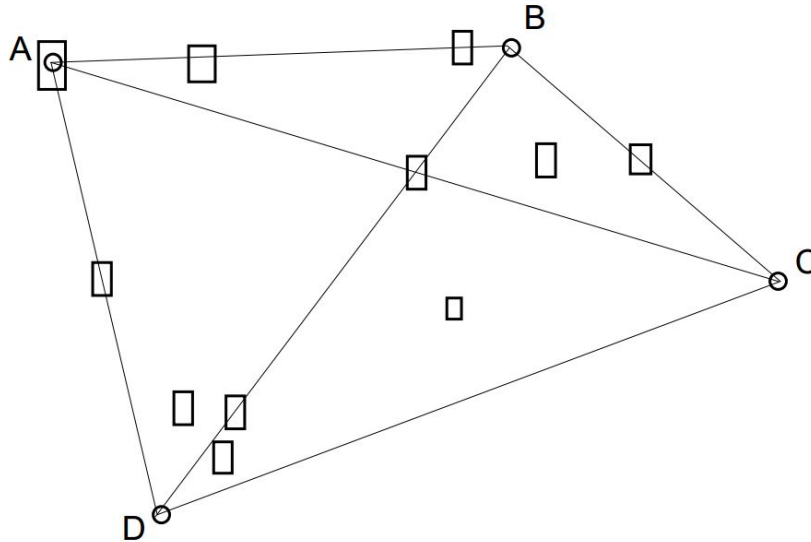


Figure 2.8: VIBE Visualization [35]

The VIBE-System (Visualization By Example) is another approach for illustrating relations between objects like documents or other aspects (see Figure 2.8). In contrast to most other techniques, where the data is positioned according to a set of predefined axes, VIBE lets users span a coordinate system by defining some reference objects on the display. These reference objects are called Point of Interests (POI), which are visually represented by unique icons. The icons of related objects are placed in this information space based on their influence regarding to the POIs. A short distance between a POI and an object indicates a strong influence between them [8, 37]. Furthermore, the size of the icon corresponds to the overall relevance of an objects related to the POIs. For visualizing document collections POIs could be described by keywords and the influence of a specific document for a POI could be calculated by the term frequency of this keyword within the document. Another approach would be to describe the POIs by reference documents and score the influence of related documents within a collection based on their cosine-similarity [35].

2.1.0.9 Jigsaw

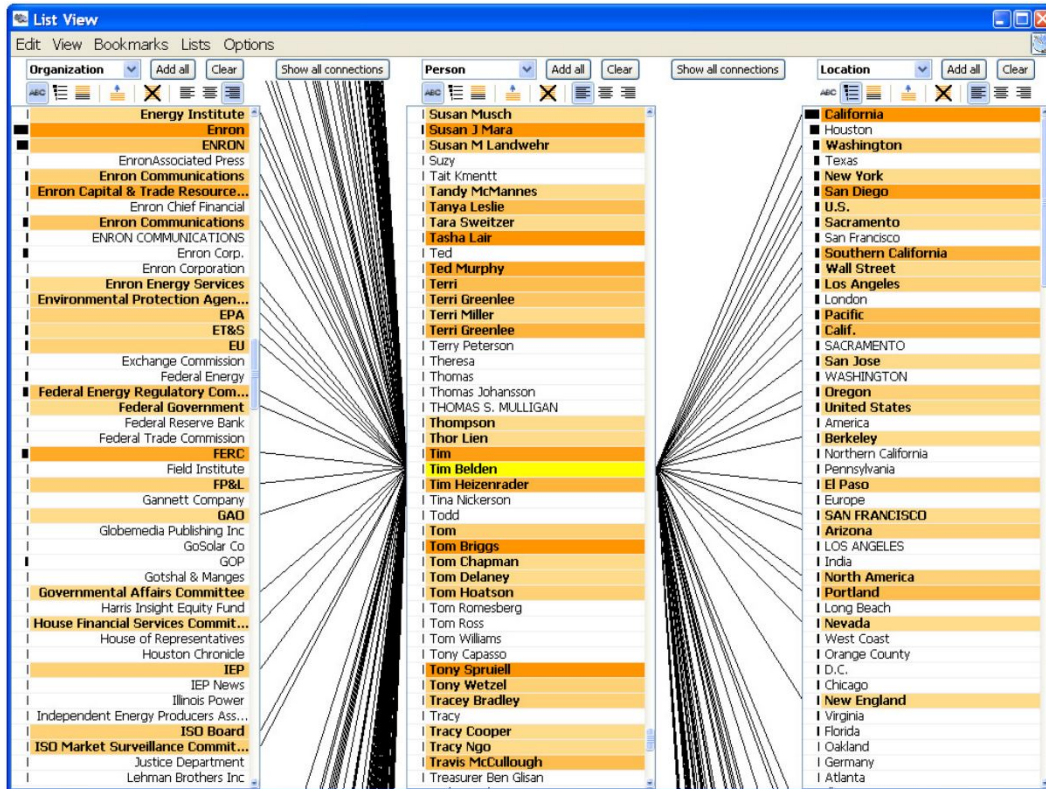


Figure 2.9: Jigsaw List View [17]

The Jigsaw system provides multiple visualization techniques for helping users to analyze large document collections. It combines a search engine with a visual multi-view of results. These views are linked so that actions like filtering, zooming, etc. in one view are applied accordingly to other views as well. Figure 2.9 shows Jigsaw's List View visualization that illustrates connections of 'Tim Belden'. The shaded of orange indicates the strength of the connection, whereas the black bars display the frequency of the keyword within the whole collection of documents. Another view of Jigsaw is the Graph View as illustrated in Figure 2.10. Documents are represented by large white rectangles and colored circles represent entities such as persons [17].

to share their bookmarks. The most popular social bookmarking site is Delicious.com, which is designed to store tagged bookmarks of registered users on the web instead of saving them locally in the browser. In addition, this platform provides a rich and easy to use API, which makes it a suitable source for different visualization approaches [21]. For instance Klerkx and Duval [25] designed a system that combines tag structures and community structures in one visualization using Delicious.com as its primary source. The main idea behind this system is to offer users new ways of finding information that could be relevant for them by inspecting material that has been tagged by other users with similar interests. Figure 2.11 illustrates this visualization which is called Aduna Cluster Map and consists of the following three parts:

1. a filter pane
2. a selection widget
3. a cluster map visualization

The filter pane enables users to search for material that has been tagged by other users, whereas the selection widget represents a list of users and tags that can be checked or unchecked by the user in order to refine the search. Besides of searching and filtering the cluster map visualizations assist the user by intuitively exploring social bookmarks in a playful manner. It visualizes bookmarks related to tags that other users have in common and also those which are inspected, but not in the collection of the other user. For example one can easily identify other users that have saved the same bookmarks or find web pages that are tagged with the same keywords. Additionally, by clicking on bookmarks in the visualization all metadata, associated with this bookmark, is shown to the user.

2.1.0.11 TopicViz

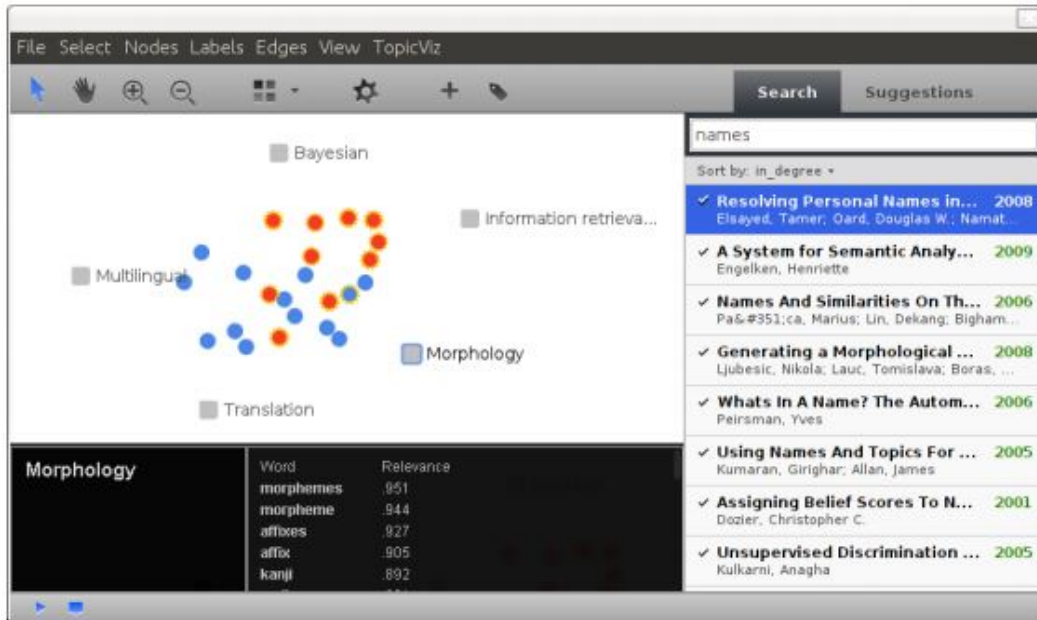


Figure 2.12: TopicViz Visualization [13]

TopicViz is an interactive system that combines a search engine with a force-directed layout visualization of documents. Like TIARA and TopicNets it uses LDA for extracting latent themes within document collections. TopicViz describes a document by a mixture of these themes and provides a visualization that offers a range of interactive functions such as zooming, refining and rearranging of topics and documents [13]. Figure 2.12 shows a TopicViz environment. The nodes represent documents and the boxes refer to their topics. The two different colors of the dots should indicate that these documents were selected by using two different queries. The distance between documents and topics correspond to the strength of the topics within the document. Thus, documents with similar topic profiles will be arranged closer to each other. Users can drag and drop topics to an arbitrary position and the documents are automatically rearranged corresponding to their relevance. Moreover the black box at the lower left shows a selected topic and its related words. A very similar approach called topic maps was developed by Newman et al. [34]. In contrast to TopicViz, which uses force directed layout (FDL) for arranging the documents in the 2-D space, topic maps is also capable of using principal component analysis (PCA) or local linear embedding (LLE) as projection technique. They state, that each of these three projection approaches has their benefits and that it depends on the scope of work which one produces the best and most useful results.

2.1.0.12 Apolo

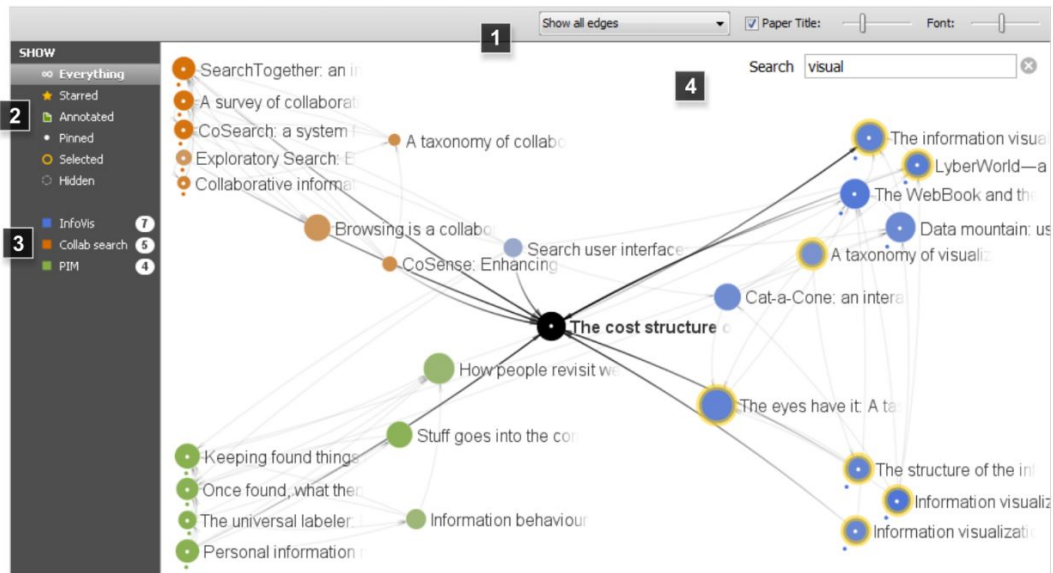


Figure 2.13: Apolo Visualization [7]

Apolo is a system that provides users with a powerful tool to incrementally explore large network data. Instead of using themes and topics, Apolo visualizes a citation network around a specified reference article. An example of such a network is illustrated in figure 2.13, where the citation data is displayed around the article ‘The Cost Structure of Sense-making’. Nodes represent articles and their size is proportional to their citation count. The directed edges of the nodes illustrate their citation relationships. Typically a user starts by selecting a reference article. The Apolo system then computes and shows the top 10 most relevant articles related to this article. Afterwards, the user starts to categorize these articles into different groups he defined before. In figure 2.13 three groups ‘Information Visualization’, ‘Collaborative Search’ and ‘Personal Information Management’ were defined, represented by the color blue, orange and green. These categorized or pinned articles are marked with a white dot in the center of the node. Whenever the group of an article changes, the Apolo system automatically computes the most likely group for all other unpinned articles using a special algorithm called Belief Propagation. The saturation of the unpinned nodes indicates the likelihood of their belonging to a certain group. In addition, the user is able to add more articles, create several subgroups and move articles between them. At the end of this incremental process the visualization represents a landscape of documents and their areas related to one single reference article [7].

2.1.0.13 TileBars

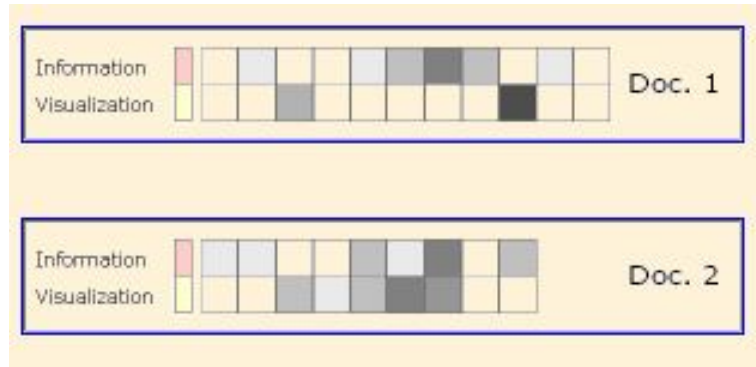


Figure 2.14: TileBar Visualization [20]

TileBars visualize distributions of specific terms within documents, helping an analyst to quickly find relevant documents within a collection. Based on predefined terms or keywords the system computes a rectangle for each document in the collection. Figure 2.14 illustrates how such a rectangle could look like. The rows within the rectangle represent the predefined keywords and the columns refer to sections within the document. The darkness of the cell indicates the frequency of the keyword within the section and the length of the rectangle corresponds to the size of the document itself. With this visualization approach analysts can quickly evaluate relevant documents and sections of documents [20].

2.1.0.14 WordTree



Figure 2.15: WordTree Visualization [47]

WordTree is a visual search tool that depicts multiple parallel sequences of words. It is used to show which words or phrases most likely follow a target word within a given context. In consequence, the phrases are arranged in a tree-like branching structure very similar to decision trees [16]. Figure 2.15 illustrates a WordTree build from Martin Luther King's famous 'I have a dream' speech, using the search term 'I'. The size of the words is proportional to their usage. This technique is often used by search engines such as Google to automatically complete a search term [47].

2.1.0.15 PhraseNet

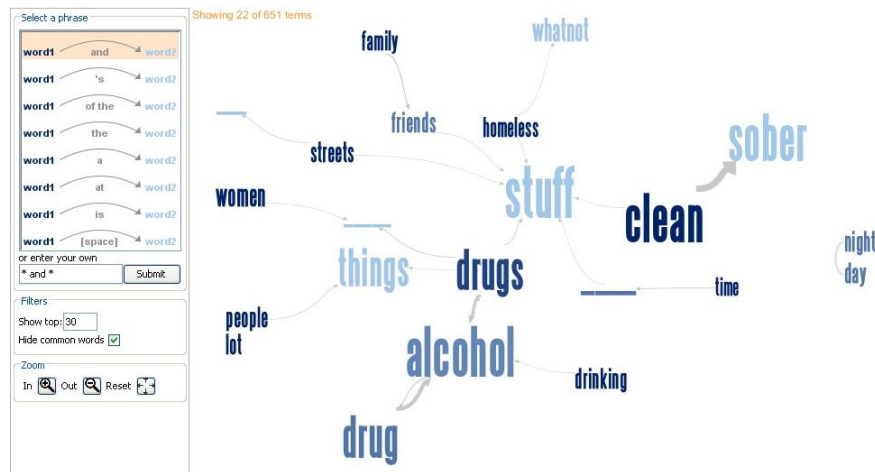


Figure 2.16: PhraseNet Visualization [45]

PhraseNet is a directed link graph that illustrates connections between concepts within a text corpus like papers, books or poems. Beside the direction of the connection, which is expressed by arrows, the width of the links indicates the strength of connected words. Moreover analysts can define a connection term such as 'and', 'or' or just a white space. Figure 2.16 shows a Phrase Net visualization from an interview with homeless people using the connection term 'and' [45].

2.1.0.16 Document Cards

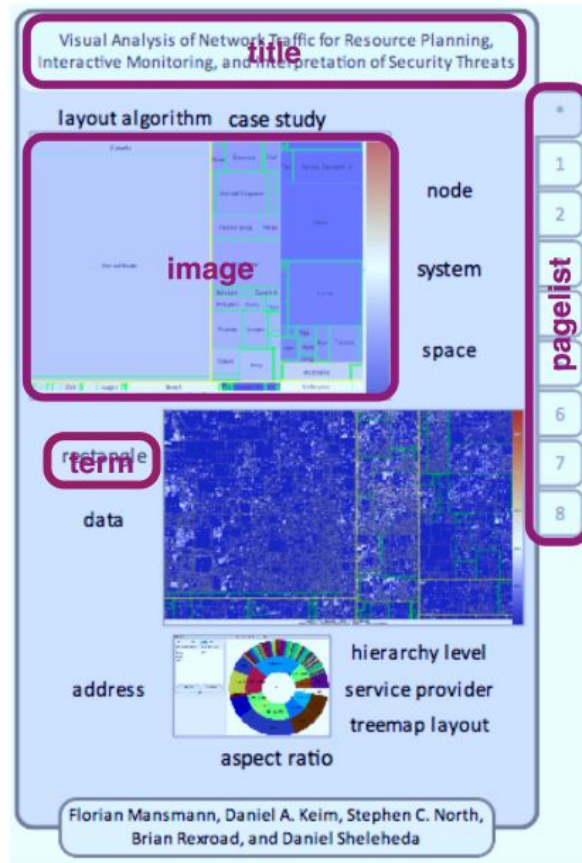


Figure 2.17: Document Cards Visualization -xx needs citation-

Document Cards is a system that represents documents as a mixture of images and text in a compressed way. It offers several interactive functions such as navigation and grouping to help the user explore large documents. Figure 2.17 shows such a Document Card that represents one single document. The marked sections of this card are clickable. For example by clicking on the title a tooltip comes up showing the abstract of the paper. By clicking a certain term on the card images related to this term are highlighted. The amount of information that is displayed on the card can be controlled by a zooming functionality. Zooming out removes less important elements from the visualization and, therefore, avoids information overload. Additionally the user is able to switch to a group view of document cards that represents a collection of documents [43].

2.2 Document Collection Management Systems

Collecting relevant information in form of papers, articles, books and web sites is typically a necessary step when doing knowledge work. Over the past decades, the access to such documents has grown tremendously and, in consequence, the need for organizing research materials has risen. Today various systems address this problem and many of them provide additional features to ease the life of researchers such as:

- citation plugins
- metadata extraction from PDF files
- recommendation engines
- sharing and synchronization of libraries
- optical character recognition (OCR)
- etc.

In the following the four most popular collection management systems used by researchers are compared.

Feature	Mendeley	EndNote	RefWorks	Zotero
Basic software package costs	Free	\$250	\$100	Free
Organization of PDFs and other documents	Yes	Yes	Yes	Yes
Web client	Yes	Yes	Yes	Yes
Desktop client	Yes	Yes	No	Yes
Extraction of embedded metadata	Yes	Yes	No	Yes
Open Web API	Yes	No	No	Yes
Social networks	Yes	No	No	Yes

Table 2.1: Comparison of document collection management systems used by researchers (adapted from [30])

According to a survey published by the Loughborough University, where more than 2000 academic staff and researches participated, Mendeley was the most widely used system in 2014 followed by RefWorks, EndNote and Zotero. The participants claimed that the main benefits of Mendeley are the rich feature set and its ease of use [14]. The most important criterions in order to integrate such a system as the document collection source for PaperViz are:

- It should be free to use (no costs).

- It should have a web client.
- It should provide a rich Web API.

As illustrated in table 2.1 only Mendeley and Zotero fulfill these requirements. After evaluating the web APIs of these two system, Mendeley was selected as the most suitable candidate to be integrated to PaperViz.

2.3 Contributions

The approaches and system described in section 2.1 are all designed to reveal novel information within different granularity levels of document collections. In order to quickly identify relevant pages or paragraphs within large collections of documents it is necessary to combine approaches of different granularity levels. Moreover, combining several approaches enables researchers to examine their repository from different perspectives. Thus, PaperViz includes a set of visualization approaches, which are all interactively connected to each other. The graph-based visualization of PaperViz, which illustrates connections between documents, collections and key phrases is novel but strongly inspired by TopicViz and the VIBE-System. Additionally, a TileBar visualization is integrated to PaperViz enabling the researcher to quickly browse to relevant pages within a document. In order to identify relevant key phrases within collections and documents a tag cloud visualization is included into the PaperViz front end.

Chapter 3

Design

Contents

3.1	Requirements	25
3.2	GUI components	27
3.3	Graph-based visualization	39
3.4	Workflow	63
3.5	Summary	64

The following chapter introduces the design of PaperViz. At the beginning the requirements and concept behind the system are described, followed by a detailed explanation of the graphical layout. As the visualization is the core element of PaperViz, it is outlined in an extra section. The chapter closes with a confrontation of the requirements and the implemented features.

3.1 Requirements

As mentioned in 1.1 the main goal of PaperViz is to speed up the interactive process of finding, evaluating and managing potential resources for scientific paper writing. Based on this goal the requirements of the system were defined as follows:

1. **Provide an intuitive way to quickly browse large document collections.**

Large document collections are often structured hierarchically and consist of many levels of sub collections. When exploring such collections using a common file system, the researcher easily loses the overview of his scientific materials.

2. **Provide high quality information at different levels of details (entire collection, sub collections, documents, single pages).**

Usually researchers have to read at least one section or the abstract of a paper to get an idea of its content. This is typically a very time consuming task, especially if researchers have to deal with large amounts of documents. Therefore, PaperViz should integrate a visualization allowing the researcher to filter out uninteresting or irrelevant documents without reading them.

The visualization approaches and systems described in 2.1 were also designed to address this issue, but most of them can only be applied at one level of granularity. Some of them illustrate the inner structure of a document like TileBars and Document Cards, others are considered to visualize relations between documents like TopicViz or Apolo. But there is currently no system that delivers information at each level of the collection —from the root collection down to single pages of documents.

3. Provide a user-driven visualization model that can be refined and developed based on the researchers interests.

Most of the existing visualization approaches, like ThemeRiver or ParallelTopics, are data-driven, meaning that the structure of the information presented to the user is predefined by the underlying data. They do not provide functions to modify and evolve the visualization based on the user's interests. This forces the users to organize their mental model according to the structure of the data, which is often not ideal when exploring large collections of documents as it requires the researcher to iteratively refine his or her search interests.

4. Assist the user in formulating a suitable search query.

As outlined in 1.1 the process of finding relevant literature is iterative and gets refined at every step by deriving additional search expressions from documents that were recognized as relevant by the researcher. But finding such search expressions typically requires reading the document, which is usually a very time consuming task (again).

5. Support researchers in organizing their useful resources.

After identifying relevant papers, it is desired to organize them in a way to quickly retrieve them for a later detailed exploration.

6. PaperViz should be implemented as web-based system.

Existing document collection visualization systems such as Apolo, TopicViz, Vibe, InfoSky, etc. are all desktop applications. Thus, users have to install additional

software on their devices and, moreover, these desktop applications are mostly platform dependant. Due to these disadvantages, PaperViz should be implemented as web-based system. This enables users to run PaperViz on every device that has a web browser installed without the need of installing additional software.

7. Provide a graphical user interface (GUI) that is intuitive and user friendly.

Navigating, exploring and managing large collections of documents usually results in quite a number of open windows and tabs. A more suitable alternative is to provide a centralized screen composed by multiple coordinated view (MCV).

3.2 GUI components

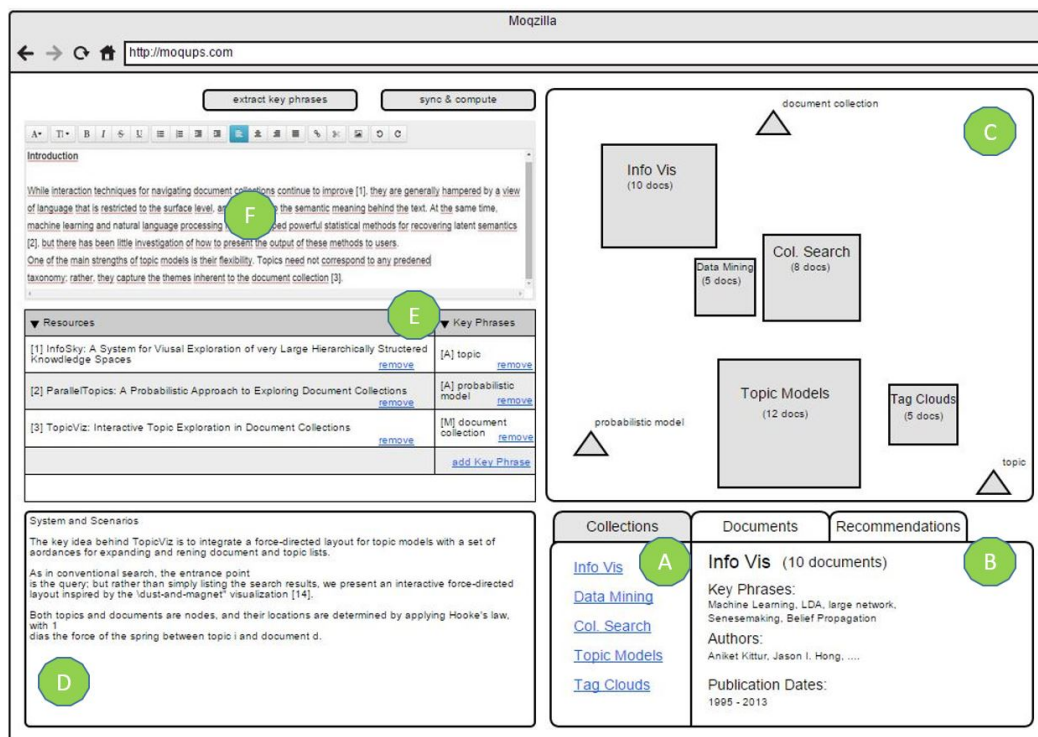


Figure 3.1: First mockup of the GUI

Based on the requirements listed in the previous section, a concept for the graphical layout of PaperViz was created. First, some mockups were designed, which were presented to a group of experts in the field of visualization and usability. The mockup, illustrated in figure 3.1, already gives an idea about the key concept behind the system:

All functions and features of PaperViz should be fit into **one centralized screen** in order to provide a fluent user experience.

The different GUI components should thereby comply with MCV principles. A MCV system is a system that uses two or more distinct views to support the investigation of a single conceptual entity [46]. Thus, changing the state of one GUI component should result in a subsequent change of the other component's state. Moreover, this layout enables researchers to use all functions of the system without switching between windows or tabs such as:

- Browse their document collections (A in figure 3.1)
- Get some metadata-information about documents and collections (B in figure 3.1)
- Retrieve high-quality information about their documents and collections by interacting with the visualization (C in figure 3.1)
- Read sections of documents (D in figure 3.1)
- Manage a list of resources and key phrases (E in figure 3.1)
- Write text and include references to the resource (F in figure 3.1)

Additionally, the visualization was conceived to provide a details-on-demand functionality. By clicking on a rectangle, which represents a specific collection, a visualization of its containing documents is shown (see figure 3.2). A detailed description of this visualization approach is given in 3.3.

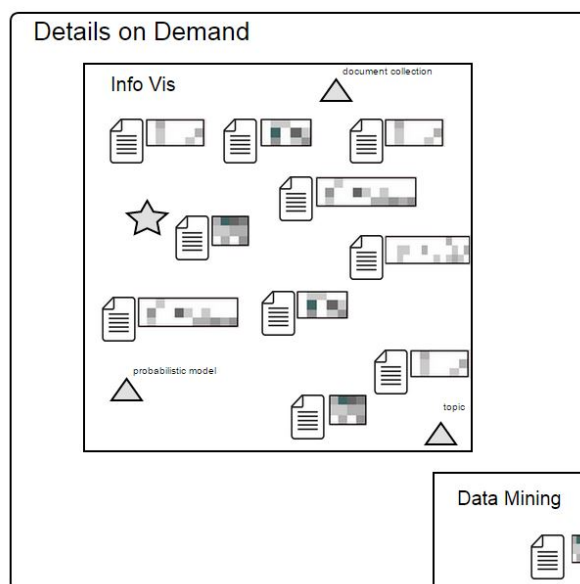


Figure 3.2: Details-on-demand mockup

After several stimulating discussions and further analysis the graphical layout of PaperViz was iteratively improved. The improvements were mainly related to the following three aspects:

1. Structure of document collections

When creating the mockups the assumption was, that document collections are organized in just two levels: the collection level and the document level. But in reality such collections are structured hierarchically. Thus, a collection can contain documents and sub collections.

2. Navigation

It is necessary to integrate some sophisticated navigation functions, especially when the collection is structured in many hierarchies. Therefore, different navigation components should be combined (e.g. tree view, breadcrumbs, graph-based visualization, etc.), which have to be consistent with the navigation depth they illustrate.

3. Visualization

As the core component of PaperViz is the graph-based visualization, much effort has been made to optimize this component. For instance, it needed many iterations to find meaningful and consistent visual encodings, such as color, shape, position and line width, for the different visualization items (collections, documents, key phrases) and their connecting lines. Moreover, additional features have been implemented to improve the informative value of the visualization.

The final result of the graphical layout is slightly different from the first mockup and illustrated in figure 3.3. It consists of the following elements:

- tree view (Figure 3.3 A)
- tag cloud (Figure 3.3 B)
- text editor (Figure 3.3 C)
- list of bookmarks (Figure 3.3 D)
- key phrase box (Figure 3.3 E)
- breadcrumbs (Figure 3.3 F)
- visualization area (Figure 3.3 G)
- metadata information area (Figure 3.3 H)
- ribbon bar (Figure 3.3 I)

The screenshot displays the PaperViz interface with several key components labeled A through H:

- A:** A search bar at the top left containing a list of search results, including 'Personalized Recommendation...', 'Clustering', and 'A Hybrid Resource Recommender Mimicking Attention-Interpretation Dynamics'.
- B:** A 'Tag Cloud' section showing terms like 'similarity', 'familiarity', 'network', 'social', 'people', and 'relationships' in varying font sizes.
- C:** A 'Type your notes here:' section with a text input field and a 'Hevea' icon.
- D:** A 'Bookmarks' section listing two papers: '[Simo2015] Attention-Interpretation Dynamics' and '[Huz2010] Connecting users and items with weighted tags for personalized item recommendations'.
- E:** A 'Key Phrases' section with a search bar and a list of tags including 'information', 'user', 'approach', 'document clustering', 'recommendation', 'logs', 'personalization', 'social', 'people', 'network', 'results', 'familiarity', 'items', 'social', 'personalization', 'people', 'network', 'results'.
- F:** A 'Personalized Recommendation of Social Software Items Based on Social Relations' network visualization showing nodes for 'personalization', 'social', 'familiarity', and 'learning' connected by edges.
- G:** A 'Visualization Area' containing a bar chart and a 'gravity index' slider.
- H:** A 'Metadata Information Area' for the paper 'Personalized Recommendation of Social Software Items Based on Social Relations', including fields for Title, Actions, Source, Type, Year, Authors, Keywords, and Abstract.

Figure 3.3: Final layout of PaperViz

In the following these elements are described in detail, grouped by their purpose.

3.2.1 User management

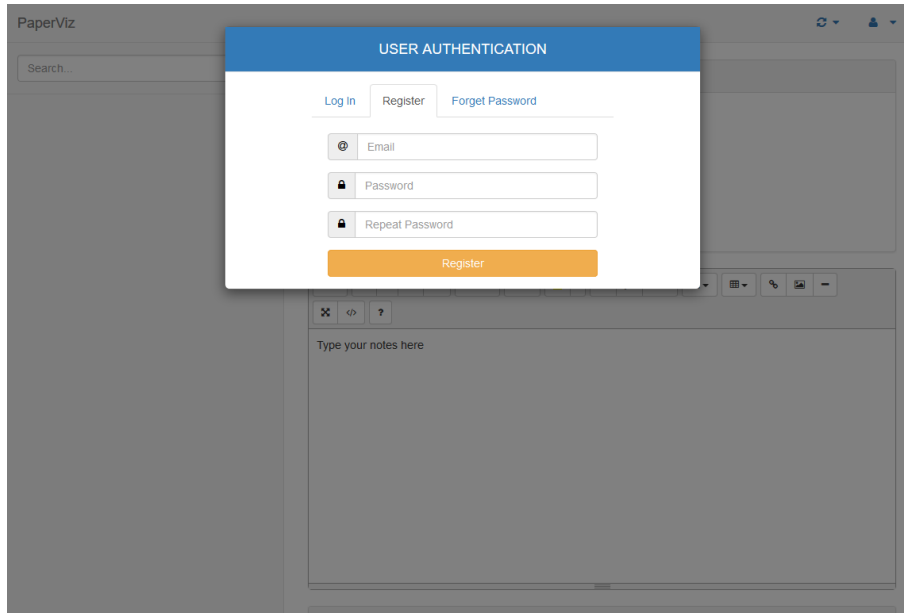


Figure 3.4: Login dialog of PaperViz

PaperViz integrates a user management functionality. This enables the system to link collections to users. Furthermore it allows users to save and load their research work. So when browsing to the PaperViz web site, a login dialog is prompted, illustrated in figure 3.4. This dialog allows users to register a new account and to log into PaperViz with an existing account. The 'Forgot Password' functionality is currently not implemented.

After a successful login, the user's email address is shown at the top right corner of the ribbon bar. Clicking on this email address opens up a sub menu, allowing the users to provide some user specific settings (see figure 3.5).

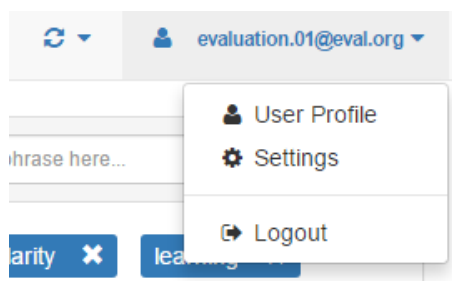


Figure 3.5: The user ribbon of PaperViz

The second ribbon (see figure 3.6), represented by a 'sync' icon, provides functions to:

1. load the user's document collection from external resources (three different options are provided)
2. load and save the current state of PaperViz

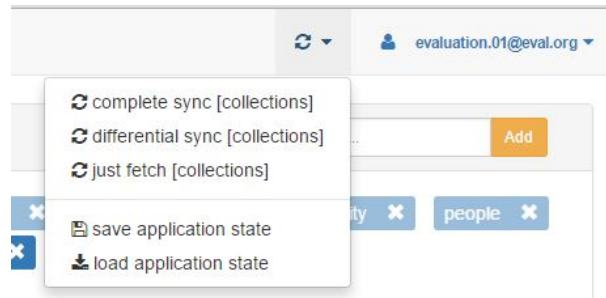


Figure 3.6: The sync ribbon of PaperViz

Further description of these functions and the user specific configuration settings is given in chapter 4.

3.2.2 Navigation

As already mentioned, many effort has been made to provide functions that allow users to quickly navigate within their document collection. Thus, PaperViz provides three different options for navigation:

1. tree view
2. breadcrumbs
3. visualization navigation (click events)

3.2.2.1 Tree view

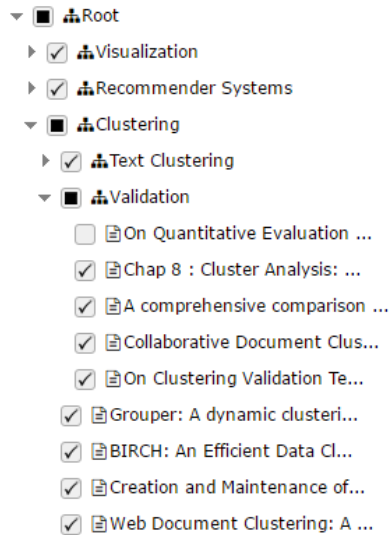


Figure 3.7: The tree view component of PaperViz

The tree view graphically presents the hierarchical structure of the document collection and is primarily used as navigation control (see figure 3.7). It contains collections and documents. Documents are marked with a text icon, whereas collections are labeled with a sitemap icon. By clicking on the arrow icon, child nodes of collections can be expanded and collapsed. Furthermore, each element has a checkbox assigned which can be used as a filter for the visualization.

3.2.2.2 Breadcrumbs



Figure 3.8: The breadcrumbs component of PaperViz

The main purpose of the breadcrumb trails is to allow researchers to keep track of their current location within the collection. It presents the hierarchical path of the selected collection or document back to the root collection (see figure 3.8). In addition, the breadcrumbs can be used to navigate to parent collections just by clicking on a specific element.

3.2.2.3 Actions

Beside the tree view control and the breadcrumbs, the visualization can also be used for navigation purposes. In an MCV system, a change of state to one view must be consistent

with all other views. Thus, all of these three options are strongly linked with each other. So when clicking on a collection or document, regardless of which control is used, the following actions take place:

- The corresponding tree view item is selected and marked with a dark-grey background.
- The breadcrumbs visualize the hierarchical path of the item.
- The collection of the selected item is presented in the visualization area.
- The tag cloud of the selected item is shown.
- Additional information of the selected item is displayed in the metadata information area.

3.2.3 Metadata information area

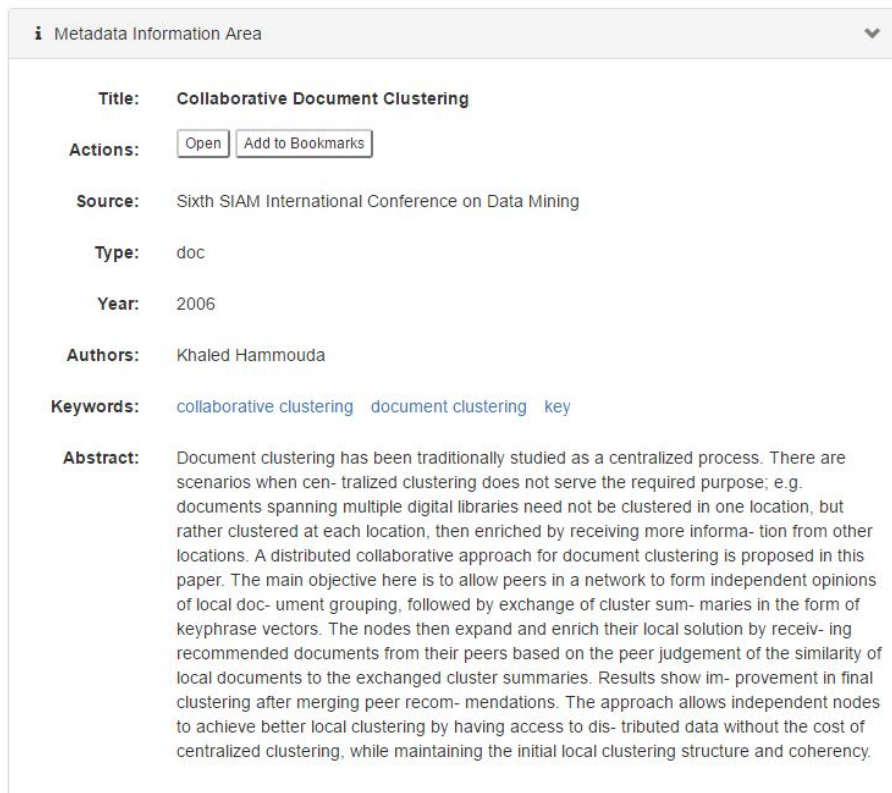


Figure 3.9: The metadata information area of PaperViz

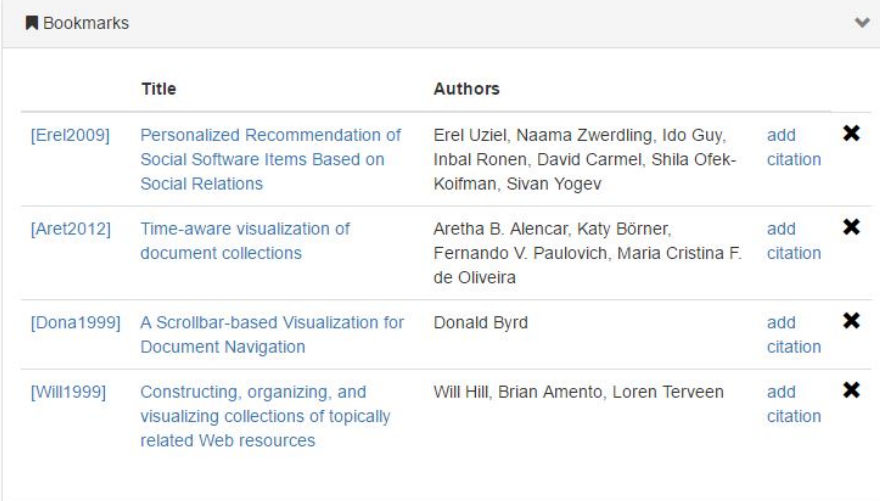
The information presented in the metadata information area varies depending on the type of the selected item. By selecting a collection, only its title, the number of containing sub collections and documents are displayed. For a document, the following information is listed (see figure 3.9): (i) the title of the document, (ii) the source or the journal, (iii) the year of publication, (iv) a list of authors, (v) keywords defined by the authors, (vi) the abstract of the paper (if one is present).

Additionally, this area contains two buttons. By clicking the ‘open’ button, the PDF-file of the document is loaded within the browser and the ‘Add to Bookmarks’ button is used to add the selected document to the list of bookmarks.

3.2.4 Bookmarks and annotations

To support researchers in organizing their documents for a later detailed exploration, PaperViz provides two components.

3.2.4.1 List of bookmarks



	Title	Authors		
[Erel2009]	Personalized Recommendation of Social Software Items Based on Social Relations	Erel Uziel, Naama Zwerdling, Ido Guy, Inbal Ronen, David Carmel, Shila Ofek-Koifman, Sivan Yogev	add citation	✕
[Aret2012]	Time-aware visualization of document collections	Aretha B. Alencar, Katy Börner, Fernando V. Paulovich, Maria Cristina F. de Oliveira	add citation	✕
[Dona1999]	A Scrollbar-based Visualization for Document Navigation	Donald Byrd	add citation	✕
[Will1999]	Constructing, organizing, and visualizing collections of topically related Web resources	Will Hill, Brian Amento, Loren Terveen	add citation	✕

Figure 3.10: The list of bookmarks

As described earlier, users can add documents to the list of bookmarks by clicking the corresponding button in the metadata information area. This list contains five columns (see figure 3.10):

1. **Abbreviation:** The abbreviation is computed by the first 4 characters of the first author and the publication year. By clicking on this item, the corresponding document is loaded in the browser.

2. **Title:** When clicking on an item in the title column, the corresponding document gets selected in PaperViz. That means, that all actions described in 3.2.2.3 are performed.
3. **Authors:** This column displays the list of authors (no further action provided).
4. **Add citation:** By clicking on ‘add citation’ the abbreviation of the document is included in the text editor at the current or last known position of the cursor.
5. **Remove:** By clicking the cross icon, the document gets removed from the list of bookmarks.

3.2.4.2 Text editor

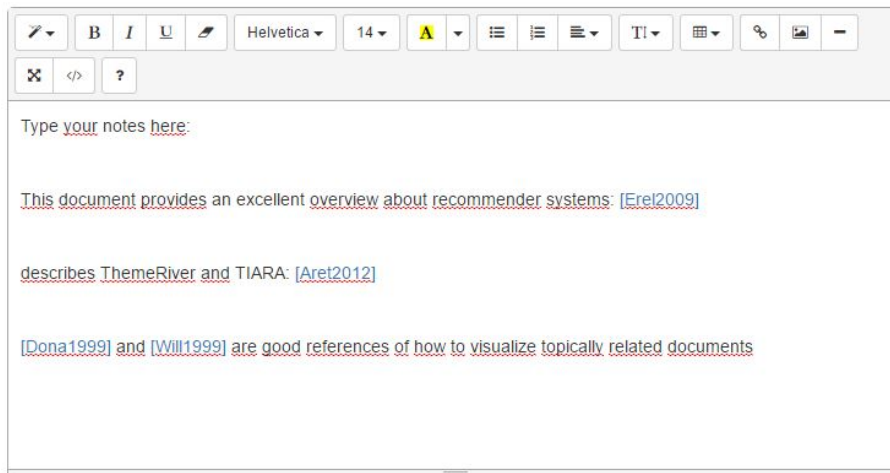


Figure 3.11: The text editor of PaperViz

When creating the concept of PaperViz, the text editor was designed for drafting sections of a research paper. But it turned out, that this component is primarily used for annotating bookmarked documents. The citation function, already explained before, supports this purpose, as documents can be directly linked to notes written with the text editor.

3.2.5 Visualization

PaperViz includes three components related to the visualization aspect, the tag cloud, the key phrase box and the graph-based visualization. As the graph-based visualization is the core component of PaperViz it needs a far more detailed description than other components and, therefore, it is outlined in a dedicated section.

3.2.5.2 Key phrase box

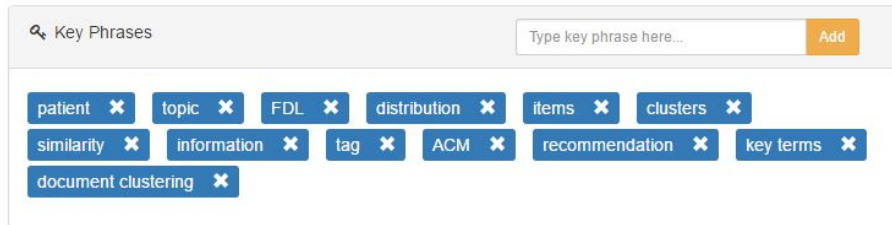


Figure 3.13: The key phrase box of PaperViz

Relevant key phrases are listed in the key phrase box (see figure 3.13). PaperViz provides three ways to add items to this box, namely: (i) by clicking on a word of the tag cloud, (ii) by clicking on an author defined keyword listed in the metadata-information area, (iii) by adding it manually.

These key phrases can then be added to the visualization area via drag and drop (see figure 3.14 –the word ‘similarity’ is currently dragged). When dropping the key phrase into a supported area, visualized with a blue background, it is transformed into a triangle. Furthermore, it is disabled in the key phrase box, ensuring that it can only be added to the visualization area once. Additionally disabled key phrases cannot be removed from the box. The key phrase is automatically enabled as soon as it is removed from the visualization area.

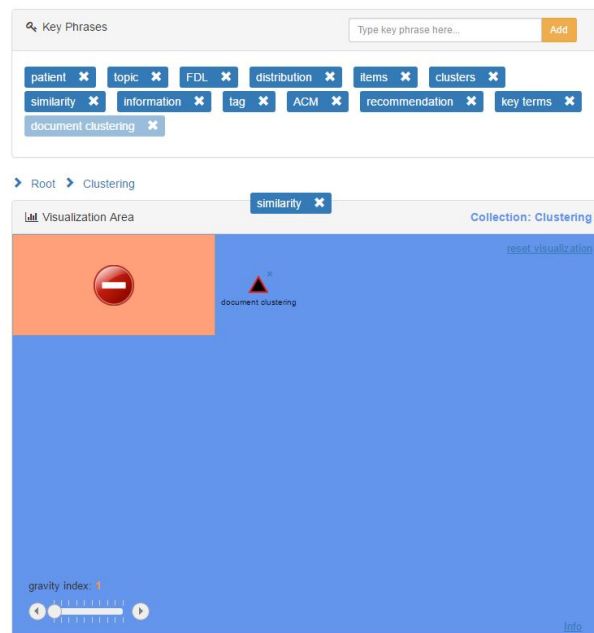


Figure 3.14: Dragging and dropping key phrases

3.3 Graph-based visualization



Figure 3.15: The graph-based visualization of PaperViz

The graph-based visualization component of PaperViz supports rapid sense making on document collections. Thus, it is designed to provide a quick overview of the content of collections, documents and single pages of documents by performing a mapping from a textual to a graphic representation. The aim is to help users to quickly spot documents and sections of documents that would likely be worth inspecting in depth prior to detailed reading. In contrast to many other visualization approaches, where the structure of the information is predefined by the data, PaperViz relies on a user-driven model. That means researchers can model the visualization based on their interests.

3.3.1 Layout

PaperViz allows users to span their own information space by interactively formulating a search query. In this context a search query consists of a set of key phrases and their coordinates within a spatial layout. Thus, key phrases and their positions are defined by the user. Documents and collections are then organized by PaperViz within this layout basing on the search query. The position of a document should thereby reveal information about its strength with respect to the query terms. So if a document strongly relates to a key phrase, the distance between them will be relatively short compared to those with a

weak relationship. This approach enables the researcher to associate the relative positions of documents to their content. In addition to the position, other features like color and size reveal potentially useful knowledge about an object. Furthermore, PaperViz provides additional mini-visualizations depicting some extra information of inspected text blocks such as the distribution of keywords. So by using their visual skills, researchers can quickly examine their collections. But in order to present such information using this visualization approach, PaperViz leverages a text mining engine provided by Sensium¹ as Software as a Service (SaaS) that scores key phrases based on their frequency count within a text block. Thus, these scores indicate relationship strengths between key phrases and text blocks. The text mining engine thereby assigns scores ranging from 0 to 1, where a score of 0 indicates no relationship and 1 states a very strong relationship. So comparing key phrases with documents and collections yields a matrix of scores. Each cell of this matrix represents the relationship strength between a key phrase and a certain item of a collection (document or sub collection).

$$S_{m,n} = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & s_{m,2} & \cdots & s_{m,n} \end{pmatrix} \quad (3.1)$$

$S_{m,n}$...	score matrix
m	...	number of key phrases
n	...	number of collection items
$s_{i,j}$...	relationship strenght (score) between the key phrase i and the collection item j

A row vector of this matrix represents the relationship strenghts between one key phrase and all items of the currently selected collection.

$$S_{KP_i} = \begin{pmatrix} s_{CI_1} \\ s_{CI_2} \\ \vdots \\ s_{CI_m} \end{pmatrix} \quad (3.2)$$

S_{KP_i}	...	score vector of key phrase i
------------	-----	--------------------------------

Analogously a column vector expresses the relationship strenghts between one document or collection and all defined key phrases.

¹<https://www.sensium.io/>

$$S_{CI_j} = \begin{pmatrix} s_{KP_1} \\ s_{KP_2} \\ \vdots \\ s_{KP_n} \end{pmatrix} \quad (3.3)$$

S_{CI_j} ... score vector of collection item j

While the elements of a score vector express relationship strengths, the **l_1 norm** of this vector measures the importance or relevance of the item.

$$R_{KP(i)} = \sum_{j=0}^{j=n} |s_{CI_j}| \quad (3.4)$$

$$R_{CI(j)} = \sum_{i=0}^{i=m} |s_{KP_i}| \quad (3.5)$$

R_{KP_i} ... relevance of key phrase i with respect to all collection items

R_{CI_j} ... relevance of collection item j with respect to all key phrases

While the relationship strength between a key phrase and a given collection item is expressed by the positions of these items, the relevance of an item is visualized by its color intensity.

3.3.2 Visual encoding

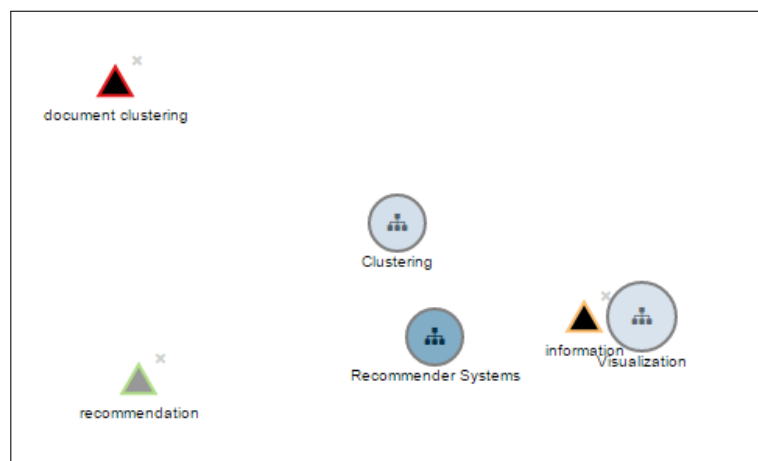


Figure 3.16: Shapes: triangles and circles

In the following the features, which are used for the visual encoding of key phrases and collection items are, explained in detail.

3.3.2.1 Shape

The shape of an object is used to differentiate node types. Key phrases are visualized as triangles, whereas collection items are represented by circles. To distinguish collections from documents the same icons as for the tree view component are used. A sitemap icon indicates a collection, while documents are labeled with a text icon (see figure3.16).

Key phrases can be added to the visualization by dragging them from the key phrase box into an arbitrary position in the visualization area. Moving a triangle directly within the visualization area is also supported. By clicking the nearby cross icon it is removed from the visualization and enabled in the key phrase box. The set of key phrases currently added to the visualization will be expressed as “search query” in the further course of this thesis. By adding at least two key phrases to the visualization area, the collection items of the currently selected collection are displayed.

3.3.2.2 Color

Each triangle has a unique border color assigned, which serves as an exclusive identifier. The qualitative color palette used by PaperViz for this unique colors was generated by ColorBrewer ².

3.3.2.3 Intensity



Figure 3.17: Relevance is visualized by intensity

²<http://colorbrewer2.org/>

Both, key phrases and collection items have a color intensity assigned representing the relevance of the object. The color intensity of a key phrase ranges from white to black and expresses its relevance according to the selected collection. The darker it is, the higher the relevance of the key phrase. In order to maximize the meaningfulness of this property, the intensity scales is normalized to values between 0 and the score of the most relevant key phrase of the search query. Thus, the key phrase gaining the highest relevance gets always the highest possible intensity (black) assigned. For instance, figure 3.17 illustrates four key phrases. It is easy to recognize ‘items’ as the most relevant key word in this query, followed by ‘similarity’. In contrast to the unique color, the intensity of a triangle may change. This occurs on following events:

1. An even more relevant key phrase is added to the visualization.
2. The currently most relevant key phrase is removed.
3. Another collection is selected by the user.

Figure 3.18 illustrates such an event, where the key phrase ‘user’ has been added.



Figure 3.18: Key phrases —intensity changed

Similar to the key phrases, the intensity of a circle expresses the relevance of a collection item according to the search query. The intensity scale is normalized as well and ranges from light blue to dark blue. By changing the search query, the intensity of the circles change accordingly. Figure 3.17 illustrates three collections, whereby the collection ‘Recommender Systems’ reflects the search query the most, indicated by its higher intensity (dark blue).

3.3.2.4 Position

In contrast to the intensity, expressing the relevance of an item, the position of a circle visualizes the relationship between a collection item and the key phrases. Thus, the location gives an indication about the contents of documents and collections. If a document

only relates to one key phrase of the search query, it will be placed very close to it. If it yields a 50% match for each of two key phrases, the corresponding circle will be located halfway between them. Figure 3.19 illustrates an example of such a layout.

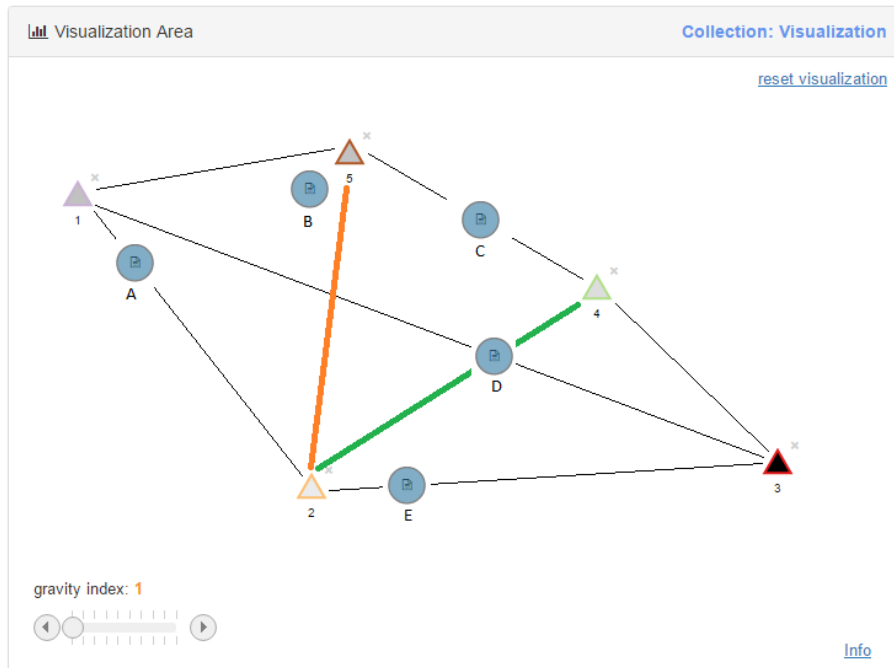


Figure 3.19: Positioning interpretation

Based on the positions of the circles and triangles, following statements can be made:

- All documents or collections, which are placed on the left side of the orange line, are influenced by key phrase '1'.
- All documents or collections, which are placed on the right side of the green line, are influenced by key phrase '3'.
- Document 'A' is influenced by the key phrases '1' and '2'. The influence of key phrase '1' is stronger.
- Document 'E' is influenced by the key phrases '2' and '3'. The influence of key phrase '2' is stronger.
- Document 'C' is equally influenced by the key phrases '5' and '4'.
- Document 'B' is influenced by the key phrases '1' and '5'. Furthermore, at least a third key phrase relates to this document, probably key phrase '2'.

- Document ‘D’ could be placed at this position for several reasons:
 - influenced by the key phrases ‘1’ and ‘3’
 - influenced by the key phrases ‘2’ and ‘4’
 - influenced by all four key phrases ‘1’, ‘2’, ‘3’, ‘4’
 - other possible combinations

Thus, it is not always easy to identify the key phrases that are decisive for the placement of the document or collection circle within the layout. In addition, the statement that the distance between a triangle and a circle expresses their relationship strength is not always true. As figure 3.19. shows, document ‘D’ could be just influenced by the key phrases ‘1’ and ‘3’, but the nearest key phrase is ‘4’. To address these problems, PaperViz provides some additional mini visualizations, which are described later in this section.

However, the position of a circle is computed by a combination of its score vector and the current locations of the corresponding key phrases. In other words, it is defined by a set of forces acting upon the circle. The magnitudes of these forces are represented by the score vector, whereas the directions are defined by the coordinates of the key phrases.

$$POS_{CI_j} = \begin{pmatrix} X \\ Y \end{pmatrix}^{CI_j} = \begin{pmatrix} s_{KP_1} \\ s_{KP_2} \\ \vdots \\ s_{KP_n} \end{pmatrix} \div \sum_{i=0}^n |s_{KP_i}| \cdot \begin{pmatrix} X_{KP_1} & X_{KP_2} & \cdots & X_{KP_n} \\ Y_{KP_1} & Y_{KP_2} & \cdots & Y_{KP_n} \end{pmatrix} \quad (3.6)$$

So to compute the position of a circle, first the score vector is divided by its l_1 norm. The resulting vector represents the normalized magnitudes of the forces acting upon the circle. By dividing the score vector by its l_1 norm the sum of these magnitudes is always 1. The normalized vector is then multiplied with a matrix consisting of the X and Y coordinates of the corresponding key phrases.

Figure 3.16 shows a simple example of a PaperViz visualization. Based on the positions of the three circles representing collections, a researcher can easily derive valuable information about their content. The ‘Visualization’ collection strongly relates to the key phrase ‘information’, because of the short distance and is not influenced by the key words ‘document clustering’ and ‘recommendation’. The circle representing the ‘Clustering’ collection is located exactly between two key phrases, which indicates a 50% match for the key words ‘document clustering’ and ‘information’. Finally, the position of the ‘Recommender Systems’ collection indicates that this collection is only influenced by

‘recommendation’ and ‘information’ and does not relate on the key phrase ‘document clustering’.

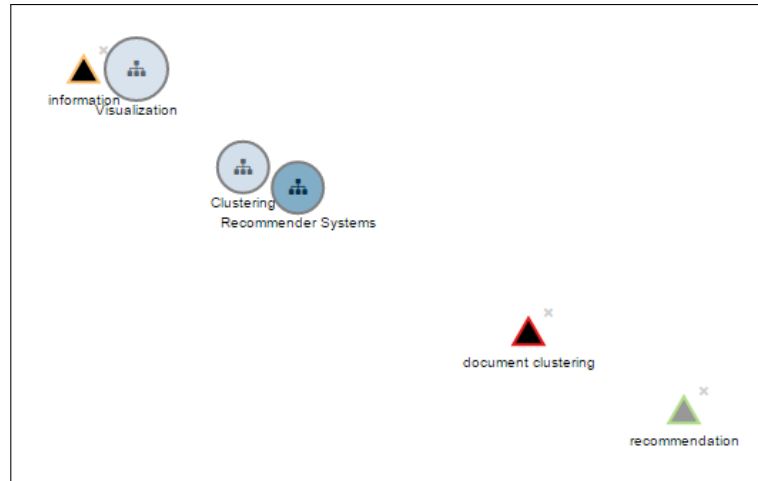


Figure 3.20: Example: badly placed key phrases

So by looking at the position of a circle, the researcher quickly gains knowledge about the content of collections and documents. But in order to yield a valuable visualization the placement of the key phrases is crucial. Figure 3.20 illustrates a visualization containing the same items as figure 3.16, but with different key phrase positions. The information that can be derived from this visualization decreased significantly in contrast to figure 3.16. It is now impossible to tell whether the collections ‘Clustering’ and ‘Recommender Systems’ relate to ‘document clustering’, ‘recommendation’ or both of them.



Figure 3.21: Example: overlapping visualization elements

Another important point that has to be mentioned is that the positioning algorithm is designed to avoid overlaps of visualization elements. Suppose two documents are only related to one key phrase. Using formula 3.6 to compute the final position of documents and collections, the key phrase and both documents would be placed at the exact same position impairing the visualization’s legibility dramatically (see figure 3.21). Furthermore,

the user can't move the key phrase to a different position, as it is overlapped by two documents. So to avoid this behavior, a second force was introduced, acting between all elements of the visualization. This repelling force pushes elements apart from each other if an overlap was identified. The simplified algorithm to avoid overlaps of two circles can be formulated as followed:

1. The circles are defined by a position (X and Y) and a radius (r).
2. The algorithm computes the Euclidean distance between the circles.

$$D = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} \quad (3.7)$$

3. Then the sum of the radii of the two circles is calculated.

$$R = r_1 + r_2 \quad (3.8)$$

4. If the sum of the radii is greater than the distance, an overlap was identified.

$$R > D \implies \text{overlap!} \quad (3.9)$$

5. If an overlap was identified, new centers for the circles are computed.

$$L = R - D \quad (3.10)$$

$$dx = \left(\frac{R - L}{2L} \right) \cdot (X_2 - X_1) \quad (3.11)$$

$$dy = \left(\frac{R - L}{2L} \right) \cdot (Y_2 - Y_1) \quad (3.12)$$

$$\begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}^{New} = \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix}^{Old} - \begin{pmatrix} dx \\ dy \end{pmatrix} \quad (3.13)$$

$$\begin{pmatrix} X_2 \\ Y_2 \end{pmatrix}^{New} = \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix}^{Old} + \begin{pmatrix} dx \\ dy \end{pmatrix} \quad (3.14)$$

This algorithm is applied to each pair of visualization objects until no overlap is identified.

3

³algorithm adapted from <https://gist.github.com/dannyko/4618287>

3.3.2.5 Size

The diameter of a circle represents the number of documents contained in a collection or the number of pages from a document. It is computed by following empirical formula:

$$r_j = \log(\text{Size}_{CI_j}) \cdot 7 + 5 \quad (3.15)$$

r_j ... radius of the circle in pixel representing the collection item j

A linear function would not yield meaningful results, as the size of documents and collections can differ significantly.

3.3.3 Interactions and Details-on-demand

The visualization supports two types of interactions allowing the researcher to inspect an element of the visualization more detailed.

1. Mouseover

By placing the mouse over a circle additional information regarding this item is displayed in form of a bar chart and connecting lines (see 3.3.3.1 and 3.3.3.2). Furthermore, other circles become transparent (see figure 3.15). This mouse event also works for triangles representing key phrases.

2. Click events

The subsequent action when clicking on a circle depends on its type. When clicking on a circle representing a collection, the user “navigates” into this collection. Meaning the containing sub collections and documents are visualized. This enables researchers to browse into deeper levels of their repository. By clicking on a document, a TileBar visualization of the document gets displayed, revealing information about single pages of the document (see section 3.3.3.3). As mentioned already in section 3.2.2.3, by clicking on a circle, further actions related to other components of PaperViz are performed as well.

3.3.3.1 Bar chart

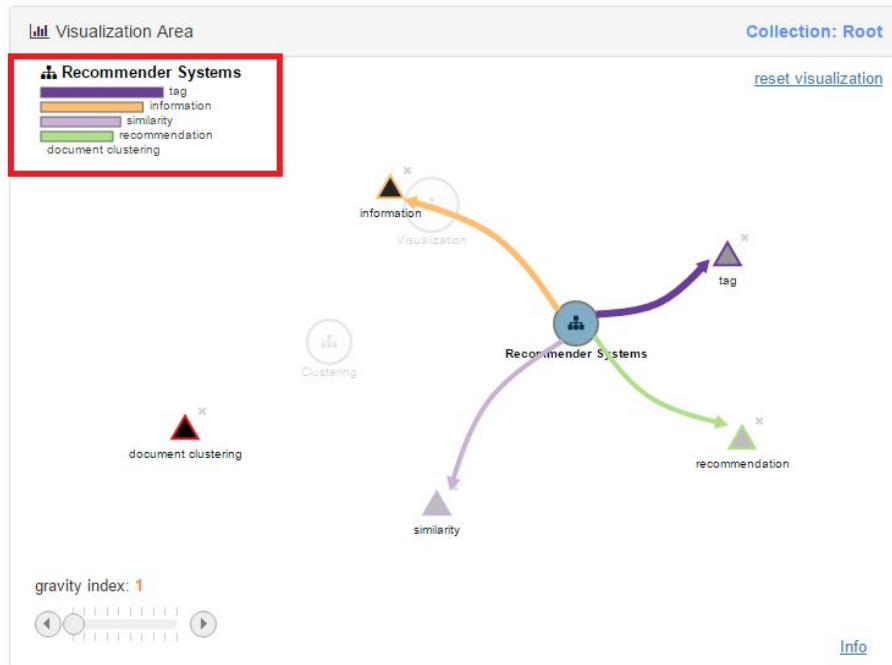


Figure 3.22: Bar chart for collections

As outlined in section 3.3.2.4, it is sometimes not possible to identify the responsible forces which are decisive for the placement of documents and collections. In order to reveal this information a bar chart was integrated to the visualization. It gets displayed at the top left corner when hovering over a triangle or circle and shows the relationship strengths between key phrases and documents or collections. To be more specific, the bar chart visualizes the score vector of the hovered element on a normalized scale. Figure 3.22 shows a bar chart of the 'Recommender Systems' collection. The colors of the bars are equal to the unique color of the corresponding key phrase. By looking at the bar chart, one can easily identify 'tag' as the key phrase with the strongest relationship, followed by 'information', 'similarity' and 'recommendation'. The key phrase 'document clustering' is not relevant to this collection and therefore no bar is displayed.



Figure 3.23: Bar chart for key phrases

When hovering over a triangle, the scores of the most relevant sub collections or documents are visualized regarding the inspected key phrase. Figure 3.23 illustrates a bar chart for the key phrase ‘similarity’. As collections and documents do not have a unique color, the bars are colored according to the unique color of the hovered key phrase. The icon next to the bar indicates whether the object is a collection or document.

3.3.3.2 Connecting lines

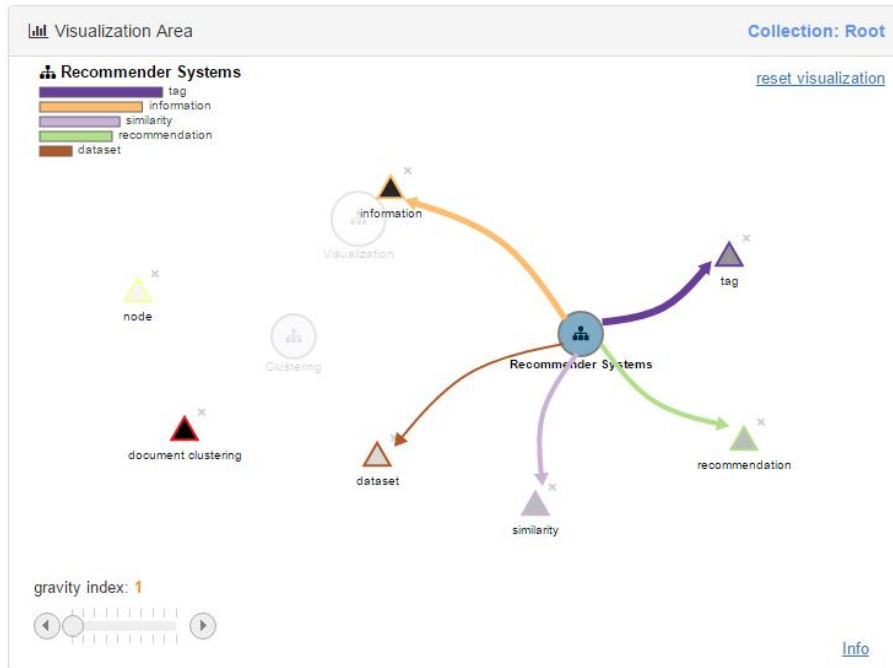


Figure 3.24: Connecting Lines

Drawing connecting lines between elements of the visualization is just another way to visualize the score vector of a hovered element (see figure 3.24). Thus, they illustrate the same information as the bar chart, but in a different form. In contrast to the bar chart visualization, these edges enable users to quickly spot the positions of related elements. The width of an edge thereby indicates the strength of the relationship between two objects. One could also state that these edges illustrate the forces acting upon the hovered object. In order to avoid overlaps, the lines are not straight but curved. This characteristic was implemented by introducing an additional intermediate point on the path of an edge (figure 3.25).



Figure 3.25: Connecting lines —overlaps are avoided

3.3.3.3 TileBars



Figure 3.26: TileBars

By clicking on a document circle, its TileBar visualization gets displayed, showing the distribution of key phrases within the pages of the selected document. This allows the researcher to quickly identify relevant pages of large documents. Figure 3.26 illustrates a TileBar visualization of PaperViz for the article ‘Connecting users and items with weighted tags for personalized item recommendations’. The columns represent the page numbers of the article, whereas the rows are defined by the search query. The intensity of the corresponding cell expresses the frequency of a key phrase for a specific page. So if the researcher is interested in the term ‘similarity’ he can quickly identify page 6 and 7 as potentially relevant. By clicking on the corresponding cell, the article is loaded in a separate window and automatically scrolled to the desired page. This feature is realized by adding the ‘page’ parameter to the URL of the PDF-file (see figure 3.27), which is supported by almost every modern PDF-viewer. Unfortunately, an option of highlighting a specific key phrase within the PDF-file is currently not provided.

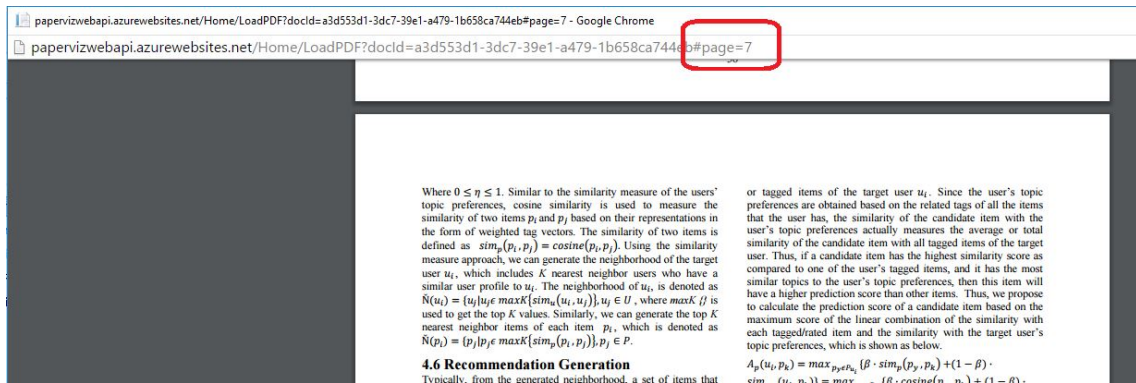


Figure 3.27: PDF page parameter

3.3.4 Additional controls

At three corners of the visualization area additional controls are placed, providing some useful features.

3.3.4.1 Gravity slider

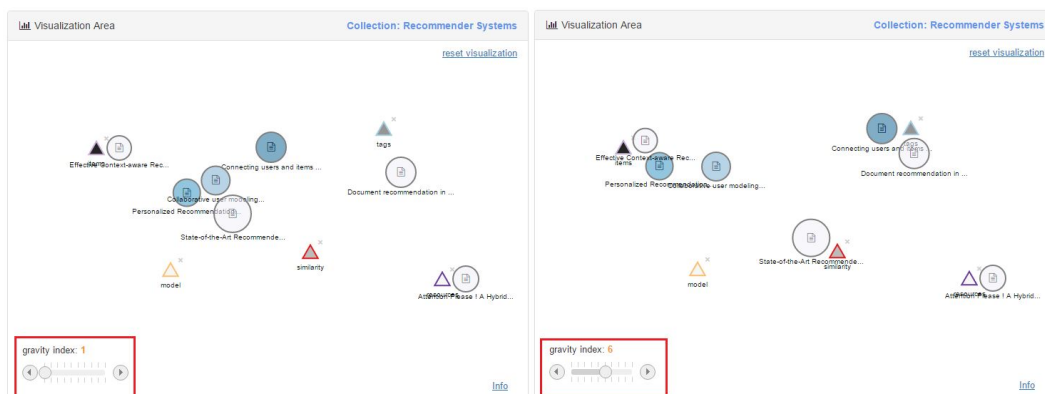


Figure 3.28: Gravity slider

By increasing the gravity index using the slider located at the bottom left, circles will float to the direction of their strongest forces. In other words, they will move towards the key phrase that best describes the corresponding document or collection. This can be useful to gain a better separation of collection items and, thus, increase the meaningfulness of the visualization (see figure 3.28). The gravity slider feature was implemented by performing an exponentiation on the scores contained in the score matrix. The exponent is thereby the gravity index.

3.3.4.2 Buttons

Two additional buttons are integrated to the visualization area. The ‘reset visualization’ button is located at the top right corner of the visualization area. By clicking it, all key phrases are removed from the visualization area. Before this button was implemented a user had to remove each key phrase one by one when reformulating a search query using completely different key phrases. The ‘info’ button is located at the bottom right corner and opens up a new browser window containing useful information about the visualization elements and properties.

3.3.5 Usage scenarios

In the following some usage scenarios are outlined of how researchers can utilize the PaperViz visualization in order to gain useful knowledge about their collection. The examples are based on a sample collection containing real publications related to the topics ‘visualization approaches’, ‘clustering’ and ‘recommendation systems’.

3.3.5.1 Scenario 1

Suppose a researcher is interested in approaches that reveal novel information about users of a certain platform. His goal is to identify relevant documents within the sample collection. As he is not very familiar with this research field, he adds the terms ‘approach’, ‘information’ and ‘user’ to the key phrase box of PaperViz. Afterwards he drags these terms from the key phrase box and drops them into the graph-based visualization area at arbitrary positions, resulting in the following illustration (see figure 3.29).

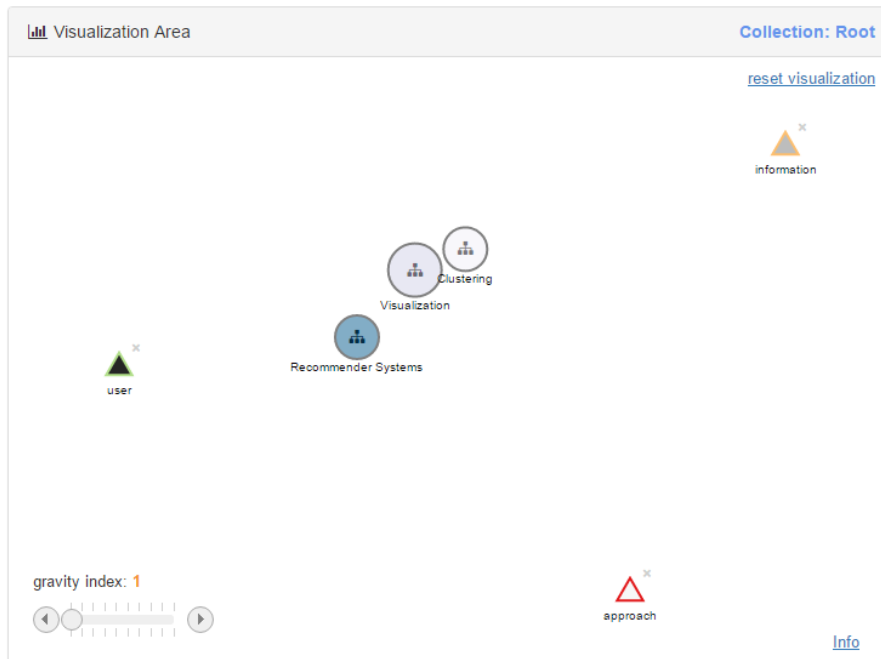


Figure 3.29: Scenario 1: PaperViz visualization

By looking at this visualization, the researcher can easily identify the ‘Recommender System’ collection as the most relevant one due to its higher intensity. Furthermore its position indicates that this collection relates on all three key phrases, whereby ‘user’ has the strongest relationship to the collection followed by ‘information’ and ‘approach’. This information is also visualized by the bar chart and the connecting lines when hovering over the collection circle (see figure 3.30).



Figure 3.30: Scenario 1: PaperViz visualization (2)

Next he clicks on the circle in order to navigate into the ‘Recommender Systems’ collection. By looking at the intensity of the circles corresponding to documents of this collection, he can identify one document that reflects his search query the most, namely ‘Connecting users and items with weighted tags for personalized item recommendation’ (see figure 3.31).



Figure 3.31: Scenario 1: PaperViz visualization (3)

Thus, this document is a good candidate for being inspected in depth. By clicking on the circle of this document, a TileBar visualization shows up revealing the inner structure of the documents regarding to the three key phrases defined by the researcher (see figure 3.32).

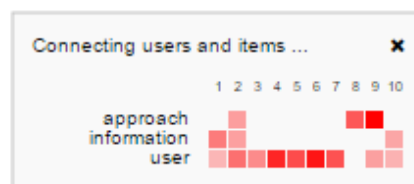


Figure 3.32: Scenario 1: TileBar visualization

As this visualization indicates that only page 2 of the document contains all three key phrases, it might be worth reading this page. By clicking on a corresponding cell, PaperViz loads the document within a separate browser window and automatically scrolls to page 2. As figure 3.33 illustrates, this page contains the ‘Related work’ section of the paper, which is typically a good starting point for researchers to get familiar with a certain research field.

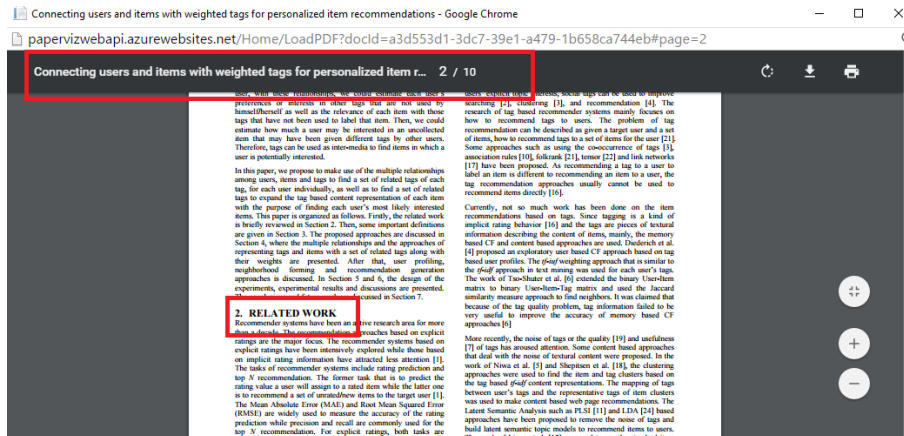


Figure 3.33: Scenario 1: identified document section

In addition, the researcher could look at the metadata information area of PaperViz to get additional information about this publication (see figure 3.34).

i Metadata Information Area

Title: Connecting users and items with weighted tags for personalized item recommendations

Actions:

Source: Proceedings of the 21st ACM conference on Hypertext and hypermedia - HT '10

Type: doc

Year: 2010

Authors: Huizhi Liang Yuefeng Li Yue Xu Xiaohui Tao Richi Nayak

Keywords: [personalization](#) [recommender systems](#) [tags](#) [web 2](#)

Abstract: Tags are an important information source in Web 2.0. They can be used to describe users' topic preferences as well as the content of items to make personalized recommendations. However, since tags are arbitrary words given by users, they contain a lot of noise such as tag synonyms, semantic ambiguities and personal tags. Such noise brings difficulties to improve the accuracy of item recommendations. To eliminate the noise of tags, in this paper we propose to use the multiple relationships among users, items and tags to find the semantic meaning of each tag for each user individually. With the proposed approach, the relevant tags of each item and the tag preferences of each user are determined. In addition, the user and item-based collaborative filtering combined with the content filtering approach are explored. The effectiveness of the proposed approaches is demonstrated in the experiments conducted on real world datasets collected from Amazon.com and citeULike website.

Figure 3.34: Scenario 1: metadata information area

For instance the author defined keywords are usually a good indication of how the

search query can be optimized. Thus, the key phrases ‘personalization’ or ‘recommender systems’ could be good candidates to be added to the visualization area. Another option of identifying additional key phrases is by looking at the tag cloud visualization of the document.

To summarize, it can be said that PaperViz enabled the researcher to quickly spot a section of a document within a collection that is without doubt worth reading. Of course the researcher should also inspect other documents, which are “marked” as potentially useful by PaperViz, in order to get a better understanding about his research field. Moreover, the metadata information area proved to be valuable as it provided the researcher with an abstract of the paper and suitable keywords that could be used for further investigations.

3.3.5.2 Scenario 2

Suppose a researcher needs information on the relation of the three topics ‘social’, ‘similarity’ and ‘learning’. He has already identified the collection that best suits these keywords and is now confronted with the following visualization (see figure 3.35).

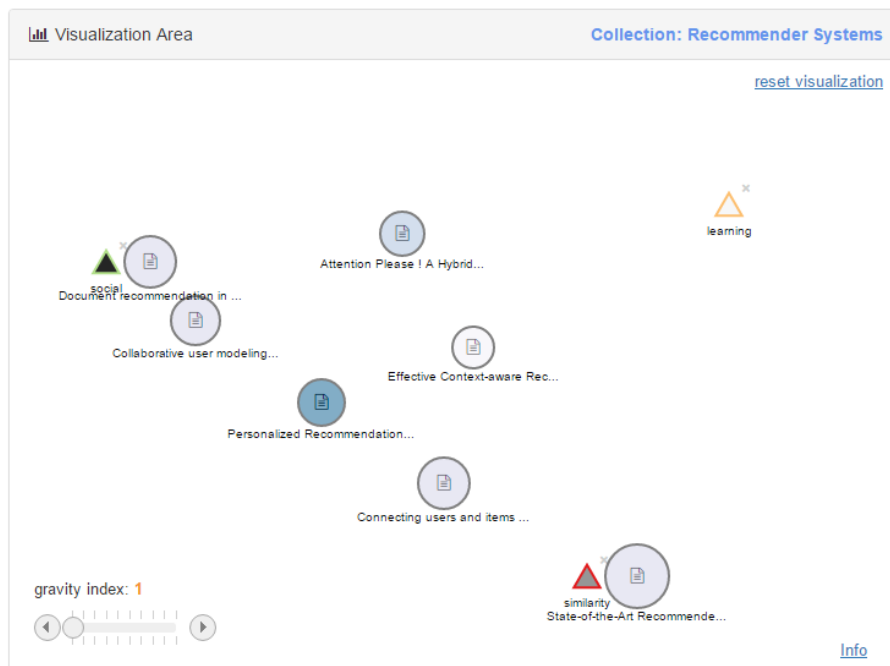


Figure 3.35: Senario 2: PaperViz visualization

The researcher can derive following information from this illustration:

- The key phrase ‘social’ is the most important one, regarding to the visualized documents, followed by ‘similarity’ and ‘learning’, due to its intensity.
- Most of the documents relate to the words ‘social’ and ‘similarity’, as their corresponding circles are placed on a virtual line between these key phrases.
- The document reflecting his search query the most is ‘Personalized Recommendation...’ as it has the highest intensity assigned.
- The document ‘Attention Please! A Hybrid...’ is influenced almost equally by the key words ‘social’ and ‘learning’ as it falls on a line between them.
- ‘Effective Context-aware Rec...’ is the only document that is placed between all three key words. Thus, it is influenced by all key phrases equally. But as the intensity of the circle is very low, this document has only a weak to no relationship to the three key phrases. Inspecting this document using the bar chart visualization indicates that it has definitely no relationship to either of the key phrases.

So this PaperViz visualization informed the user that his collection does not contain any documents that relate on all three topics he is interested in. Thus, it might be advisable to add additional documents to this collection. As the key phrase ‘learning’ only relates to one document of his collection, the researcher should focus on retrieving documents containing this key phrase.

3.3.5.3 Scenario 3

It is assumed that a researcher is interested in documents about clusters. In addition, these documents should not, or only weakly, relate to the topics ‘algorithms’, ‘trees’ and ‘links’. In order to formulate such a query with PaperViz, the researcher could just place the key phrase ‘clusters’ on one side of the visualization and the other three key phrases on the other side. A PaperViz visualization of this query is illustrated in figure 3.36.

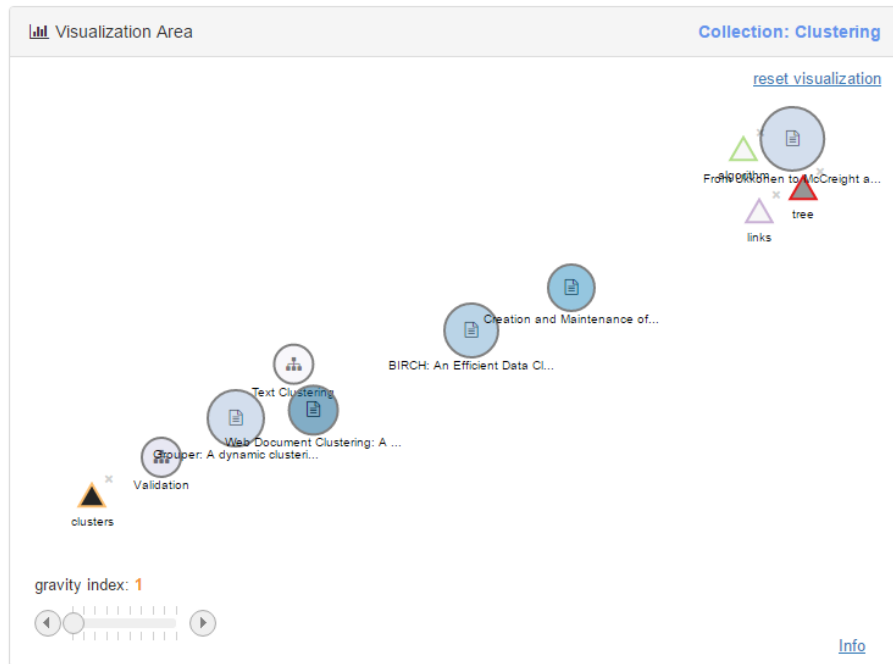


Figure 3.36: Scenario 3: PaperViz visualization

Thus, documents and sub collections that are placed on the bottom left corner of the visualization are worth inspecting, whereas those which are placed at top right corner are irrelevant regarding to the researcher's interests. In order to gain a better separation, the researcher could utilize the gravity slider as illustrated in figure 3.37.

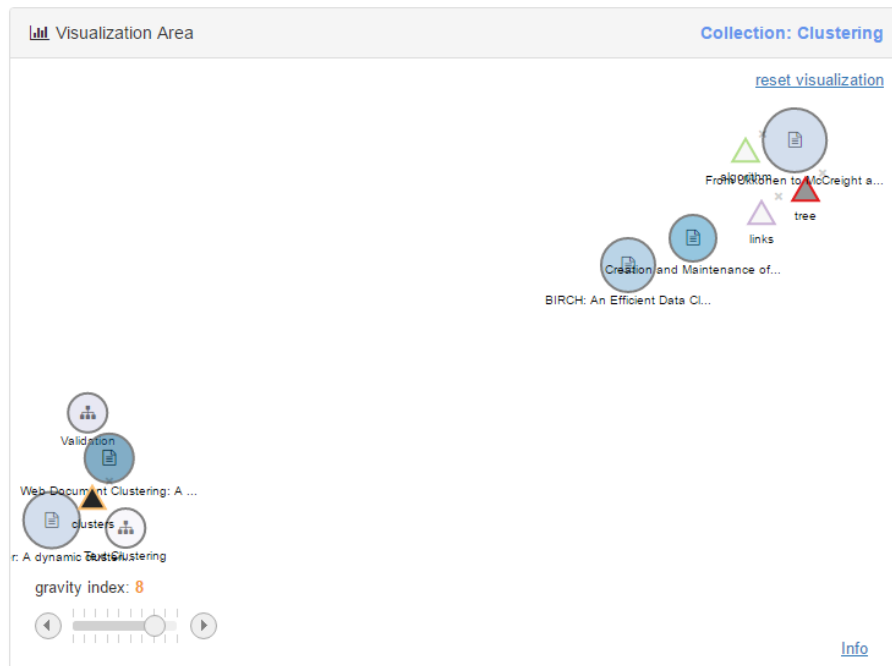


Figure 3.37: Scenario 3: PaperViz visualization (2)

So PaperViz can also be used to formulate “negative” search queries. Defining such a query using traditional search engines is almost impossible.

3.3.5.4 Conclusion

Within this section, three examples are provided of how the visualization of PaperViz could be used to examine a user’s document collection. The purpose of the first scenario is to illustrate the value of the color coding. The second one focuses on the positioning aspect and the third one introduces a quite unusual way of formulating a query. The PaperViz visualizations of these scenarios are fairly simple to understand and therefore, the bar chart visualization and the connecting lines are not necessarily needed to retrieve the required knowledge.

3.4 Workflow

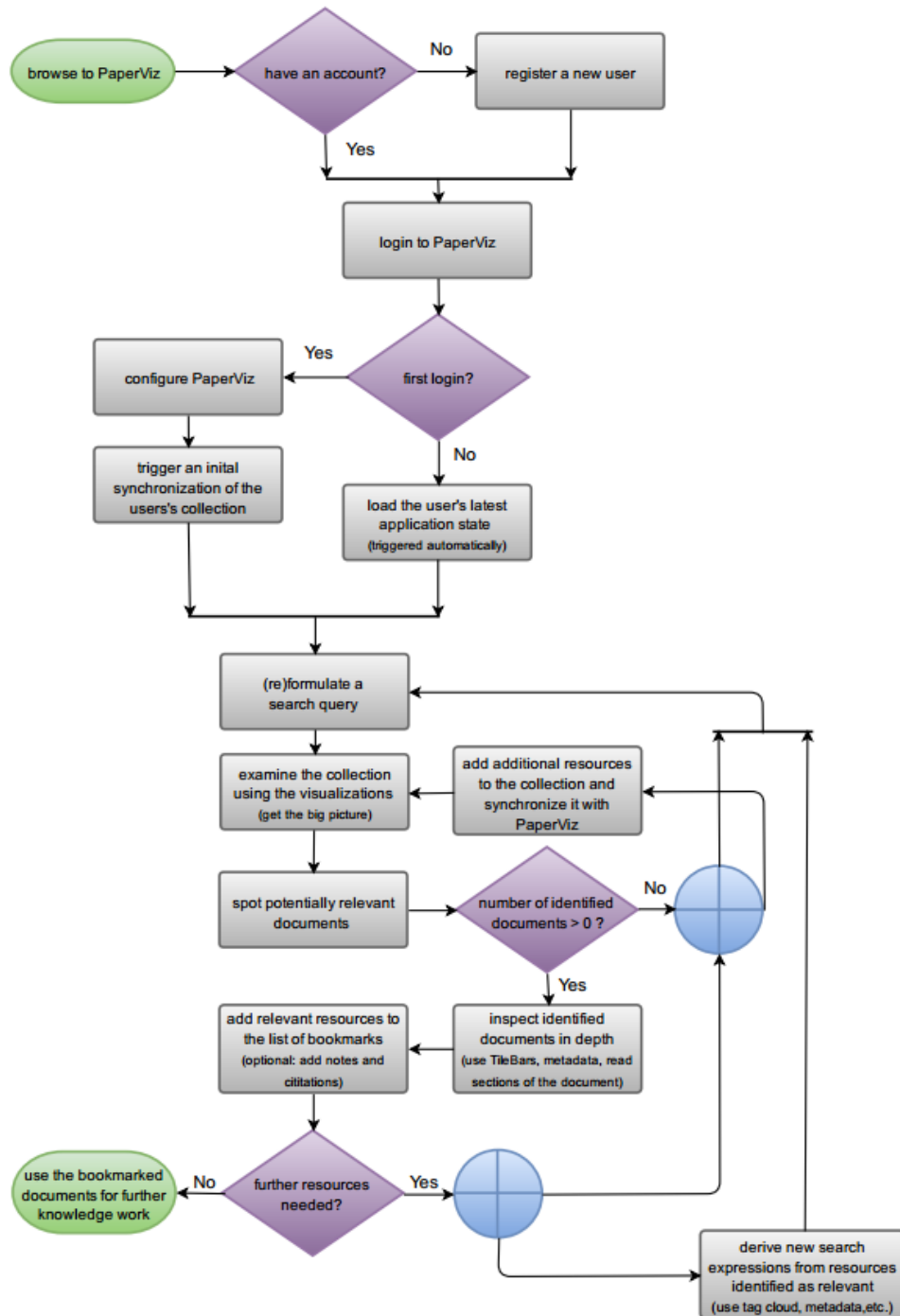


Figure 3.38: Workflow

The typical workflow of finding and organizing relevant resources using PaperViz is illustrated in figure 3.38.

3.5 Summary

This chapter describes the graphical user interface and the main functions of PaperViz from a user perspective. It is designed to fulfill all requirements outlined in section 3.1. One key aspect of the system is that components were carefully designed to comply with MCV principles and all of them are fit into one single web page. For instance when clicking on a circle within the visualization area not only the visualization but also the tag cloud, the tree view, the breadcrumbs and the metadata information area reflect on this event. This enables the user to get a lot of information from different perspectives about documents and collections with just one single click. In section 3.3 the graph-based visualization of PaperViz is described in detail. It provides several additional features allowing the user to quickly examine collections, documents and single pages. Furthermore, it contains numerous animations and transitions which, unfortunately, cannot be illustrated within this thesis in a meaningful way.⁴ However, a lot of effort has been made to provide smooth animations that enhance the expressiveness of the visualization and give valuable feedback to the user. In order to prove that PaperViz fulfilled all defined requirements the following table illustrates a confrontation of the requirements and the implemented features.

Requirements	Features, components or aspects
Provide an intuitive way to quickly browse large document collections	Researchers can use the tree view control, the breadcrumbs and the graph-based visualization to browse their collections.
Provide high quality information at different levels of details (entire collection, sub collections, documents, single pages)	The tag cloud and the graph-based visualization provide information about sub collections and documents. The TileBar visualizes the frequency of key phrases within single pages of a document.
Provide a user-driven visualization model that can be refined and developed based on the researchers interests	The information space of the visualization is based on a search query defined by the user. Users can always reformulate and refine this query.

⁴watch the tutorial video <https://www.youtube.com/watch?v=td9ENIIIZGI&feature=youtu.be> for a demonstration

Assist the user in formulating a suitable search query	The tag cloud visualizes potentially relevant key phrases of documents and collections that can be used to formulate a query.
Support researchers in organizing their document collections	The list of bookmarks and the text editor can be used to annotate and retrieve documents.
PaperViz should be implemented as web-based system	The front end of PaperViz is a modern highly responsive web application. It provides saving and loading functions. The researchers work is persisted in an online database
Provide a graphical user interface (GUI) that is intuitive and user friendly	The GUI was carefully designed to comply with MCV principles.

Table 3.1: Confrontation of requirements and features

Chapter 4

Implementation

Contents

4.1	Information flow	67
4.2	Backend processing service	68
4.3	Web front end	86

As briefly explained in the introduction, PaperViz consists of two main components. The web front end, whose design was outlined in the previous chapter, and the backend processing service. One purpose of this service is to integrate the interfaces to two external systems, Mendeley and Sensium. Mendeley is used as the document collection source system for PaperViz, whereas Sensium is employed for extracting and scoring key phrases of text blocks. The following chapter introduces the information flow between these systems and PaperViz. Moreover the architecture and the implementation of the two main PaperViz components are described.

4.1 Information flow

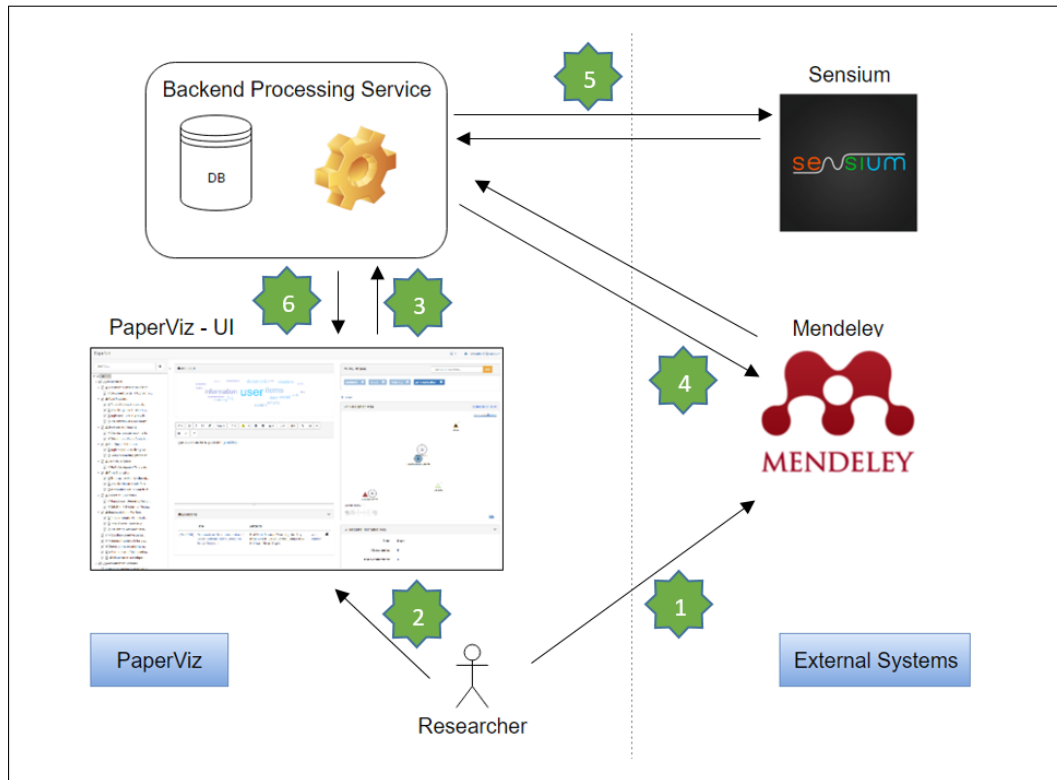


Figure 4.1: The information flow between PaperViz and external systems

Figure 4.1 illustrates the general information flow between the external systems and the components of PaperViz.

1. First of all, researchers have to create their document collection. This is done via Mendeley by creating an appropriate folder structure and adding potentially relevant documents to these folders. Mendeley therefore provides both, a desktop client and a web client. Users can add documents to their collection by searching Mendeley's public catalog or by simply adding documents from a file system.
2. Afterwards, the researcher logs in into PaperViz.
3. The web application then requests the data to be visualized from the backend service.
4. Thus, the backend service first fetches the researcher's collection from Mendeley.
5. Next, the service employs Sensium to extract and score key phrases of this collection and its containing documents.

6. The data retrieved from Mendeley and Sensium is then processed by the backend service and stored into a database. Finally this processed data is sent to the web front end where it gets visualized for detailed analysis.

Downloading documents from external resources and applying text mining algorithms to them is typically a very time consuming task. Therefore, PaperViz caches the information gathered from Mendeley and Sensium in a database for a later faster retrieval. In order to react to changes made to a collection, PaperViz provides functions to detect and process these changes.

4.2 Backend processing service

The backend processing service acts as an intermediary between the web front end of PaperViz and the external systems and serves following main purposes:

- **Authentication server and user management**

As already mentioned, researchers have to register a user to get access to PaperViz. The users and the login credential flow are managed by the service.

- **Connection to Mendeley**

The service consumes the web services of Mendeley to fetch the user's document collection.

- **Text extraction from PDF-files**

Publications and research papers are typically stored in PDF-format. In order to apply text mining algorithms, it is necessary to extract the plain text of such documents first.

- **Connection to Sensium**

Sensium's web API is consumed by the backend service to extract and score key phrases of text blocks extracted from PDF-files.

- **Data preparation**

The main purpose of the backend processing service is to process the data obtained from the external systems. A set of functions are applied to this data in order to transform it into a proper structure for further analysis.

- **Data storage**

As already mentioned, the backend processing service is connected to a database for caching the information retrieved from Mendeley and Sensium. Moreover, it allows users to store and load their work.

- **Connection to the PaperViz front end**

The backend service exposes a web API that is consumed by the PaperViz front end.

In the following the most important frameworks and libraries, which were used for implementing the backend service, are explained briefly. Moreover, an insight of how the external systems and the front end are connected to the backend service is provided.

4.2.1 Frameworks and libraries

The back end processing service is developed in .NET with C# and hosted inside of an Internet Information Services (IIS) application. Thus, all frameworks and libraries used are built on top of the .NET framework.

4.2.1.1 Web API 2

The backend processing service is based on Microsoft's Web API 2 framework. This framework allows the implementation of RESTful services to expose data in different formats to a large variety of clients including browsers, mobile phones or tablets. REST stands for Representational State Transfer and could be defined as the software architectural style of the World Wide Web. RESTful services typically use the Hypertext Transfer Protocol (HTTP) in combination with the HTTP verbs (GET, POST, PUT, DELETE, etc.) to provide access to resources [10]. A resource refers to an object as in object orientated programming (OOP) and is exposed with an API endpoint using the Uniform Resource Identifiers (URI). By calling this endpoint in combination with an HTTP verb, clients are able to fetch or manipulate the resource. For instance, to load the user settings of the currently logged in user of PaperViz, the following request has to be issued:

```
GET /api/AccountSettings
```

The data is exchanged using the JavaScript Object Notation (JSON) format, which is a lightweight alternative to the Extensible Markup Language (XML) and can be easily interpreted by web application via JavaScript.

A detailed description of the endpoints exposed by the backend service is given in a later section.

4.2.1.2 OAuth 2.0

As only authenticated users should have access to the resources of PaperViz the APIs of the backend service are secured by utilizing the OAuth 2.0 authorization framework. OAuth is a widely adopted, open protocol for users to delegate authorization in a secure manner [29]. The simplified OAuth 2.0 login credential flow used by PaperViz of this authentication standard is illustrated in figure 4.2.

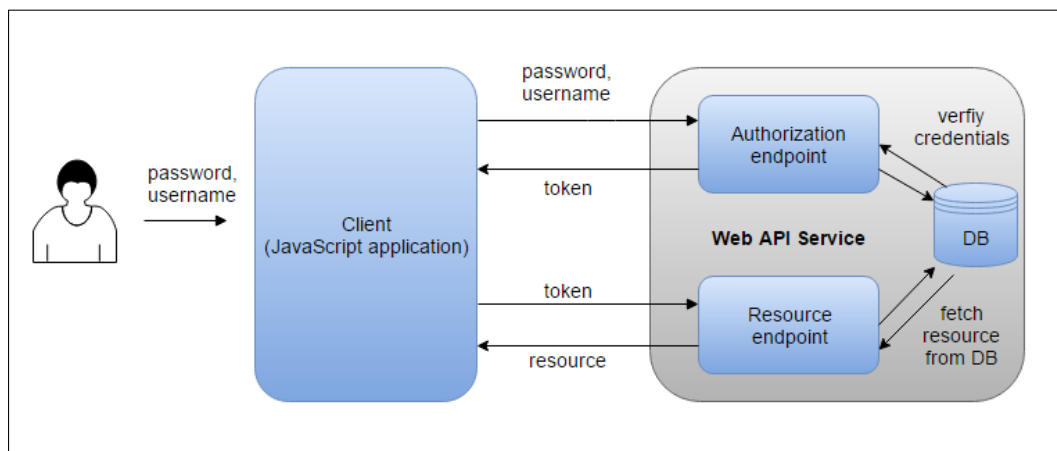


Figure 4.2: OAuth 2.0 login credential flow

So first, the client sends a request containing the username and password to the authorization endpoint. The server verifies the client's credentials and returns an access token. This token needs to be included in the Authorization header of each subsequent HTTP request in order to access protected resources of the API. PaperViz uses the Bearer authentication scheme to transmit these tokens. An example of a client-side AJAX request using this authentication scheme is given in section 4.2.2.3.

4.2.1.3 Entity Framework

The Entity Framework (EF) is used as an object-relational (OR) mapper that allows developers to build data-oriented software applications without the need of writing complex database queries [33]. The EF supports the so called Code First development approach that was also used for implementing PaperViz. This approach enables researchers to define the .NET classes first, which are then automatically mapped to a relational database by the EF. The database structure of PaperViz is outlined later in this chapter.

4.2.1.4 RestSharp

RestSharp is a .NET library that provides functions to call RESTful APIs. It supports features such as automatic XML and JSON parsing, different authentications scenarios such as OAuth and asynchronous requests [9]. PaperViz uses this library to consume the web API of Mendeley and Sensium.

4.2.1.5 iTextSharp

iTextSharp is a .NET port of the Java iText library. It enables developers to create, inspect, adapt and maintain PDF-files. This library is utilized by PaperViz to extract plain text from PDF-files in order to apply text mining algorithms to it.

4.2.2 Web APIs

On the one hand, the backend service exposes web services, which are consumed by the PaperViz front end. On the other hand, it consumes the web services of Mendeley and Sensium. Thus it acts as a client and as a server regarding to web APIs.

4.2.2.1 Connection to Mendeley

As mentioned in 2.2 Mendeley is one of the most popular reference managers. It was released in 2008 and purchased by the Elsevier publishing company in 2013. Beside the reference management component, that enables researchers to read, organize and cite all their research from one library, the platform includes additional components, which were developed to support researchers, students, lecturers and librarians in their daily work. For instance, the social component of Mendeley enables users to join groups dedicated to specific topics where they can start discussions or find additional interesting literature. Users can also follow inspirational researchers in order to get notifications when new papers have been published by them. ‘Mendeley Careers’ is the latest component they introduced. It is comparable to a job management platform for researchers where users can search for science and technology jobs in institutions worldwide. Mendeley also offers a large variety of clients, like a desktop client for Windows, Mac and Linux, a web client and apps for iPhone, iPad and android. However, PaperViz only leverages a very small feature set of this social reference management application as it only acts a document source system for PaperViz. Thus, this work will not examine all features of Mendeley in detail, but focuses on its web services that are consumed by PaperViz. Detailed information about Mendeley can be found at [31].

The standards and frameworks used by the Mendeley web services are quite similar to those of PaperViz’ backend processing service. The authentication is managed by the

OAuth 2.0 framework, resources can be accessed via a RESTful API and the data exchange format is JSON. Mendeley provides three different authorization flows [29]:

1. **Client Credentials**

Tokens created by this option are only capable of accessing the public catalog of Mendeley. Private document collections of users cannot be accessed and therefore the Client Credentials grant type is not a suitable authorization flow for PaperViz.

2. **Implicit**

This grant type creates short-lived access tokens to access the entire Mendeley API including user specific resources. The disadvantage of this option is, that such tokens expire after an hour and the user must be prompted to authenticate again if that happens.

3. **Authorization Code**

This authorization flow issues long-lived tokens and provides functions to refresh the tokens if they expire. As such tokens are allowed to access user specific resources as well, this option is the most suitable grant type for PaperViz.

There are several ways to integrate this Authorization Code grant type flow. One way is that users obtain a token manually via the Mendeley web page. This token can then immediately be used by the client application to access Mendeley's catalog. If it expires, the application utilizes the refresh token for obtaining a new valid token. So after the user provided the access token and some other properties (refresh token, client id, client secret), the authentication could be managed completely by the application without the need of prompting a Mendeley login form. For that reason, this strategy is also used by PaperViz. By clicking the settings button at the top right corner of the PaperViz screen, a modal dialog shows up where users can manage their Mendeley settings (see figure 4.3). Furthermore, this dialog contains a link to the Mendeley web page where users can create their tokens.

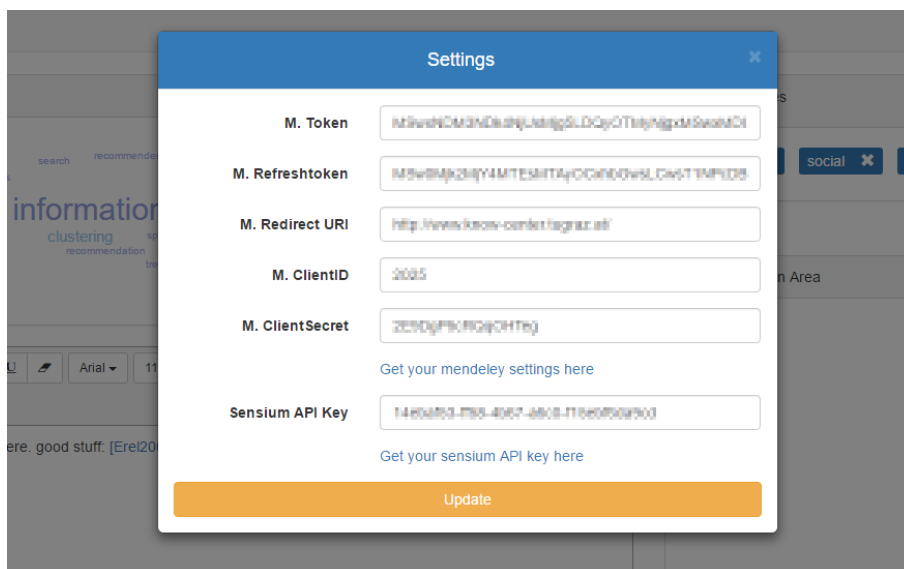


Figure 4.3: The settings dialog of PaperViz

Mendeley offers a large variety of web API methods for authorized clients such as:

- creating, fetching, deleting and updating document collections and their metadata
- creating, uploading, deleting, downloading and searching PDF-files
- documents metadata lookup
- semantic uplifting and document enrichments
- fetching user profiles and public groups
- social networking functions

PaperViz only consumes a small subset of these functions in order to fetch documents from the Mendeley repository (see table 4.1).

Endpoint	HTTP Verb	Purpose
/profiles/me	GET	To get the user profile of the currently logged in user.
/folders	GET	To get the folder structure.
/folders/{folderId}/documents	GET	To get documents contained in a folder. The response does not include a file, but meta-data information about the document.
/files?document_id={documentId}	GET	To get information about the files which are assigned to a document (size, name, fileId, type, filehash, etc.). A document can have multiple files assigned.
/files/{fileId}	GET	To download a file.

Table 4.1: Consumed Mendeley web API functions

4.2.2.2 Connection to Sensium



Figure 4.4: Sensium Workflow

Sensium is a scalable data mining and analysis platform, developed by the Know-Center GmbH. It provides a simple web API enabling developers to integrate the services provided by Sensium into their applications. As illustrated in 4.4 it also provides SDKs for Java and Wordpress. However, as the PaperViz' backend processing service is written in .Net

only the RESTful API is consumed by our system. This API offers a wide variety of text analysis features such as [15]:

- language recognition
- tokenization
- stemming and normalization
- part-of-speech tagging
- sentence segmentation
- named entity recognition
- temporal event extraction
- key phrase extraction
- summarization

To call these methods, clients have to obtain a free API key from Sensium first. Providing this key to the PaperViz system is done by using the settings dialog illustrated in figure 4.3. PaperViz only consumes the key phrase extraction feature of Sensium to extract and score key phrases of text blocks. The corresponding POST request looks as follows:

```
POST https://api.sensium.io/v1/extract HTTP/1.1
Accept-Encoding: gzip,deflate
Accept: application/json
User-Agent: RestSharp/105.2.3.0
Content-Type: application/json
Host: api.sensium.io
Content-Length: 136
Connection: Keep-Alive

{"apiKey":"{the API key}","text":"{the text}":["Summary"]}
```

By specifying the ‘Summary’ extractor, the response will contain a list of key phrase objects in JSON format containing following properties:

- **Text:** textual representation of the key phrase
- **Score:** relative score of the key phrase, specifying its’ importance relative to the other key phrases
- **Occurrences:** list of occurrence objects, demarking the occurrences of the key phrase in the main text

This information is used to compute the score matrix and the TF-IDF scores of documents and collections, which are then visualized by the PaperViz front end.

The algorithm used by Sensium to compute these measures is called TextSentenceRank algorithm which is an extension of the TextRank algorithm. The TextRank algorithm is a graph-based ranking algorithm, that was designed to either detect key sentences or key phrases within a text corpus. It iteratively computes the relevance of a node (phrase or sentence) within a graph by a voting mechanism, where all predecessor nodes vote for specific node. In order to construct the graph, some other text-mining techniques have to be applied to the text beforehand like tokenization and part-of speech tagging. However, based on the idea, that key phrases occur more often in key sentences the TextSentenceRank algorithm combines the key phrase and key sentence approaches of the TextRank algorithm, resulting in a more accurate computation of key phrase ranks. Further information on these two text mining algorithms can be found in [42].

4.2.2.3 Connection to the PaperViz front end

The backend processing service exposes a list of web API methods that are consumed by the PaperViz front. To access these resources, clients need to be authenticated according to the OAuth 2 standard explained in 4.2.1.2. Like the web APIs of Mendeley and Sensium, PaperViz uses the JSON format for exchanging data objects. The list of the REST endpoints implemented by the backend processing service is illustrated in table 4.2.

Endpoint	HTTP Verb	Purpose
/api/Account/Register	POST	to register a new PaperViz user
/Token	POST	to obtain an access token (to login)
/api/Account/Logout	POST	to logout
/api/AccountSettings	GET	to load the user settings
/api/AccountSettings	POST	to set the user settings
/api/ApplicationStates	GET	to load the user defined key phrases, the list of bookmarks and the content of the text editor previously saved by the user
/api/ApplicationStates	POST	to save the user defined key phrases, the list of bookmarks and the content of the text editor
/api/DocumentCollection	GET	to load the document collection of the user including the score matrix and the TF-IDF scores
/api/UserTracings	GET	to save user interactions for evaluation purposes

Table 4.2: Exposed web API functions of the PaperViz' backend processing service

The PaperViz front end issues AJAX requests to consume these endpoints. Requests to secured resources have to contain the bearer access token in the HTTP header. The following code snippet illustrates the typical structure of such a client-side AJAX request:

```
1 var requestData = function (postParams, successFunction) {
2   var uri = baseUrl + "/api/resource";
3   $.ajax({
4     type: "POST",
5     data: postParams
6     url : uri,
7     crossDomain: true,
8     beforeSend: function (xhr, settings) {
9       xhr.setRequestHeader("Authorization", "Bearer " + accessToken)
10    },
11    success : function (data) {
12      successFunction(data);
13    },
14  });
15 }
```

Listing 4.1: AJAX POST Request

Requesting the document collection object

As listed in table 4.2 the web front end issues a GET request to the `/api/DocumentCollection/` endpoint in order to retrieve the information needed for the visualization. This endpoint supports the parameter ‘SyncType’ that accepts following three values:

1. complete

As already mentioned, the backend processing service caches the user’s document collection as well as the extracted key phrases and scores in a database for a later faster retrieval. By specifying the synchronization type ‘complete’, this data gets discarded. In addition, the complete process of fetching the user’s Mendeley catalog, extracting key phrases, data processing and data storing is triggered.

2. differential

This synchronization type detects differences between the cached version of the user’s document collection and the current version of the user’s Mendeley repository. Thus, only documents and sub collections that were added since the last synchronization are loaded and processed by the backend service. Removed documents and sub collections are discarded. This option is extremely useful, as a complete synchronization could be very time consuming, especially for large document collections.

3. none

By specifying the synchronization type ‘none’ the backend processing service returns only the cached version the user’s document collection.

Users can trigger the request to fetch the document collection from the backend service by clicking the ‘sync’ ribbon at the top left corner of the PaperViz front end and selecting the desired option (see figure 4.5). Moreover a request with the synchronization type ‘none’ is issued automatically after the user logs in.

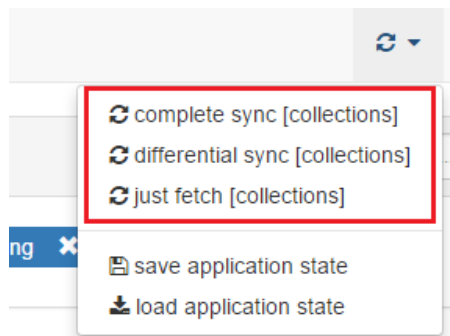


Figure 4.5: Synchronization options of PaperViz

During this synchronization process, a dialog is shown, informing the user about the status of the loading process (see figure 4.6).

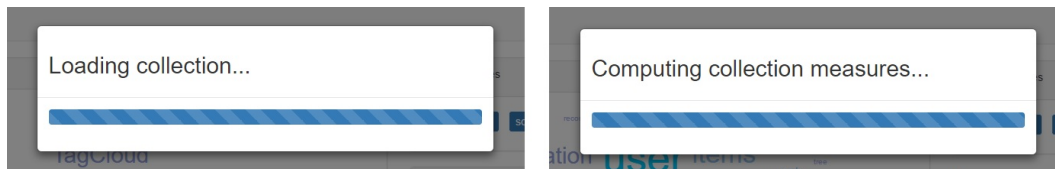


Figure 4.6: Loading animation of PaperViz

The waiting time primarily depends on the size of the user’s collections, respectively the number of changes made to the collection since the last synchronization. The initial loading process could last several minutes, as all documents of the user’s collection have to be downloaded from Mendeley. Furthermore, the backend processing service has to extract the plain text of this documents before employing Sensium to extract and score key phrases. Figure 4.7 illustrates the workflow of this loading process.

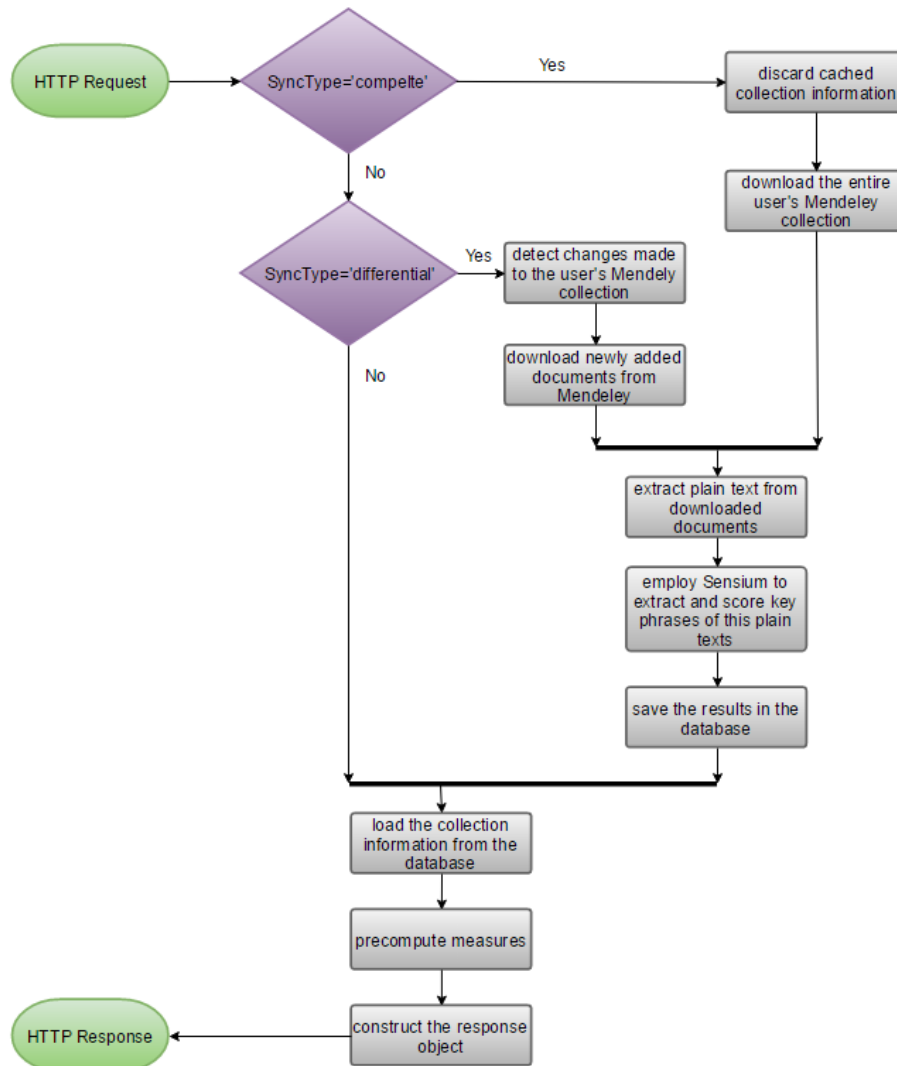


Figure 4.7: Flow diagram of the loading process

After the backend service has processed the request from the front end, it returns a collection objection in JSON format containing all information about the user's collection. The class diagram of the collection object is illustrated in figure 4.8.

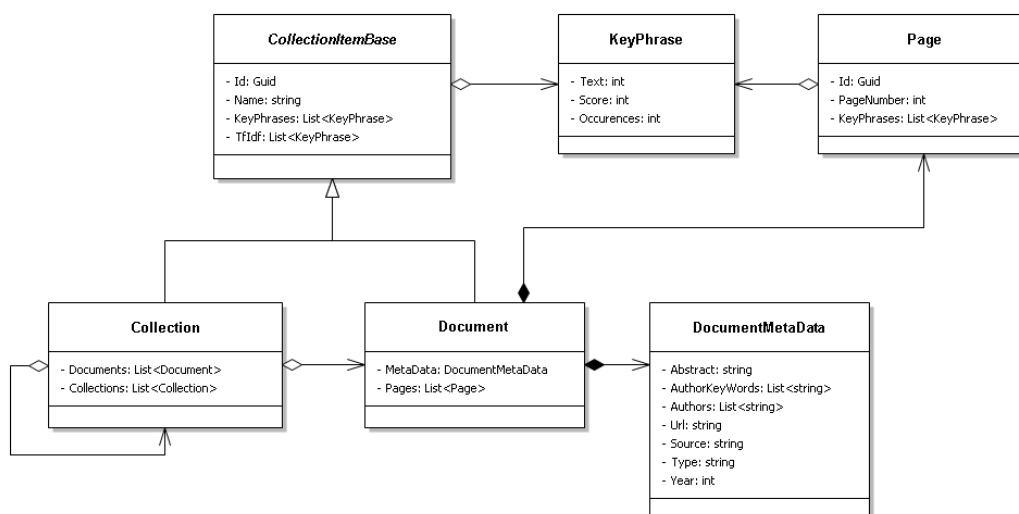


Figure 4.8: Class diagram of the document collection response object

As shown in this figure, the response object contains following information:

- the structure of collection (sub collections and documents)
- key phrase information for sub collections, documents and pages of documents (text, score, occurrences)
- TF-IDF scores for sub collections and documents
- metadata information about documents (name, journal, abstract, authors, etc.)

So this object contains all information about the user's document collection that is required by the PaperViz front end. As a result, no further requests have to be issued by the front end to browse and analyze the collection. The big advantage of this approach is, that the UI can immediately respond to user actions without the need of requesting additional information from the backend service. Another approach would be, to load detailed information about sub collections, documents and pages of documents only on demand. This would result in a faster initial load, but also in a significant decrease of the UI's responsiveness.

Loading and saving the application state

The `/api/ApplicationStates/` endpoint is used to save and load the current status of PaperViz. The corresponding requests can be issued by clicking the 'sync' ribbon and selecting the desired option (see figure 4.9).

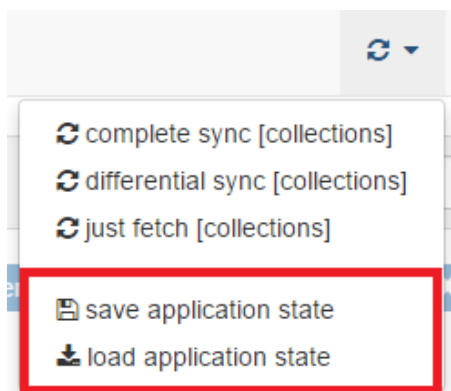


Figure 4.9: Loading and saving the application state

In addition, a request to load the previously saved state is issued automatically after the user logs in into PaperViz. The following information thereby loaded from the database of the backend processing service:

- the list of bookmarked documents
- the content of the text editor
- the key phrases defined in the key phrase box

4.2.3 Data processing and preparation

One requirement of PaperViz is to provide information on different granularity levels of document collections. The most detailed level is thereby a single page of a document. In order to retrieve such information, following steps have to be performed by the backend processing service:

1. download the PDF-files from Mendeley
2. extract the plain text of these files, page by page
3. remove control characters
4. employ Sensium to extract and score key phrases of each page
5. store the results in a database

This page specific information is illustrated by the TileBar visualization of the PaperViz front end. Moreover, it is used by the backend processing service to compute the key phrase scores of documents and collections. This is done by simply calculating the arithmetic average key phrase scores of the single pages for each level of the document collection and normalizing the resulting values on a scale between 0 and 1. In order to prove the correctness of this computation, several tests with documents have been made. The scores returned by Sensium for a document were equal to the aggregated scores computed by the

backend processing service. Thus, there is no need to send the entire text of a document or collection within a single request to Sensium, which accelerates the process of gathering the score matrix significantly. Another advantage of this approach is, that key phrase specific information needs to be stored only for single pages as storing this information for documents and collections would be redundant. In short, Sensium is just used to score key phrases of single pages and all other measures are computed by PaperViz based on these scores. The backend processing service is thereby responsible for calculating the score matrix, the TF-IDF measures and for constructing the collection response object. The front end, on the other hand, computes the query specific properties for the main visualization (see section 3.3).

4.2.4 Data storing

The backend processing service utilizes the EF for loading and saving data to a relational database. As the EF works best with Microsoft based technologies, the Microsoft SQL Server was chosen as database management system. Figure 4.10 illustrates the tables and their relationships of the PaperViz database.

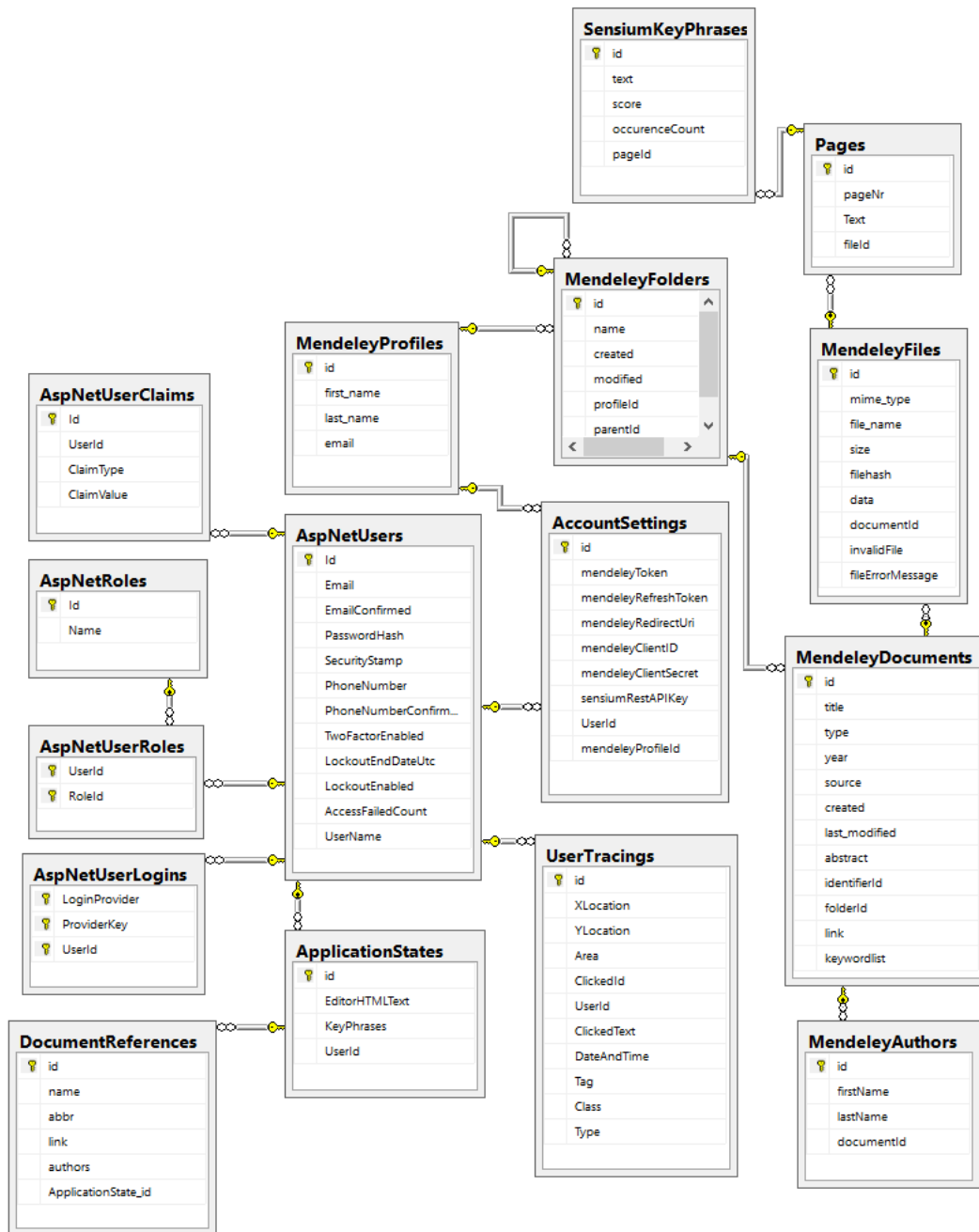


Figure 4.10: Database diagram

The tables beginning with the prefix 'AspNet' are responsible for the user management and are not explained more detailed in this theses. The others are described in table 4.3.

Table name	Content
AccountSettings	Contains the user specific settings to access the web APIs of Sensium and Mendeley.
MendelyProfile	Contains the user profiles of Mendeley.
MendeleyFolders	Represents the collection structure of a user's Mendeley catalog.
MendeleyDocuments	Contains metadata information about the documents contained in a folder.
MendeleyAuthors	Contains the list of authors related to a document.
MendeleyFiles	Contains the files which are assigned to a document of Mendeley. Moreover, additional information such as type, size, name or filehash are stored in this table.
Pages	Contains the page number and the plain text for each page of a file.
SensiumKeyPhrases	Contains the key phrases, their scores and occurrences of every page.
ApplicationStates	Contains the content of the text editor, the defined key phrases and the list of referenced documents of a specific user.
DocumentReferences	Contains information about the referenced documents, which is shown in the 'list of bookmarks' panel.
UserTracings	For evaluation purposes (see chapter 5).

Table 4.3: Description of the database tables

4.3 Web front end

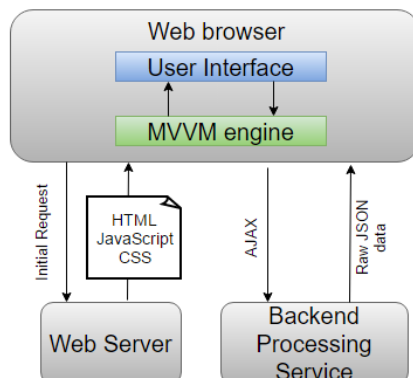


Figure 4.11: Front end architecture

The web front end of PaperViz was implemented as SPA to provide a fluid user experience. It is a pure client-side web page, where the layout and the entire application logic is loaded in the initial page load (see figure 4.11). Furthermore, the collection information, which is visualized by the front end, is fetched from the backend processing service with a single AJAX request. As already mentioned, this request is automatically issued by the front end after the user logs in. Thus, users usually have to wait a couple of seconds until the application and the data is loaded. This waiting time mostly depends on the size of the user's collection and the amount of changes they have made to their Mendeley collection since the last login. Large collections with over 200 documents can result in a waiting time of over one minute. So the initial load of PaperViz is a quite time-consuming process. But this approach enables the web page to respond immediately to user interactions and prevents constant round-trips to the server. In addition users can run the front end locally, using the file URI scheme, as no particular hosting runtime is required. However, the biggest challenge when implementing the front end was to speed up the computation time for the main visualization of PaperViz. As this visualization is based on a user defined query, no precalculation is possible. Thus, all properties of this visualization such as the positions and the intensity of objects have to be computed immediately after the query was reformulated by the user. At an early stage of the implementation the JavaScript engine seemed overtaxed with this task, causing the visualization to appear choppy. Complex search queries in combination with a large document collection even caused the browser to become unresponsive (to freeze). For that reason, two improvements were made to speed up the computation time. First, the collection object returned by the backend service is transformed into a structure that allows a faster computation of

the visualization properties. Second, a HTML5 web worker is employed to perform this computation. A web worker is a JavaScript script that is executed on a dedicated thread in the background, independently of other user-interface scripts. By combining these two aspects, the computation time for the visualization decreased dramatically, resulting in a highly responsive visualization. Due to the fact, that the PaperViz front end is a SPA and, thus, contains the entire application logic in form of JavaScript code, another challenge was to organize this code in a meaningful way. For that reason, the web application utilizes a client-side Model-View-ViewModel (MVVM) framework in combination with the module design pattern, which are described later in this section. Interactions with the backend processing service component are implemented with AJAX calls. Thus, blocking of the UI is avoided when loading huge data sets such as document collections. The API methods consumed by the front end were already outlined in section 4.2.2.3.

4.3.1 Frameworks, libraries and design patterns

The application logic of PaperViz was implemented with JavaScript and references a number of JavaScript libraries and frameworks. The most important ones and their purpose regarding to PaperViz are explained briefly in the following.

4.3.1.1 jQuery

jQuery is a feature-rich JavaScript library that aims to simplify the client-side scripting of HTML. 65% of the top 10 million highest-trafficked web applications use of this library and many other JavaScript libraries are based on jQuery [22]. It makes things like HTML document traversal and manipulation, animation, event handling and AJAX integration much simpler. Moreover, it is open-source software and therefore free to use.

4.3.1.2 Bootstrap

Bootstrap is a widely used web front end framework, which is designed to ease the development of dynamic websites. Beside a number of jQuery plugins it contains HTML- and CSS-based design templates for buttons, tables, navigation and other interface components [5]. In addition a lot of Bootstrap themes and templates are available on the web that can be used as the basic layout for a web application. For instance, PaperViz is based on the ‘SB Admin 2’ template, which is free to download and contains a variety of powerful jQuery plugins (see figure 4.12).



Figure 4.12: Bootstrap template 'SB Admin 2'

4.3.1.3 Knockout

Knockout is a client-side JavaScript MVVM-framework that helps separating the concerns of domain data, view components and UI logic. In addition, it allows automatic synchronization between the view component (HTML) and the domain data (model) through a 2-way data binding by leveraging the native event management features of the JavaScript engine. The MVVM components regarding to Knockout can be defined as follows [26]:

- **Model (data):** the raw JSON data coming from the backend service
- **View (HTML and CSS):** the user interface
- **ViewModel (JavaScript):** responsible for the UI logic and acts as an intermediary between the view and the model

This approach is quite powerful when dealing with forms, tables or data grids. It allows developers to quickly create web based CRUD operations including validation and computed properties. But on the other hand several UI controls, such as the tree view control or the text editor used by PaperViz, cannot be combined with this framework in a meaningful way. Moreover, Knockout does not come with a built in way to modularize the code of the web application. It is possible to put the entire code of the application logic into a single file. Thus, for providing a solid system architecture, PaperViz utilizes the JavaScript Module Pattern.

4.3.1.4 JavaScript Module Pattern

The Module Pattern is a commonly used JavaScript design pattern that assigns modules to components of a web application. It could be defined as simple structural foundation, which aims to keep the code both cleanly separated and organized. This pattern thereby mimics the concept of classes, enabling developers to implement private and public variables and methods. Private methods and variables are shielded from the global scope and can only be accessed from the module itself. This hides complexity and keeps the global namespace of JavaScript clean [38]. Following code snippet illustrates a simple example of a module. It can be compared to a static class of object orientated programming languages.

```
1 var exampleModule = (function () {
2
3   var myPrivateVar, myPrivateMethod;
4   // A private counter variable
5   myPrivateVar = 0;
6   // A private function which logs any arguments
7   myPrivateMethod = function( foo ) {
8     console.log( foo );
9   };
10
11  return {
12    // A public variable
13    myPublicVar: "foo",
14    // A public function utilizing privates
15    myPublicFunction: function( bar ) {
16      // Increment our private counter
17      myPrivateVar++;
18      // Call our private method using bar
19      myPrivateMethod( bar );
20    }
21  };
22 })();
23
24 // Usage:
25 // Call the public function
26 exampleModule.myPublicFunction("test");
27 // Get the public variable
28 var foo = exampleModule.myPublicVar;
```

Listing 4.2: JavaScript Module Pattern

PaperViz implements such modules for almost every front end component. For instance the AJAX requests that consume the RESTful API of the backend service are packed

within a module. Other examples are the tree view control, the text editor or the graph-based visualization. The code of each module thereby resides in a dedicated file named after the module. As this pattern enables developers to only expose a public API to other components of the application, it is quite easy to exchange specific modules. Moreover the components of PaperViz are developed in a way that they can easily be integrated to other web applications.

4.3.1.5 D3.js

D3.js (Data-Driven Documents) is a library designed to create rich and interactive visualization in web browsers using HTML, SVG and CSS. It is used to bind arbitrary data to SVG objects which are then added to the Document Object Model (DOM). Moreover it provides a large set of functions to style and manipulate these objects. Implementing transitions and animations to visualize data changes can be accomplished very easily. Although the learning curve of D3.js is quite steep, it is a very powerful library to visualize large data sets in web browsers. Hence, this library was used to implement the graph-based visualization of PaperViz.

4.3.2 Summary

This chapter describes the technical implementation of the two main components of the PaperViz system. The backend processing service is based on Microsoft's Web API 2 framework and acts as an intermediary between the PaperViz front end and the external systems Mendeley and Sensium. Beside consuming and exposing RESTful web services, it is in charge of data processing, data caching and user management. The data processed by the backend processing service is visualized by the PaperViz web front end, which is implemented as SPA in order to provide a smooth user experience. By combining several well-known client side libraries, frameworks and pattern it offers PaperViz users a large variety of functions and features allowing them to quickly examine document collections. The core component of the web front end is the graph-based visualization, which was implemented using D3.js.

Chapter 5

Evaluation

Contents

5.1	Methodology	91
5.2	Participants	96
5.3	Procedure and Conditions	96
5.4	Results	97
5.5	Conclusion	102

As previously mentioned, PaperViz is the first system that combines different visualization techniques and other features to support researchers in writing scientific papers. In order to ascertain, whether these features are well implemented and easy to use, an evaluation was carried out. The goal of this evaluation was to measure the usability and usefulness of the system. Moreover, the feedback of the participants was used to identify functions and features, that are worth implementing or need improvement. Another interesting aspect of this evaluation was to recognize how the tool was used by the participants. For instance, which of the three possible ways of navigating within a collection is primarily used or how the key phrases are placed within the visualization area. As the target audience of PaperViz consists of expert users and non-expert users, the PaperViz web front end was evaluated with a formal user study involving both of these user groups.

5.1 Methodology

In this study, participants had to accomplish five tasks with PaperViz. Each task had a different scope, but all of them were based on the same collection, which was created prior to the evaluation in Mendeley. After finishing a task, the participants had to answer several questions assessing the workload and the difficulty level of the task. After working off all five tasks, they were asked to do three questionnaires to provide feedback about

the features, the usability and the personal opinion of PaperViz. In addition, every action performed by the evaluators was recorded and saved within the database of the backend processing service in order to retrieve additional knowledge about how the system is used. The whole evaluation process was designed in a way, that it could be accomplished unattended. But it was soon realized, that especially non expert-users needed some support in order to quickly get used to the system and to understand the different visualization features. Thus, a project member supervised the evaluation and helped the participants in case they had any problems in accomplishing a task. Moreover, the supervisor analyzed how the evaluators used PaperViz and took some additional notes regarding potential relevant observations.

5.1.1 Evaluation preparation

In order to facilitate the evaluation process, several tasks had been carried out prior to the evaluation, which are listed in the following:

- **Create a sample collection in Mendeley**

To keep the study in a manageable timeframe, a hierarchically structured sample collection was created for the evaluation in Mendeley. Thus, participants didn't have to create their own collection, which also ensured that everyone was confronted with the same baseline condition. The sample collection consists of three main collections 'Visualization', 'Clustering' and 'Recommender Systems'. Each of these collections contains real publications structured in a certain number of sub collections. All in all, the repository consists of 51 documents and 14 sub collections.

- **Create evaluation users for PaperViz**

Registering a PaperViz user and configuring the system was not part of the evaluation process. But to allow more reasonable analysis of the tracked user actions, it was necessary, that every participant used a dedicated PaperViz user. Thus, 30 evaluation users were created and all of them were linked to the sample collection mentioned before. Moreover, a web page was created, allowing participants to pick one of these 30 evaluation users.

- **Create an evaluation version of PaperViz**

Some features and functions of PaperViz were not part of the evaluation process and therefore, an evaluation version of the system was created. Features like synchronizing the document collection or the configuration panel were disabled. In addition, a JavaScript code was implemented tracking all user interactions and sends the in-

formation to the backend processing service where it gets stored in the database for later detailed analysis.

- **Create a video explaining the components and features of PaperViz**

In order to quickly introduce PaperViz to the evaluators, a video was created, showing all features of the system that can be used during the evaluation. This video was uploaded to youtube and has a length of 4 minutes and 34 seconds.

- **Create the guideline and the questionnaire**

A google form was created that guides the participant through the evaluation process. This form contains all the information (introduction, web links, tasks, questionnaires) that is necessary to conduct the evaluation.

5.1.2 Tasks

As already mentioned, each participant had to perform five tasks with different scopes and an increasing level of difficulty. Each task contains:

1. an introduction, explaining the goal of the task
2. a step by step guide describing the steps, which have to be performed to fulfill the task
3. questions that should be answered by using the features of the PaperViz system
4. some useful hints

5.1.2.1 Training Task

The main goal of the training task was to make the participants familiar with PaperViz. They were guided through a typical use scenario, where they had to use every function and feature of the system, which is needed to accomplish the subsequent tasks.

5.1.2.2 Task 1

In Task 1 participants had to interpret the intensity coding of the visualization in order to identify the relevance of collections and documents. In addition, they had to determine relationship strengths between key phrases and collections using the bar chart and the connecting lines. Thus, the goal of this task was to evaluate if the relevance and relationship strengths are visualized in an understandable way.

5.1.2.3 Task 2

Task 2 focused on the positioning mechanism of the graph-based visualization. Thus, the participants had to interpret the positions of circles within the visualization area in order to identify potentially relevant and irrelevant documents. In addition, this task contains some sub tasks where the navigation and filtering functions had to be used.

5.1.2.4 Task 3

Task 3 focused on the TileBar- and the tag cloud visualization of PaperViz. In order to accomplish this task, participants had to identify a certain page within a document that reflects a certain keyword the most. Moreover, they had to refine their search query by identifying an additional key phrase using the tag cloud visualization or the metadata information area.

5.1.2.5 Task 4

In task 4 participants had to formulate a “negative” search query. Thus, the goal was to identify documents and collections that contain the keyword ‘clusters’ but are not, or only weakly related to the keyword ‘tree’. In order to visually separate potentially relevant from non-relevant documents, they were asked to use the gravity slider to accomplish this task. In addition, they had to bookmark and annotate a document that had been identified as relevant.

After accomplishing a taskm the participants had to answer the following three questions (multiple choice, 7-point-likert-scale):

1. **Performance:** How successful were you in accomplishing what you were asked to do (perfect, failure)?
2. **Effort:** How hard did you have to work to accomplish your level of performance (very low, very high)?
3. **Frustration:** Overall, this task was (very easy, very difficult).

5.1.3 Questionnaires

After completing all tasks, the participants were asked to provide feedback on PaperViz in the form of three questionnaires (see appendix A).

5.1.3.1 Questionnaire 1

The first questionnaire consisted of 16 7-point-likert scale questions that examined the different functions and features of PaperViz. The goal of this questionnaire was to identify components of the system that need to be improved.

5.1.3.2 Questionnaire 2

The second questionnaire consisted of 11 7-point-likert scale questions and aimed to measure the usability of PaperViz based on the System Usability Scale (SUS). It contained questions like if the participants would use PaperViz frequently or if they felt confident in using the system.

5.1.3.3 Questionnaire 3

In contrast to the other two questionnaires, the third one contained some open questions asking the evaluators about their personal opinion about PaperViz. Thus, they could point out some features that would be worth implementing in order to increase the usability of the system.

5.1.4 Data sets

During the evaluation process, following data was collected:

- The answers to the task based questions.
- The answers regarding the three questionnaires.
- The performed user actions to accomplish the tasks. Thereby following information was recorded:

current user ID

used component (e.g. tag cloud, tree view, graph-based visualization, etc.)

X and Y location of the mouse event

ID of the clicked, hovered or dragged element

text of the clicked, hovered or dragged element

parent HTML element of the clicked, hovered or dragged element

CSS class name of the clicked, hovered or dragged element

date and time of the action

type of the action (click, drag, drop, hover)

- Notes taken by the supervisor regarding potential relevant observations and a personal interview at the end of the evaluation consisting of three questions:

What were the main challenges while accomplishing the tasks?

Do you have any ideas on how to improve the usability of PaperViz?

Are there any features, functions or components that you are missing?

5.2 Participants

5.3 Procedure and Conditions

As already mentioned, the evaluation process was supervised by a project member. Therefore, it was first of all necessary to schedule an appointment with the participants. Exactly 50% of these appointments were done personally, the other half was conducted via web conference. Thus, the supervisor could always observe how the participants were using PaperViz. 12 out of 16 participants used their own devices for the evaluation and they were able to choose between Mozilla Firefox (19%) and Google Chrome as web browser (81%). The procedure of this evaluation was divided into three phases: Introduction, Training and Feedback.

5.3.1 Introduction

At the beginning of the evaluation process, the supervisor provided the participants with a short overview on what awaits them within the next 30 to 60 minutes. Moreover he explained the purpose of PaperViz and the idea behind creating this system. Afterwards, the participants opened the google form in a web browser. This form contains three web links. The first link points to the video, which the participants were asked to watch in order to get an idea of the functions and features provided by PaperViz. The second one links to the web page, where the participants had to pick a “free” PaperViz user and mark it as “taken”. By clicking the third link, the participants browsed to the web front end of Paperviz. After logging in with an evaluation user, the session moved on to the training phase.

5.3.2 Training

In order to get familiar with the PaperViz system, the participants needed to accomplish the training task. The steps to accomplish this task were explained very detailed in the google form. In addition the supervisor supported the participants and provided useful information and tips regarding the interpretation of the visualizations and the other

functions of PaperViz. After finishing the training tasks, the participants should have been able to perform the other four tasks autonomously.

5.3.3 Feedback

At the beginning of the feedback phase, the participants had to accomplish the other four tasks. During this phase, the supervisor acted primarily as observer and only intervened in cases where the participants had problems in understanding or completing a task. As already mentioned, the evaluators had to answer the three questions listed in 5.1.2. for each task. After completing these tasks, the participants were asked to answer the three questionnaires listed in 5.1.3. The feedback phase closes with a short interview where the participants were asked for comments about PaperViz.

5.4 Results

The SUS score of PaperViz was calculated based on the responses to questionnaire 2. As table 5.1 illustrates, the system received an aggregated score of 89.38.

scale	μ	σ
SUS	89.38	7.38
SUS Usable	88.02	8.59
SUS Learnable	94.79	5.99

Table 5.1: PaperViz' System Usability Scale scores

In order to interpret this result, [39] published a graph showing how the percentile ranks associate with SUS scores and US letter grades (see figure 5.1).

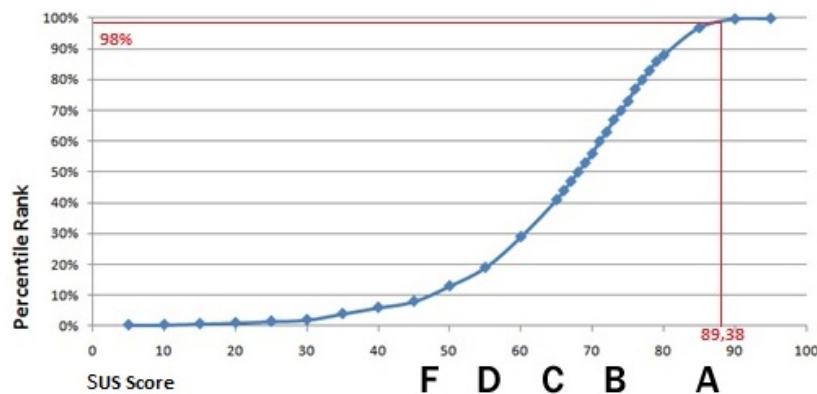


Figure 5.1: System Usability Scale Grading Graph

This graph indicates, that the mean SUS score is 68, which can be interpreted as a grade of a C. The 89.38 of PaperViz converts to a percentile rank of 98%, meaning that it would receive an A and has a higher perceived usability than 98% of all products tested.

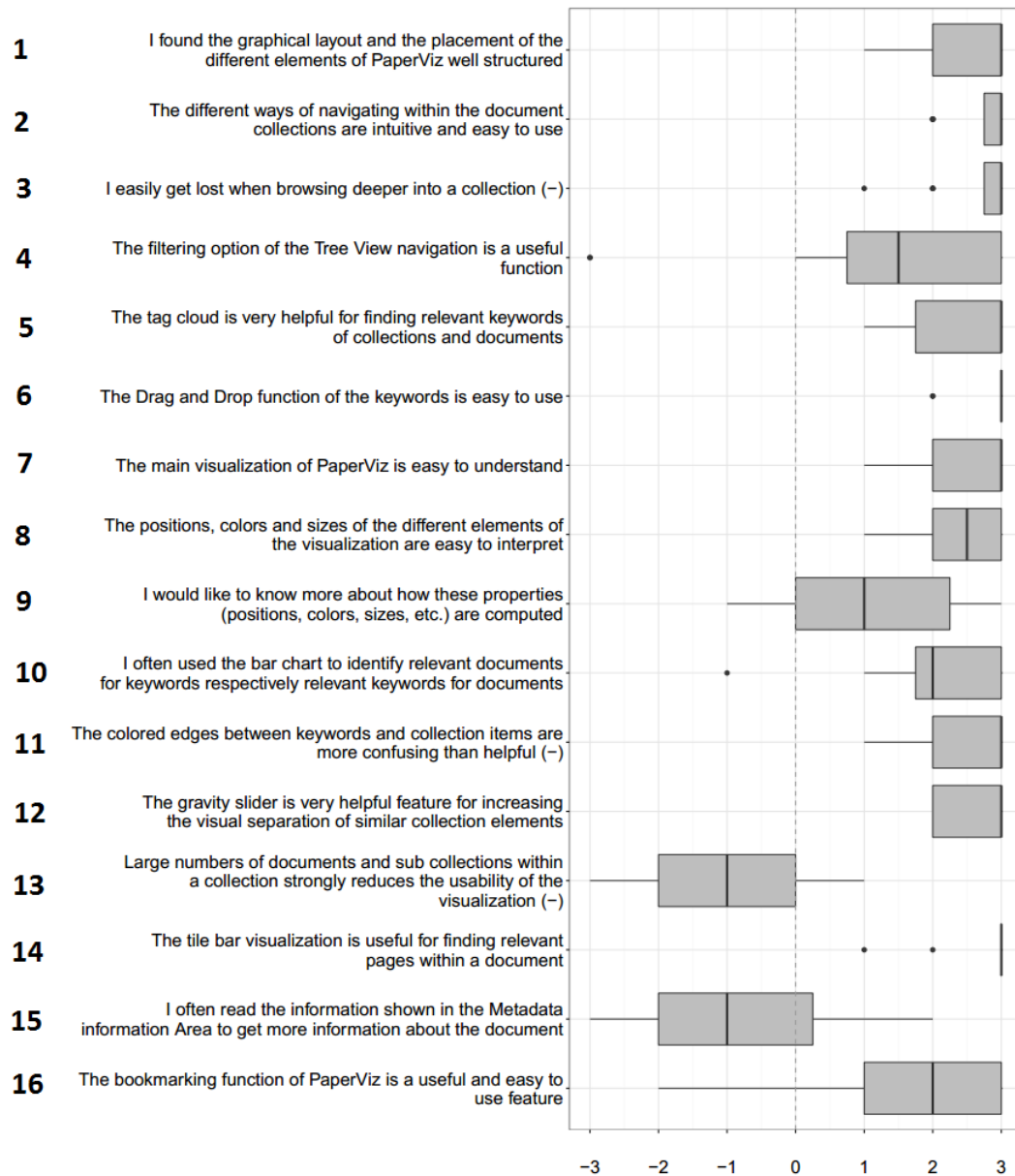


Figure 5.2: Scores of the questions about the functions and features of PaperViz (-3 = strongly disagree, 3 = strongly agree)

Figure 5.2 visualizes the scores of the questions about the functions and features (questionnaire 1). Some questions are inverted, marked with a “(-)” after the question. Based on these boxplots two main weaknesses of the system can be identified:

1. **Graph-based visualization (question 13):** If a collection, which is currently visualized by the graph-based visualization, contains a lot of documents or sub collections, the usability and readability of this visualization is strongly reduced. In addition such a scenario could cause a circle to be placed at a complete different position than originally calculated, leading to a misinterpretation of the graph. This happens if many documents or sub collections have nearly identical relative relationship strengths to the defined keywords. Placing their corresponding circle at the same position is not possible, as overlaps are avoided by visualization.
2. **Metadata Information Area (question 15):** The Metadata Information Area of PaperViz was not recognized as a valuable feature by the participants. This could have several reasons. First and foremost, it was not necessary to read the information shown within this panel to complete the tasks. For instance, in order to find additional keywords related to a collection, most participants preferred to use the tag cloud instead of looking at the author keywords. Another reason is, that this panel is placed at the lower right corner of the screen. According to Peep Laja [27] this position gets the least attention from users. Furthermore, when working with low resolution screens (1366 x 768 pixels and less), users have to scroll down in order to view this area.

In addition, two other features were identified that need improvement:

1. **Filtering (question 4):** Filtering out irrelevant documents and sub collections can only be accomplished by using the tree view component. But in order to identify such literature, the graph-based visualization has to be analyzed. Thus, after identifying documents that a user wants to be removed from the visualization, he is forced to search for this documents within the tree view component. This is not user friendly, as it requires unnecessary effort and time.
2. **Bookmarking (question 16):** In order to link a text within the text editor to a document, users can use the bookmarking and annotation functions of PaperViz. But to accomplish this task, three actions are required. First, the document has to be selected. Next the “add to bookmarks” button has to be clicked and finally the user has to click the “add citation” button within the list of bookmarks.

Despite these weaknesses, the boxplots also illustrate that most of the implemented functions received a positive feedback. As the graph-based visualization is the core component of PaperViz, it should be noted, that the participants had no problems in understanding and interpreting it (question 7 and 8).

The results regarding the workload and the task difficulty are illustrated in figure 5.3. Lower scores thereby indicate a better result. The boxplots for the training task look as expected. The participants were guided through the steps by the supervisor and, therefore, they had no problems in accomplishing the task ('Performance'). However, as they have never worked with PaperViz before, they had to learn how the different functions and features are used and how to interpret the graph-based visualized. Thus, 'Effort' and 'Frustration' received higher scores. Task 1 was the first task they had to accomplish on their own and as the figure indicates, the participants still faced some uncertainties in using PaperViz. The decreasing scores for task 2 and 3 indicate, that participants learned very quickly how to use PaperViz. Almost everyone accomplished task 3 without any problems and they stated that this task did not cost much effort. They had the feeling of being successful and felt very confident using the tool. Because of this reason, it is interesting that task 4 turned out to be the most difficult task. In order to accomplish this task, the participants had to formulate a partly "negative" search query. To be more specific, they had to identify documents that are strongly related to one key phrase, but weakly related to another key phrase. This is a quite uncommon scenario, as classic search engines do not support such search queries. Therefore, some participants had problems in finding the correct strategy to solve this task. Thus, the higher scores are mainly based on the fact, that this kind of search query cannot be formulated using classic search engines.

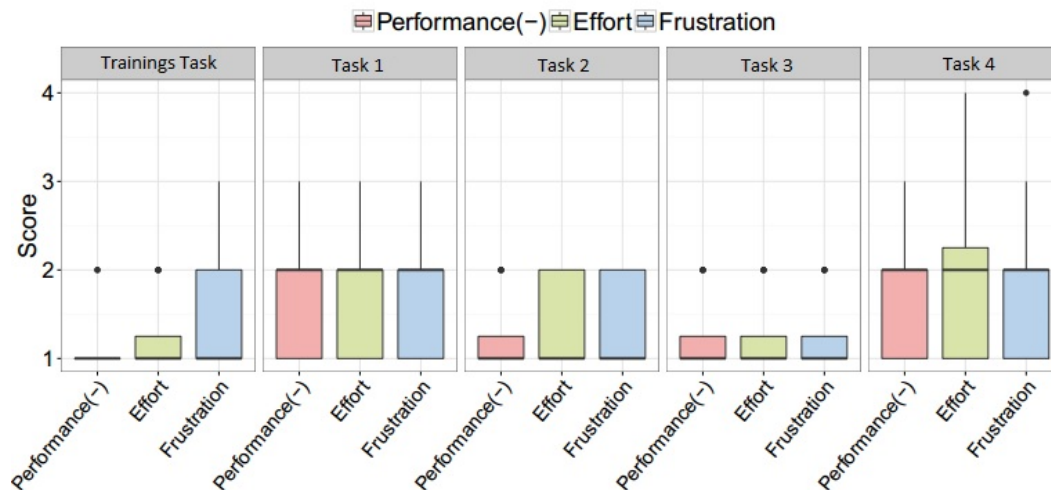


Figure 5.3: The results of the personal feelings about the success in accomplishing a task (performance), the effort to accomplish their level of performance (effort) and the difficulty-level of the task (frustration). (lower scores are better)

Table 5.2 illustrates the log statistics of user activities traced in the course of the evaluation. The figures look as expected and thus, no additional knowledge can be derived from this numbers. The only interesting fact is that only 8 out of 16 participants used the breadcrumbs component for navigating to the parent collection. The others preferred using the tree view component, which usually takes more time and effort. A reason for this could be that they were not aware of the breadcrumbs navigation feature. If this is the case, underlining the elements of the breadcrumb trail to emphasize that these items are hyperlinks, could improve this component.

Type	Action	μ (σ)	Users
control (exploration)	key phrases added (manually)	11.6 (6.4)	16
	key phrase added (tag cloud)	10.3 (4.9)	16
	key phrase added (metadata information area)	1.2 (0.4)	6
	key phrase removed	8.3 (7.6)	11
	navigation (tree view)	11.9 (7.9)	13
	navigation (visualization)	72.6 (64.2)	16
	navigation (breadcrumbs)	2.5 (1.7)	8
	reset	18.1 (11.5)	16
prediction (awareness)	key phrase dropped	16.4 (9.9)	16
	key phrase moved	17.3 (12.5)	16
	key phrase hovered	125.6 (101.9)	16
	document hovered	114 (88.2)	16
	collection hovered	89.8 (66.6)	16
	gravity slider changed	80.1 (53.9)	14
drill-down (explanation)	document clicked	9.7 (4.2)	16
	document opened (TileBars)	5.4 (3.7)	16
	document opened (metadata information area)	1 (0)	4
	document bookmarked	5.8 (4.7)	16
	document cited	2.8 (1.7)	16
	document unbookmarked	1 (0)	1
	text editor used	5.6 (5)	16

Table 5.2: Log statistics of user activities traced in the course of the evaluation

At the end of the evaluation, participants were asked about their personal opinion of PaperViz (questionnaire 3) followed by a short interview with the supervisor. The most interesting findings are listed in the following:

- **Learning by Doing:** At the beginning, the graph-based visualization and their properties were hard to understand. But when working with the component, users get familiar with it very quickly.
- **Layout improvements:** The usability of PaperViz decreases significantly when working with low resolution screens. It requires the user to use the scroll function of the web browser very often. In addition, the visualization area is quite small compared to high resolution screens. Because of these reasons, PaperViz should support a flexible layout, meaning the users can adjust the layout of the components based on their needs to avoid scrolling. Furthermore, a function that undocks the visualization component from the main window would circumvent this problem. The resulting floating window can then be scaled by the user.
- **More drag and drop:** It was observed that many participants tried to use drag and drop for actions like adding a key phrase from the tag cloud to the visualization or adding a document to the list of bookmarks. This is currently not supported by PaperViz, but would increase the usability of the tool tremendously as it saves time and unnecessary clicks.
- **LaTeX integration:** When using the text editor for drafting sections of a paper, it would be nice to have a feature that converts the content into LaTeX markup. In addition, it should be possible to export the list of bookmarks into a BibTeX file.

5.5 Conclusion

According to the SUS, the perceived usability of PaperViz is better than 98% of all products tested. The participants had almost no problems in accomplishing the tasks and they stated that the graph-based visualization is easy to understand and very valuable. Despite this very positive result, some weaknesses of the tool were identified based on the evaluation results. The most important finding was that the graph-based visualization of PaperViz gets difficult to interpret if lots of circles have to be visualized. This is definitely an aspect where the system needs improvement. Some other minor features could be improved as well. For instance it would be possible to provide more drag and drop support in order to reduce the number of clicks to perform actions like bookmarking a document or adding key phrases from the tag cloud to the visualization. In addition, some participants suggested other features on how to enhance the systems value, like a BibTeX extraction function or a flexible layout where users can adjust the components positions based on their needs.

Chapter 6

Conclusion & Future Work

Spotting high quality and relevant information within a large collection of documents is one of the main challenges researchers face nowadays when they perform investigative or learning tasks. Because of this reason, this thesis presents an innovative Web-based system called PaperViz, which is designed to support researchers with this task. The core component of PaperViz is an interactive graph-based visualization, designed to reduce the time of examining large document collections. This user-driven visualization, which is based on a key phrase extraction and weighting algorithm, allows the user to interactively inspect collections of documents at different levels of details. Moreover, the system provides other features that help researchers during the whole process of writing scientific literature, such as a Web-based text editor, a document metadata extraction function and a bookmarking and annotation function. These components enable users to quickly get an overview about the contents of document collections and their semantic structures. This in turn allows them to quickly spot relevant documents and pages of documents within such collections. Also PaperViz supports users in identifying additional key phrases that can be used to refine the graph-based visualization in order to retrieve more specific results and to find further interesting resources. By using PaperViz' bookmarking and annotation function, users can easily create and manage a list of relevant resources for a later detailed exploration. Generally speaking PaperViz helps researchers to significantly speed up the process of finding, evaluating and organizing scientific literature. Besides explaining all features of PaperViz in chapter 3, this thesis provides detailed insights on the architecture and technical implementation of the system in chapter 4. In order to identify the strengths and weaknesses of the system, a formative user study was conducted which is documented and discussed in chapter 5. The results of this evaluation indicate, that PaperViz is a suitable tool for supporting researchers in writing scientific papers. Despite the fact that the system receives an SUS score of 89, meaning that it has a higher perceived usability than 98% of all other products tested using SUS, some weaknesses were identified during

the evaluation. Furthermore some additional features would increase the value PaperViz:

1. Usability improvements:

- PaperViz should support a flexible layout, allowing the users to adjust the layout of the different components based on their needs.
- Providing more drag and drop support would save a lot of time and clicks and enhance the usability of features like the bookmarking and annotation function significantly.
- If the graph-based visualization contains an unmanageable amount of circles, PaperViz should aggregate clusters of circles in order to increase the usability of the visualization.
- Filtering out specific collection items should also be possible using the graph-based visualization component. Currently only the tree view component provides this feature.
- The initial loading time of the collection data should be reduced by implementing an intelligent incremental loading process.

2. Valuable extensions:

- Participants of the evaluation claimed that they would like to have a feature that exports a list of bookmarks into a BibTeX file.
- Especially when creating search queries consisting of a large number of key phrases (>5), it is sometimes difficult to place these key phrases in a way that maximizes the meaningfulness of the visualization. Therefore an automatic layout function for the key phrase placement would be a valuable feature.
- It would be possible to integrate other document collection visualization approaches into PaperViz. Such as Parallel Topics or ThemeScape (see section 2.1). This would enable researchers to analyze their collections from different perspectives.

3. Integrating other existing systems:

One of the big goals for the future is to combine PaperViz with other systems of similar scopes. uRank for instance is another user-driven ranking-based visualization tool designed to assist exploration and navigation of document collections. As the visualization of this tool does not support interactive features like drilling down or analyzing items in depth, it would be desirable to integrate at least the graph-based visualization model of PaperViz into uRank. Another project called EEXCESS (Enhancing Europe's eXchange in Cultural Educational and Scientific reSources) provides recommendation tools, delivering personalized recommendations from cultural and scientific repositories without having to explicitly search for this content. By implementing the API of EEXCESS, PaperViz would be capable of listing recommended documents, based on the list of bookmarks or the defined search query. This would definitely increase the value of PaperViz greatly.

To the best of our knowledge, PaperViz is the first Web-based tool that provides a variety of functions and features to support both, exploration of document collections and scientific paper writing (see chapter 2). Although the evaluation results indicate that it is an excellent tool, some features and functions need further improvement. To go one step further, PaperViz could be extended with additional valuable features and combined with other existing systems in order to gain more trust and acceptance.

Appendix A

Questionnaires

Abbr.	Questions	Likert-scale labels
Q1	How successful were you in accomplishing what you were asked to do?	one (Failure) to seven (Perfect)
Q2	How hard did you have to work to accomplish your level of performance?	one (Very High) to seven (Very Low)
Q3	Overall, this task was:	one (Very difficult) to seven (Very easy)

Table A.1: Questionnaire about workload and task difficulty for the evaluation of PaperViz

Abbr.	Questions
Q1	I found the graphical layout and the placement of the different elements of PaperViz well structured.
Q2	The different ways of navigating within the document collections are intuitive and easy to use.
Q3	I easily get lost when browsing deeper into a collection.
Q4	The filtering option of the Tree View navigation is a useful function.
Q5	The tag cloud is very helpful for finding relevant keywords of collections and documents.
Q6	The Drag and Drop function of the keywords is easy to use.
Q7	The graph-based visualization of PaperViz is easy to understand.
Q8	The positions, colors and sizes of the different elements of the visualization are easy to interpret –the meaning of these properties are clear to me.

Q9	I would like to know more about how these properties (positions, colors, sizes, etc.) are computed.
Q10	I often used the bar chart to identify relevant documents for keywords respectively relevant keywords for documents.
Q11	The colored edges between keywords and collection items are more confusing than helpful.
Q12	The gravity slider is a very helpful feature for increasing the visual separation of similar collection elements.
Q13	Large numbers of documents and sub collections within a collection strongly reduce the usability of the visualization.
Q14	The tile bar visualization is useful for finding relevant pages within a document.
Q15	I often read the information shown in the “Metadata information Area” to get more information about the document.
Q16	The bookmarking function of PaperViz is a useful and easy to use feature.

Table A.2: Questionnaire about the functions and components of PaperViz

Abbr.	Questions
Q1	I think that I would like to use PaperViz frequently.
Q2	I found PaperViz to be simple.
Q3	I thought PaperViz was easy to use.
Q4	I think I could use PaperViz without the support of a technical person.
Q5	I found the various functions in PaperViz were well integrated.
Q6	I thought there was a lot of consistency in PaperViz .
Q7	I would imagine that most people would learn to use PaperViz very quickly.
Q8	I found PaperViz very intuitive.
Q9	I felt very confident using PaperViz.
Q10	I could use PaperViz without having to learn anything new.

Table A.3: The System Usability Scale questions adapted from [40]

Appendix B

Acronyms

List of Acronyms

SPA	Single Page Application
TIARA	Text Insight via Automated Responsive Analytics
LDA	Latent Dirichlet Allocation
VIBE	Visualization By Example
POI	Point of Interests
FDL	Force Directed Layout
PCA	Principal Component Analysis
LLE	Local Linear Embedding
OCR	Optical Charater Recognition
MCV	Multiple Coordinated View
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
SaaS	Software as a Service
IIS	Internet Information Services
OOP	Object Orientated Programming
JSON	JavaScript Object Notation
EF	Entity Framework
OR	object-rational
MVVM	Model-View-ViewModel
DOM	Document Object Model
SUS	System Usability Scale
REST	Representational State Transfer

Bibliography

- [1] Alencar, A. B., Börner, K., Paulovich, F. V., and de Oliveira, M. C. F. (2012). Time-aware visualization of document collections. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 997–1004. ACM.
- [2] Basole, R. C. (2015). Text and document visualization. <https://cs4460.files.wordpress.com/2014/01/cs4460-spr15-08-textdocuments.pdf/>. [Online; accessed 17-March-2016].
- [3] Blankenship, D. (2010). *Applied research and evaluation methods in recreation*. Human Kinetics.
- [4] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.
- [5] Bootstrap (2015). Bootstrap. <http://getbootstrap.com/>. [Online; accessed 13-April-2016].
- [6] Cao, N., Sun, J., Lin, Y. R., Gotz, D., Liu, S., and Qu, H. (2010). Facetatlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1172–1181.
- [7] Chau, D. H., Kittur, A., Hong, J. I., and Faloutsos, C. (2011). Apolo: making sense of large network data by combining rich user interaction and machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 167–176. ACM.
- [8] Chen, C. (2006). *Information Visualization: Beyond the Horizon*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- [9] Community, G. (2015a). Restsharp. <http://restsharp.org/>. [Online; accessed 13-April-2016].
- [10] Community, W. (2015b). Representational state transfer. https://en.wikipedia.org/wiki/Representational_state_transfer/. [Online; accessed 13-April-2016].
- [11] Dou, W., Wang, X., Chang, R., and Ribarsky, W. (2011). Paralleltopics: A probabilistic approach to exploring document collections. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 231–240. IEEE.
- [12] Drahomira, Herrmannova, P., and Knoth (2012). Visual search for supporting content exploration in large document collections. *D-Lib Magazine*, 8(7).

- [13] Eisenstein, J., Chau, D. H., Kittur, A., and Xing, E. (2012). Topicviz: Interactive topic exploration in document collections. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 2177–2182, New York, NY, USA. ACM.
- [14] Frank, Parry, E., and Walton (2014). Results of a bibliographic software survey. http://www.lboro.ac.uk/media/wwwlboroacuk/content/library/downloads/surveyresults/RefWorks_EndNote_Mendeley_Survey_Report_2014.pdf/. [Online; accessed 18-March-2016].
- [15] für wissenschaftliche Anwendungen und Systeme Forschungs-und Entwicklungs GmbH, K. (2015). Sensium. <https://www.sensium.io/index.html>. [Online; accessed 21-May-2015].
- [16] Google (2015). Word trees. <https://developers.google.com/chart/interactive/docs/gallery/wordtree#overview/>. [Online; accessed 17-March-2016].
- [17] Görg, C. and Stasko, J. (2008). Jigsaw: investigative analysis on text document collections through visualization. In *Second International Workshop on Supporting Search and Sensemaking for Electronically Stored Information in Discovery Proceedings*.
- [18] Gretarsson, B., Odonovan, J., Bostandjiev, S., Höllerer, T., Asuncion, A., Newman, D., and Smyth, P. (2012). Topicnets: Visual analysis of large text corpora with topic modeling. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(2):23.
- [19] Havre, S., Hetzler, B., and Nowell, L. (2002). Themerivertm: In search of trends, patterns, and relationships. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20.
- [20] Hearst, M. A. (1995). Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 59–66, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- [21] Hwang, J. Y., Lee, J. S., and Lee, H. (2010). Designing a visual analytic system to represent bookmark sharing data.
- [22] jquery Foundation, T. (2015). jquery. <https://en.wikipedia.org/wiki/JQuery/>. [Online; accessed 13-April-2016].
- [23] Kaser, O. and Lemire, D. (2007). Tag-cloud drawing: Algorithms for cloud visualization. *CoRR*, abs/cs/0703109.

- [24] Kienreich, W., Sabol, V., and et al. (2003). Infosky: A system for visual exploration of very large, hierarchically structured knowledge spaces. In *IN PROCEEDINGS DER GI WORKSHOPWOCHE LLWA - WORKSHOP DER FACHGRUPPE FGWM (FACHGRUPPE WISSENSMANAGEMENT)*.
- [25] Klerkx, J. and Duval, E. (2009). Visualising social bookmarks.
- [26] Knockout (2015). Knockout. <http://knockoutjs.com/>. [Online; accessed 13-April-2016].
- [27] Laja, P. (2012). 10 useful findings about how people view websites. <http://conversionxl.com/10-useful-findings-about-how-people-view-websites/>. [Online; accessed 08-August-2016].
- [28] Marchionini, G. (2006). Exploratory search: From finding to understanding. *Commun. ACM*, 49(4):41–46.
- [29] Mendeley (2015). Mendeley authorization - oauth 2.0. http://dev.mendeley.com/reference/topics/authorization_overview.html/. [Online; accessed 13-April-2016].
- [30] Mendeley (2016). Compare reference management systems. <https://www.mendeley.com/compare-mendeley/>. [Online; accessed 18-March-2016].
- [31] Mendeley (2017). Mendeley website. <https://www.mendeley.com/>. [Online; accessed 19-March-2017].
- [32] Michael and Dittenbach (2010). Scoring and ranking techniques - tf-idf term weighting and cosine similarity. *Information Retrieval Facility*.
- [33] Microsoft (2015). Entity framework. <https://msdn.microsoft.com/en-us/data/ef.aspx/>. [Online; accessed 13-April-2016].
- [34] Newman, D., Baldwin, T., Cavedon, L., Huang, E., Karimi, S., Martinez, D., Scholer, F., and Zobel, J. (2010). Visualizing search results and document collections using topic maps. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2):169–175.
- [35] Olsen, K. (1991). *Ideation Through Visualization: The VIBE System*. Research reports. School of Library and Information Science, University of Pittsburgh.
- [36] Olsen, K. A., Korfhage, R. R., Sochats, K. M., Spring, M. B., and Williams, J. G. (1993a). Visualization of a document collection: The vibe system. *Information Processing & Management*, 29(1):69 – 81.

- [37] Olsen, K. A., Korfhage, R. R., Sochats, K. M., Spring, M. B., and Williams, J. G. (1993b). Visualization of a document collection: the vibe system. *Inf. Process. Manage.*, 29(1):69–81.
- [38] Osmani (2012). *Learning JavaScript Design Patterns*. O’Reilly Media.
- [39] Sauro, J. (2011). Measuring usability with the system usability scale (sus). <http://www.measuringu.com/sus.php/>. [Online; accessed 08-August-2016].
- [40] Sauro, J. and Lewis, J. R. (2012). *Quantifying the user experience: Practical statistics for user research*. Elsevier.
- [41] Sciencebuddies (2016). Resources for finding and accessing scientific papers. http://www.sciencebuddies.org/science-fair-projects/top_science-fair_finding_scientific_papers.shtml/. [Online; accessed 18-March-2016].
- [42] Seifert, C., Ulbrich, E., Kern, R., and Granitzer, M. (2013). Text representation for efficient document annotation. 19(3):383–405.
- [43] Strobel, H., Oelke, D., Rohrdantz, C., Stoffel, A., Keim, D. A., and Deussen, O. (2009). Document cards: A top trumps visualization for documents. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1145–1152.
- [44] Thomson (2011). Themescape. http://www.intellogist.com/wiki/Report:Thomson_Innovation/Viewing_Results/Analyzing_Results/ThemeScape/. [Online; accessed 17-March-2016].
- [45] van Ham, F., Wattenberg, M., and Viegas, F. B. (2009). Mapping text with phrase nets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1169–1176.
- [46] Wang Baldonado, M. Q., Woodruff, A., and Kuchinsky, A. (2000). Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI ’00*, pages 110–119, New York, NY, USA. ACM.
- [47] Wattenberg, M. (2011). Word tree. <http://fernandaviegas.com/wordtree.html/>. [Online; accessed 17-March-2016].