Eric Gergely, BSc

A Visually Aided Interactive Content-Based Digital Exploration System for Document Collections

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Compuer Science

submitted to

Graz University of Technology

Supervisor

Prof. Dr. Tobias Schreck
Institute of Computer Graphics and Knowledge Visualisation

Graz, Austria, Oct. 2020

Abstract

Finding relevant papers from document collections is an important task for researchers and scholars. However, retrieval systems and visualization frameworks for scientific publications often tend to neglect visual information (i.e., figures) of papers. The aim of this thesis is to implement a prototype document retrieval system for medium-sized document collections, that utilizes textual and visual information for the retrieval process. The system consists of five different views to support the user along different stages of the exploration process. Those views provide the user with various tools such as text similarity searches, content-based image retrieval, different visualization techniques and more. We evaluate the system by creating a fictional character called Alice, who conducts several scenarios on a given document collection. Our system provides promising results for the given scenarios and shows correlations between visual and textual information.

Kurzfassung

Relevante Dokumente in einer gegebenen Dokumentsammlung zu finden ist eine wichtige Aufgabe, mit der Forscher und Studierende oft konfrontiert sind. Leider vernachlässigen Retrievalsysteme und Visualisierungen für wissenschaftliche Arbeiten oftmals visuelle Inhalte (z.B. Bilder) von Dokumenten. Das Ziel dieser Arbeit ist es, ein Prototyp-Retrievalsystem für mittelgroße Dokumentsammlungen zu implementieren, welches sowohl textbasierte als auch bildbasierte Informationen bei der Suche verwendet. Unser System beinhaltet fünf unterschiedliche Ansichten, welche den Benutzer in den verschiedenen Phasen der Dokumentexploration unterstützen. Die verschiedenen Ansichten bieten dem Benutzer eine Vielzahl an Möglichkeiten, wie z.B. Suchen anhand von Text- oder Bildähnlichkeiten, unterschiedliche Visualisierungsmethoden und vieles mehr. Wir evaluieren das System indem wir eine fiktive Figur, namens Alice, unterschiedliche Szenarien an einer Dokumentsammlung durchführen lassen. Unser System zeigt vielversprechende Resultate für die beschriebenen Szenarien, welche unter anderem auf Zusammenhänge zwischen bildlichen und textbasierten Inhalten hinweisen.

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 indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Date	Signature

Acknowledgments

First and foremost, I would like to thank my supervisor Prof. Dr. Tobias Schreck for the opportunity to write this thesis. Further, I want to thank him and my advisor Lin Shao for always taking time for various meetings and providing me with great feedback and new ideas when I was stuck along this journey. Also, special thanks go to Hendrik Lücke-Tieke, who attempted various meetings and shared his knowledge with us by providing valuable feedback.

Moreover, I want to thank all my friends, especially Markus and Stefan, who made this journey so enjoyable.

Lastly, I want to thank my whole family for supporting me. Especially, my parents Claudia and Gerhard, and also my grandparents Erika and Karl, who always had an open ear for me and supported me throughout my whole life.

Thank you all!

Contents

List of Figures

1 Introduction

Contents

1.1 Information Retrieval in the Scientific Domain

Researchers and scholars are often confronted with the task of finding relevant literature or scientific publications within their field of research. Over the years, a vast amount of techniques and systems (e.g., information retrieval systems) has been developed and researched in order to facilitate information retrieval. Different tools are utilized depending on the task, research field, document corpus and document collection size. Traditionally, information retrieval systems, especially digital libraries, strongly rely on text or meta data based queries instead of images and visualizations. However, in past years new systems that use visualizations for meta data have been developed. Yet, visualization methods for scientific publications often neglect visual information (i.e., figures) [52]. Nonetheless, researchers recognized the importance of images for analyzing and finding relevant documents. Thus, different systems and visualizations employing visual information were developed. A major problem is, that many of them solely focus on images. Still, textual information should not be neglected either. As a consequence, the objective of this thesis is to create a proof-of-concept prototype document retrieval and visualization framework called Doxplorer. In addition to other tools, Doxplorer uses a novel, interactive document visualization that can use both, text and image search functionalities. The system aims to aid users in understanding and exploring the contents of a medium-sized document

collection. Hence, different tools like full-text search, summary creation, and image search are used in combination with a radial visualization method to support the exploration of a given document corpus. The proposed radial visualization method is heavily influenced by the RadViz visualization method and creates a 2D document landscape for multi document exploration. Creating such a system leads to a couple of questions:

- How to efficiently search and store documents?
- How to visualize the contents of a document?
- How to find relations between documents?
- In what way can documents differ from or resemble one another?
- How to convey the information to the user?

1.2 Information Retrieval Systems and Digital Libraries

Nowadays, various different kinds of information retrieval exist which are discussed in Chapter 2. Tamine-Lechani et al. [42] explain that Information Retrieval Systems (IRSs) basically have the goal to return the most relevant documents in response to a user query. Further, they describe that *IRSs* generally store documents and queries in textual objects, such as words. Consequentially, this indicates neglecting other information (e.g., visual information). Undoubtedly, a digital library can be considered as an *IRS*. In simple terms one can think of a digital library as of a traditional library but in digital form; meaning that the contents (documents) are stored digitally. Of course, there are many different definitions which also change over time. For example, in 1998 the Digital Library Federation defined digital libraries as:

"Organizations that provide the resources, including the specialized staff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily available for use by a defined community or set of communities" [33].

Digital libraries are an important source for researchers and students to find scientific publications within their respective field of research. They often consist of large-sized document corpora which makes it difficult to find all documents that could be of interest to the user. This is especially true for digital libraries that are provided online. Collections of such size (e.g., ACM Digital Library) often solely utilize text-based user queries and filtering by meta data for information retrieval. But, there are also other information retrieval systems that focus on small or medium-sized document corpora which allows for different visualizations. They can be very useful for exploring subsets of larger collections. While

many information retrieval systems and their visualizations are text and meta data based, depending on the domain, images or visual representations of results can be as important as textual information. This brings up challenges, such as how to best store, explore, combine and display these different types of information. While the prototype system of this thesis focuses on medium-sized document collections, it definitely has commonalities with digital libraries (e.g., full-text search).

1.3 Document Visualization

The term "document" can be used to describe a variety of different data types in the digital domain. If not stated otherwise in this thesis, the term document is used to describe a Portable Document Format (PDF) document that usually contains text, images and other meta data. Being bound by time constraints, most people do not have the time to fully read every paper that could be of interest to them. This leads to the question: How can we tell if a document is important without reading it fully? Document visualizations try to provide a solution to this very problem. M. Li et al. [14] provide a general overview of different document visualization techniques. They point out that document visualizations can be categorized in single or multi document visualizations, where single document visualizations primarily focus on single words and core contents of documents, while multi document visualizations tend to focus on relations, topics and concepts. The overview they provided indicates that most of the research in this field focuses on text-based content and meta data. However, some researchers developed systems that also make use of visual information, which greatly benefits certain domains (e.g., visual analytics). For example, Strobelt et al. [39] introduced a visualization method called Document Cards which shows the core contents of a document in concise form. These aforementioned Document Cards contain information such as important terms, authors, images and other meta data, on a single page.

1.4 Outline

This thesis is structured in six chapters. Chapter 2 is about previous work that is related to this thesis. Afterwards, Chapter 3 describes the ideas and core concept behind the system, while Chapter 4 explains the implementation of the system by describing algorithms, libraries and the system design. Those chapters are followed by Chapter 5, which includes use cases and an evaluation of the system on the VAST data set. Chapter 6 is the last chapter and starts with a discussion and an outlook on future work. The chapter ends with a conclusion summing up the thesis.

2 Related Work

Contents

This thesis aims to enhance traditional text-based document retrieval with an interactive visualization and content-based image retrieval by creating a prototype document retrieval system. This chapter is dedicated to put this work into perspective with already existing work from related fields.

2.1 Information Retrieval Systems

In the age of technology we are confronted with large amounts of information more than ever. Finding relevant information in a seemingly infinite sea of data is a crucial but difficult task. Therefore, after the invention of computers Information Retrieval (IR) was already born in the 1950s as a necessity to tackle this problem as described by A. Singhal [37]. In his work he also describes that information retrieval has come a long way in past years. Moreover, Singhal describes the most commonly used models of standard *IR* (text-based) which are vector space models [32], probabilistic models and inference network models [45] as follows:

Vector space models in general, are a very high-dimensional vector space where each term constitutes a single dimension. Typically, in most cases terms consist of words or phrases. This allows for queries and documents to be represented by vectors. If a term occurs in the document (or query), the vector entry of the dimension for this term is a non-zero value. Consequentially, it is zero otherwise. A similarity score between documents

and queries can be calculated by using their respective vectors. The calculation of the similarity scores varies depending on different vector space models but often depicts an angle between two vectors.

Probabilistic models typically calculate a relevance probability estimation of documents to a given query. Documents of a given collection are then sorted in decreasing order depending on their probability score. In literature this principle is often referred to as *the probabilistic ranking principle* [29]. This type of model was originally proposed by Maron and Kuhns [22]. The implementation of the probability estimation varies between different probabilistic models. Over the years many different approaches were introduced.

Inference network models realize the retrieval of documents as inference process in an inference network [45]. This means that for each document there is a weight which defines how strong the document instantiates a term. A document score is established by combining the term weights of the document for a specific query. The calculation of the weights can be chosen freely and depends on the specific inference network model.

While our system focuses on images, meta data and text, it is important to mention that information retrieval in general also covers other data types such as audio [12]. In modern information retrieval there exist various different branches; for example, Private Information Retrieval (PIR) [7], Cross-Language Information Retrieval (CLIR) [26], Content Based Image Retrieval (CBIR) [38] and contextual information retrieval which are active fields of research.

Tamine-Lechani et al. [42] argue that the amount of information makes traditional *IR* approaches less effective. Similarly, X. Shen et al. [35] consider the absence of user and search context as a major limitation for information retrieval systems. Around 2010, in their work, L. Tamine-Lechani and her colleagues [42] also revealed a growing interest towards contextual information in *IR*. They explain this rise of interest due to users being overwhelmed by the information overload they receive when issuing queries to information retrieval systems. Their work describes that contextual information retrieval relies on different sources (e.g., user interests, preferences, time and location). While context-based information retrieval systems use different definitions for the core concept of users' context, they have the same core idea which is to present the user with the most accurate information by utilizing the provided context [42]. For instance, X. Shen et al. [35] studied how implicit feedback can help to better the accuracy of results returned by the information retrieval system. Their approach uses search contexts, such as past navigation actions and queries, to retrieve better results from the system. Figure 2.1 provides an overview of different contextual information sources. As we can see, there are various different aspects of context to further refine information retrieval.

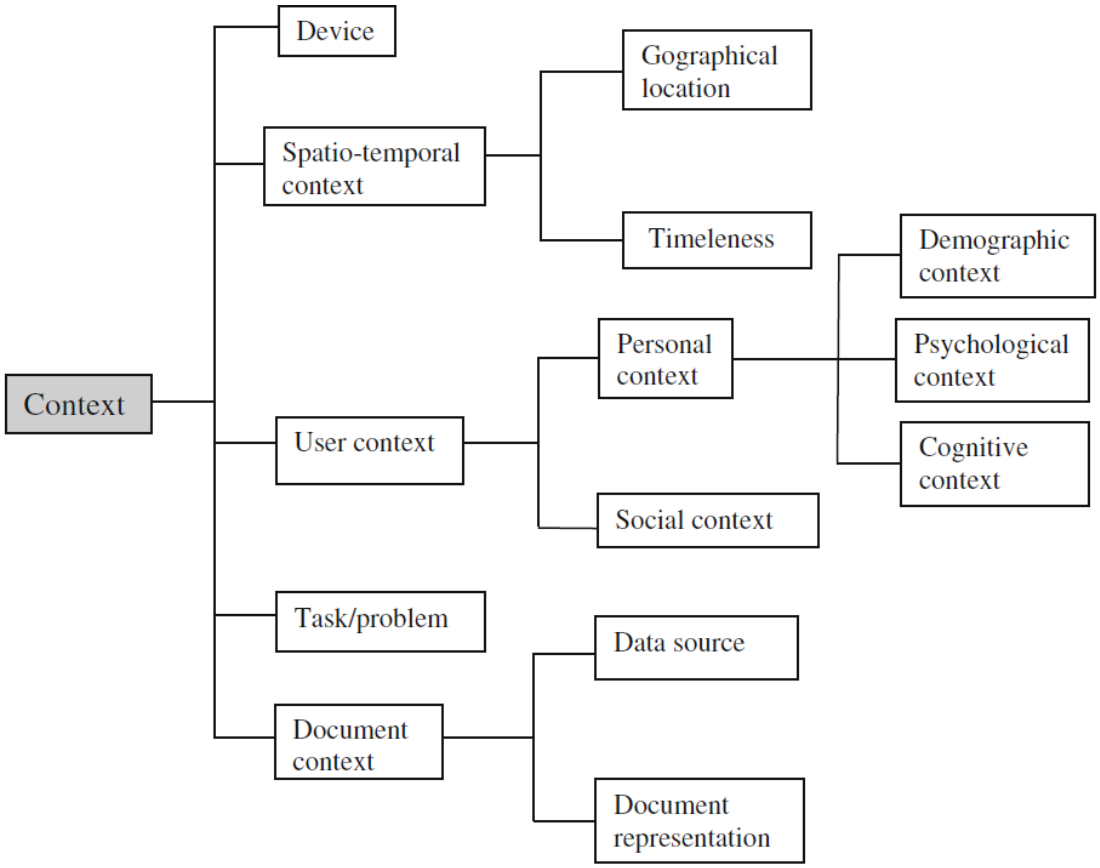


Figure 2.1: Different aspects of context in the information retrieval domain. Figure taken from [42].

In recent years, also a lot of research was done in the field of *PIR* [1, 2, 13, 34, 41]. As described by K. Banawan et al. [1], the *PIR* problem is defined by a user retrieving a specific message without any of the individual databases knowing the identity of the message. Research in this area tends towards capacities of *PIR* with certain database structures, about *PIR* with side information or noisy *PIR*.

In the last years, researchers, such as B. Mitra et al. [23], also investigated neural ranking models for learning vector representation of text. In their work, they point out that tremendous improvements have been made in the fields of computer vision, speech recognition and machine translation tasks by using neural network models consisting of multiple hidden layers, which are called deep architectures. Around 2017, work has begun in the *IR* community to use this neural models in order to expand the state of the art or even achieve new levels of performance as in other areas of computer science [23]. Figure 2.2 illustrates the increase in interest.

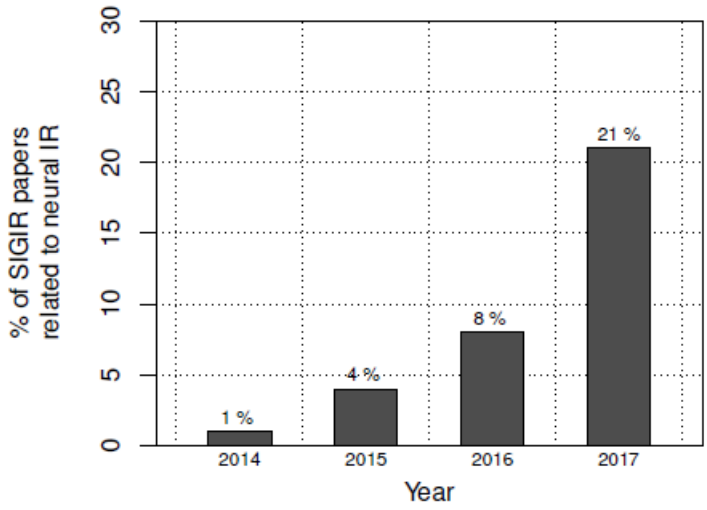


Figure 2.2: Percentage of neural *IR* papers from the ACM SIGIR conference for different years. Figure and description are taken from [23].

2.2 Document Visualization

According to M. Li et al. [14] efficient and effective visualization tools became a necessity due to the amount of information and documents available to us nowadays. In an overview, they present the fundamental concepts of document visualizations while also discussing their main challenges. They categorize document visualization in three different groups: single document visualization, document collection visualization and extended document visualization. Single document visualizations focus on single words, phrases and core contents of a document. Whereas, document collection visualizations target relations, themes and concepts of collections. Extended document visualizations are more concerned with attributes that are outside the document content and applied in special fields, for example, social media. In 1996, B. Shneiderman [36] introduced a type by task taxonomy for visualizations. According to his mantra, there are seven different tasks a visualization can aim for. Those tasks are:

- *Overview:* Gain an overview of the entire collection.
- *Zoom:* Zoom in on items of interest.
- *Filter:* Filter out uninteresting items.
- *Details-on-demand:* Select an item or group and get details when needed.
- *Relate:* View relationship among items.

- *History:* Keep a history of actions to support undo, replay, and progressive refinement.
- *Extract:* Allow extraction of sub-collections and of the query parameters.

Our prototype combines different ideas to achieve most of these principles to a certain degree within a single system.

2.2.1 Single Document Visualization

Single document visualizations can be of great use depending on the collection size and task at hand. Furthermore, they should provide a concise overview of the core contents of a document and its text features [14]. To this date, there exist various distinct techniques with different priorities for visualizing single documents. There are also visualizations which could be classified as both, single and document collection visualizations.

An example of such a visualization is called *Document Cards* and was proposed by Strobelt et al. [39]. They describe the creation process of those cards as a pipeline. After extracting the text, they find the most relevant key words by text mining. Simultaneously, they perform image extraction and preprocess the retrieved images. The last two steps in the pipeline are image packing and text placement. In Fig. 2.3 numerous different example Document Cards are shown. The cards present authors, title, important terms and images of a document on a single page. Moreover, on the right side the user has the option to select any page and view it fully. Users can inspect the page where a specific image is located by clicking on the image. Additionally, terms can be selected to highlight their occurrences. This means that lines and page numbers (on the right) that contain the selected term are highlighted. Also, if the caption text of an image contains a selected term, the image is highlighted. Hovering over the non-image space of a Document Card displays a tooltip which shows the abstract of the corresponding document. This visualization can help to grasp the core contents of a document quickly and view single pages fully if needed. While one card only represents a single document, the authors implemented a system which shows all Document Cards for a given collection in a matrix view as can be seen in Fig. 2.3. Consequentially, the visualization is not only able to visualize single documents but also multiple documents at the same time. In order to aid the exploration process, we also included a simpler, more basic version of such Document Cards within our system in the *VisualExplorerView*. While Document Cards also include images and meta data, such as authors, most single document visualizations tend to be only text-based.

A well known and widely used text-based visualization technique are Tag Clouds [14, 15, 49] (especially for news media). Tag Clouds generally provide an overview of term frequencies for a given text. Note that the input text can also stem from multiple

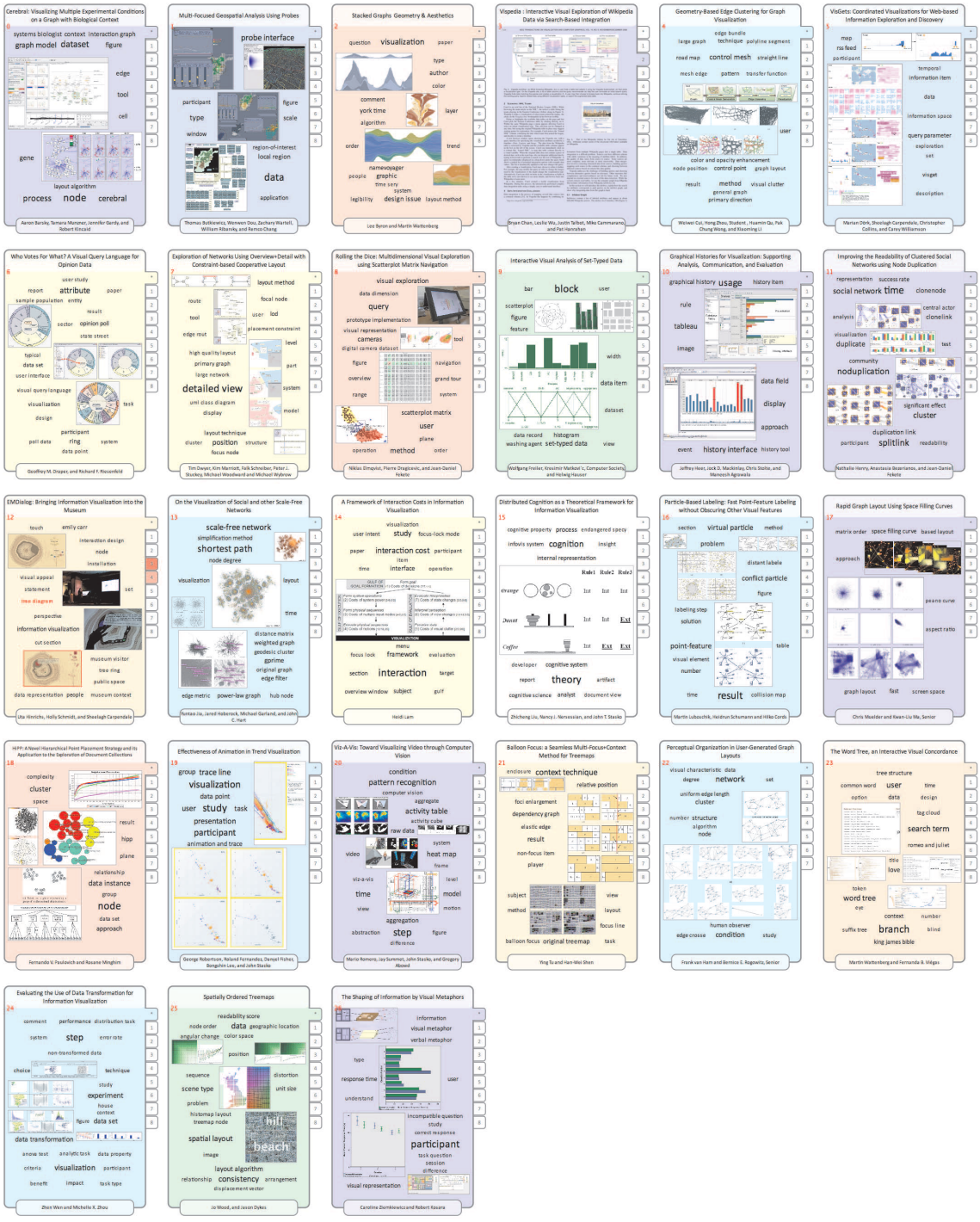


Figure 2.3: This figure shows Document Cards for visualizing documents contained in the IEEE InfoVis 2008 proceedings. When selecting a term its frequency for the different pages is illustrated on the right side of the Document Card (higher frequency is indicated by a "stronger" red). [39]

documents and is not restricted to a single one. The layout and graphical representation can be done in various different ways. Still, most of the time position, colors and size are used to visualize the importance of a term. This means, for example, that important terms might have a bigger font-size or are centered in the middle. However, the importance could also be only shown by color and size while the layout may be alphabetically. Unimportant terms can be dropped in order to prevent the visualization from congestion. For this reason, Venetis et al. [48] conducted a study on which terms to select best for Tag Clouds by comparing different tag selection algorithms. In their work, they compared a popularity algorithm, a tf-idf based algorithm and the maximum coverage algorithm and show that selecting the correct terms is still a difficult challenge. The algorithms performed differently depending on the domain of the text.

There are other visualizations, such as Semantic Graphs [31] proposed by D. Rusu et al., which target to reveal the semantic structure of a document instead of frequency distributions. In order to build such graphs, they start by extracting **subject - verb - object** triples for every sentence with the Penn Treebank parse tree. Then, those triplets are assigned to their corresponding entities. In order to find as many good links as possible, they use WordNet synsets together with pronominal anaphors. Furthermore, in their work Rusu and his colleagues propose a way to generate summaries by using the semantic graph and the triplet list together. Figure 2.4 shows an example of a semantic sub-graph.

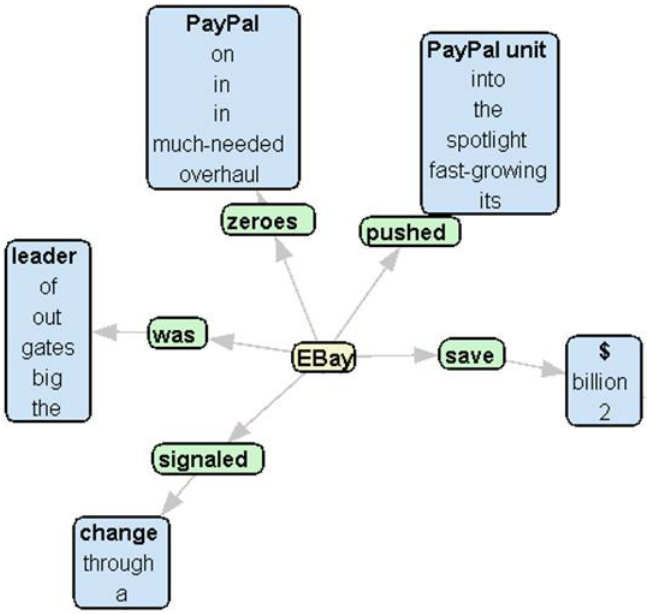


Figure 2.4: Semantic sub-graph of a given text. [31]

Even if a single term is considered important, it might not deliver enough information without proper context. Thus, there are visualizations which target to provide not only term frequencies but also the information in which context words are used. Therefore, M. Wattenberg and F. B. Viegas [50] introduced a visualization and information retrieval method called Word Tree. Their visualization is solely text-based and can be queried in order to explore various input texts. Their method is based on the idea of "keyword in context" [11], where query results are displayed together with snippets that show the terms along with the text surrounding them. Figure 2.5 shows a small example of a Word Tree for the search term "if love". The authors also describe that in this visualization the relative size of terms is determined by the square root of their frequency. Users can enter a search term for creating a Word Tree. After the initial Word Tree is established, the user can select terms or phrases from that tree to generate a new one. Of course, the user can also enter new search terms. This interactivity allows broadening and narrowing a text search.

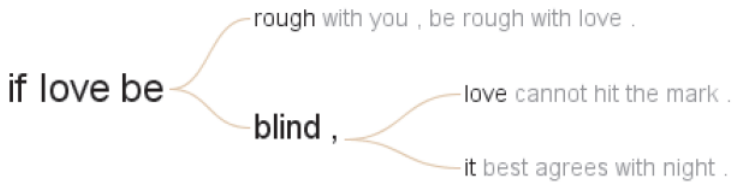


Figure 2.5: Example of a Word Tree for search term "if love" in Romeo and Juliet. [50]

2.2.2 Document Collection Visualization

Visualizations for entire document collections are crucial due to the amounts of data we are confronted with nowadays. They can help to find hidden and core topics, relations among documents, topic changes over time and more. M. Li et al. [14] classify document collection visualizations from different aspects:

1. Visualization of themes
2. Visualization of core contents
3. Visualization of changes over different versions
4. Visualization of document relationships
5. Visualization of document similarity

It is important to mention, that document collections can also be visualized by combining different visualizations within a single framework. For example, P.

Riehmann et al. [28] proposed a visualization framework that helps users analyze and understand the contents of medium-sized document collections. Their framework utilizes document visualizations that focus on different aspects. Their visualizations are based on glyphs which show structure and temporal information. Among other features, they implemented a *TopicCalendar* along with *TopicTrends* in order to provide users with a topic overview. Moreover, a so called *ConceptCircuit* reveals different entities of a collection. Those entities can depict anything ranging from places to persons.

A. Dong et al. [52] created another interesting visualization framework called *VIStory*, which focuses on visual information (i.e., figures) of documents. Their motivation was that exploration methods for scientific publications often neglect visual information. The *VIStory* interface incorporates a Faceted View, which enables querying by meta data. Moreover, the interface comprises documents encoded as glyphs which are arranged in a ThemeRiver [16] layout in the Storyboard View. Glyphs are basically paper rings which summarize image attributes such as color and size. A paper ring consists of different segments, where each segment depicts a figure from the paper. The length and color of each segment is determined by its corresponding figure. Figure 2.6 provides and example of such a glyph. Although VIStory enables filtering by meta data and reports all images from a document, it does not support a content-based image search functionality. While it is possible to do high-level analysis, such as investigating which colors are used most by a specific author, there is no efficient way find similar visualizations or figures. In contrast, our prototype aims to enhance their idea of using visual information for exploration of scientific publications by enabling the user to search for documents with similar images. Furthermore, our prototype combines this idea with the strengths of text-based retrieval methods.

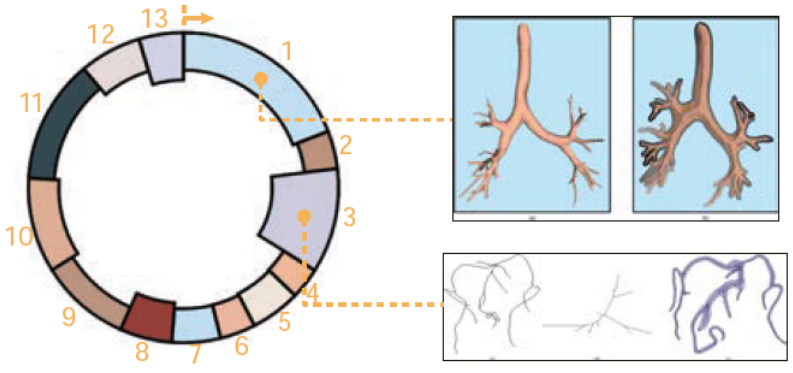


Figure 2.6: Shows an example of a paper ring glyph where the input paper consists of 13 images. Figure taken from [52].

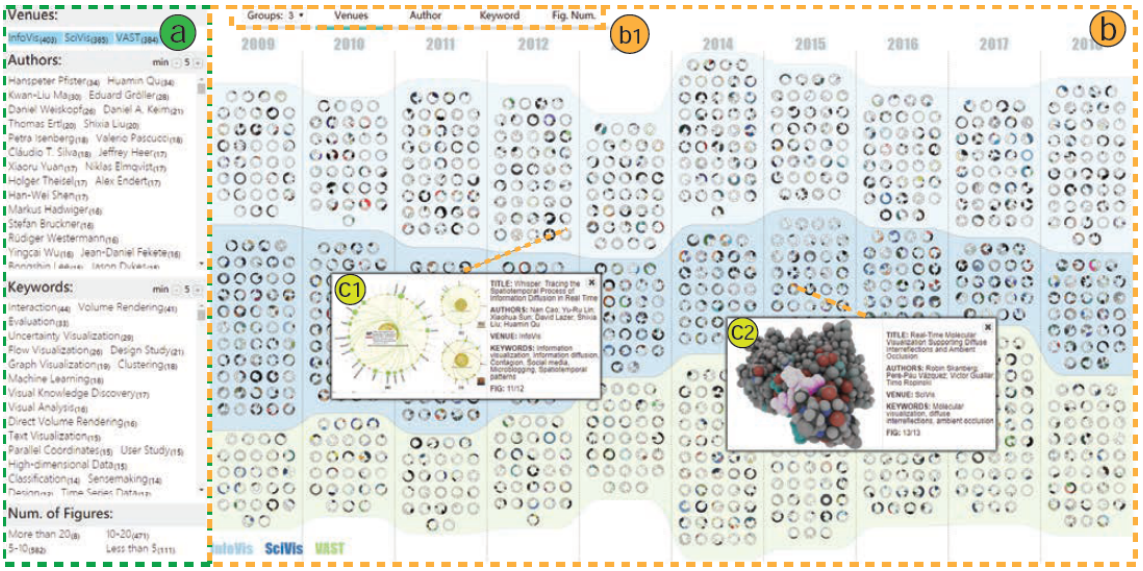


Figure 2.7: VIStory interface showing three different data sets of scientific publications. On the left panel is the Faceted View for meta data querying. The main panel shows the three collections in a theme river layout with publication years as columns. Selecting a glyph shows the Endgame View, which displays information such as title, keywords and images. Figure and description are taken from [52].

Finding document affiliations to certain topics or discovering underlying topics of a collection can be especially helpful for the exploration process. Document landscapes, such as the IN-SPIRE ThemeView [44] and IN-SPIRE Galaxy [51], can be used as a visualization tool for showing document distributions over topics. J. Thomas et al. [44] describe ThemeView as a 3D visualization that represents topics as mountains while the distance between two mountains reflects the topic similarities. Further, they explain that mountains are separated by keywords and that their height is relative to the respective topic strengths. In their work, they also describe the Galaxy visualization which is a 2D visualization where documents are represented by points which form a theme galaxy. J. Thomas and his colleagues explain that the galaxies use either hierarchical clustering or k-means clustering. The centroids of clusters are also visualized along with text labels showing their corresponding core themes. In another work, P. C. Wong et al. [51] write about strengths and weaknesses of the Galaxy visualization and further describe that users can interactively explore the galaxy by selecting documents or moving aside uninteresting clumps. Moreover, they defined that after each interaction the Galaxy is re-clustered and re-projected.

In contrast, there are topic centered visualizations that have a strong emphasis on the temporal aspect as well. For example, S. Havre et al. [16] presented a method called ThemeRiver. In their work, they describe that the “river” flows through time from left to right while changing its width according to topic strengths. Further, they explain that

the river consists of differently colored currents which represent single topics or groups of topics. By using this technique, they can visualize the relevance of topics at different times compared to other topics. ThemeRiver as a visualization aims to illustrate theme changes over a course of time [16]. Figure 2.8 provides an example of the ThemeRiver prototype system visualization proposed by S. Havre and her colleagues.

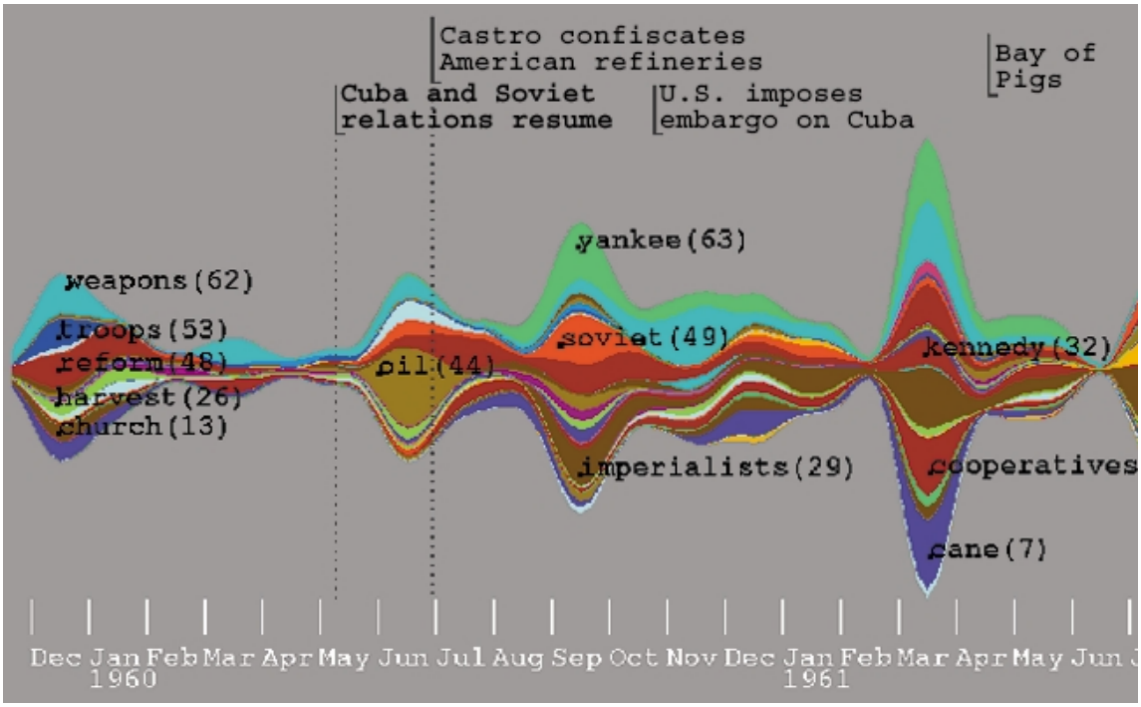


Figure 2.8: Illustrative example of a ThemeRiver visualization with X-axis representing time while Y-axis represents topics. Figure taken from [16].

On the other hand, there are document collection visualizations that aim to depict relations among different documents. M. Dörk et al. [9] created such a visualization technique called PivotPaths which provides a graphical and interactive interface for exploring relations based on meta data (e.g., keywords, authors and citations) in scientific publications. Their work clarifies that filtering is useful for information retrieval but that it is hard to comprehend the effect of filter operations without seeing the relations among different filter options and items. Further, M. Dörk and his colleagues point out that traditional filtering usually results in abrupt changes from one data set to another. As a consequence, their visualization aims to depict relations between different meta data attributes which can help to understand filter results. For example, they show that the user can select an author and show all his meta data relations (see Figure 2.9). Naturally, there are various other visualizations based on meta data relations such as ContexTour [19] and FacetAtlas [3].

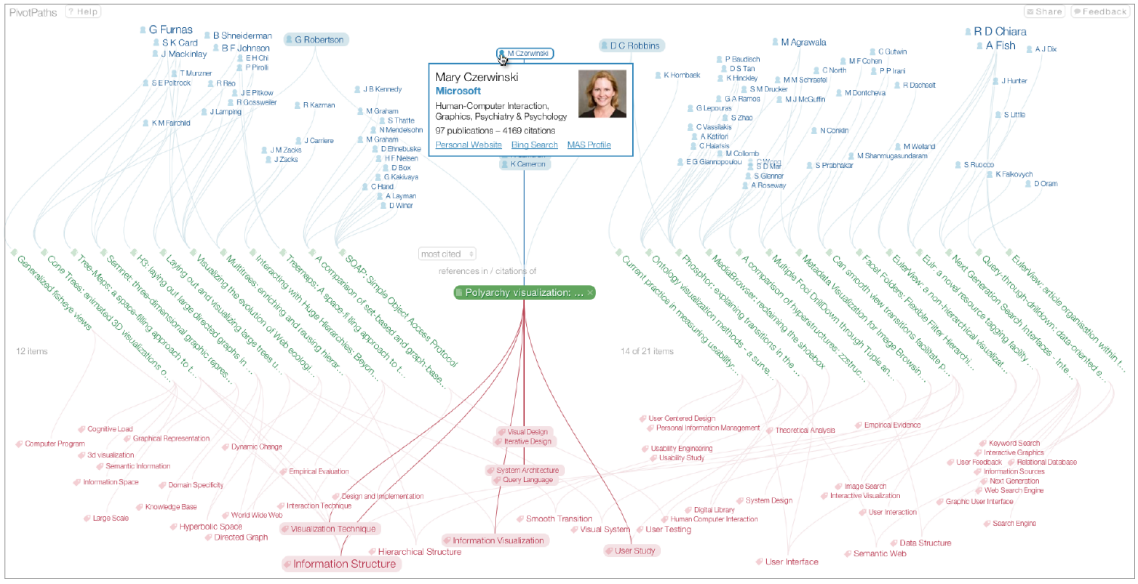


Figure 2.9: PivotPath visualization of a document where references of a specific paper and its citations are used to show details for a selected author. Figure and description are taken from [9].

2.3 Text Summarization

Text summarization aims to extract the core information from an authentic text and provides the user with a concise summary report [24]. Automatic text summarization can save a lot of time when researching or trying to get an overview of documents or document collections. There are two different types of text summarization, namely, (1) extractive text summarization and (2) abstractive text summarization [40]. Extractive text summarization methods create summaries by selecting sentences from the original document; whereas abstractive methods generate summaries with arbitrary words and expressions, which resemble summaries created by humans [43]. In their work, J. Tan et al. [43] describe abstractive summarization as the ultimate goal of text summary research. Nevertheless, they depict a previous shortage of research in this direction due to immaturity of text generation techniques. But on the other hand, their work points out that impressive progress was made by using neural networks lately. Still, J. Tan and his colleagues describe abstractive summarization to be in a primitive stage and that extractive methods have better evaluation scores on benchmark data sets. However, they state that abstractive methods are generally able to generate better summaries, while being more difficult to produce. The findings of V. Dalal and Dr. L. Malik [8], who conducted a survey on extractive and abstractive text summarization techniques, support this claim. They also came to the conclusion that abstractive approaches can outperform extractive methods in terms of quality but that they are more expensive computationally. Nonetheless, there were also significant developments in the field of extractive summarization where neural networks

were also widely used, as shown in a survey on extractive text summarization conducted by N. Moratanach et al. [24]. Figure 2.10 provides an overview of different methods which are discussed in their survey. J. Cheng and M. Lapata [6], for instance, introduced (extractive) neural summarization as a data-driven approach by extracting sentences and words. Their system uses a document encoder together with an attention-based extractor where two models are trained on hundreds of thousands of document-summary pairs. Their proposed system achieved promising results with the sentence based model on the DUC 2002 test data set using Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [18] metrics [6]. Moreover, Nallapati et al. [25] also developed a very promising method that uses a recurrent neural network based sequence model for generating extractive summaries. In fact, results indicate that neural sentence extraction performs better than traditional extraction methods [43].

Categories	Methodology	Concept	Advantages	Limitations
SUPERVISED LEARNING APPROACHES	Machine Learning approach Bayes rule	Summarization task modelled as classification problem	Large set of training data improves the sentence selection for summary	Human interruption required for generating manual summaries
SUPERVISED LEARNING APPROACHES	Artificial Neural Network	Trainable summarization - neural network is trained , pruned and generalized to filter sentences and classify them as "summary" or "non-summary sentence"	The network can be trained according to the style of human reader. The set of features can be altered to reflect user's need and requirements	1)Neural Network is slow in training phase and also in application phase. 2) It is difficult to determine how the net makes decision. 3) Requires human interruption for training data
SUPERVISED LEARNING APPROACHES	Conditional Random Fields (CRF)	Statistical modelling approach which uses CRF as a sequence labelling problem	Identifies correct features and provides better representation of sentences and groups terms appropriately into its segments	1) focuses on domain specific which requires an external domain specific corpus for training step. 2) Limitation is that linguistic features are not considered
UNSUPERVISED LEARNING APPROACHES	Graph based Approach	Construction of graph to capture relationship between sentences	1) Captures redundant information 2)Improves coherency	Doesn't focus on issues such as dangling anaphora problem
UNSUPERVISED LEARNING APPROACHES	Concept oriented approach	Importance of sentences calculated based on the concepts retrieved from external knowledge base(wikipedia, HowNet)	incorporation of similarity measures to reduce redundancy	Dangling anaphora and verb referents not considered
UNSUPERVISED LEARNING APPROACHES	Fuzzy Logic based approach	Summarization based on fuzzy rule using various sets of features	improved quality in summary by maintaining coherency	membership functions and work of the fuzzy logic system

Figure 2.10: Overview of different supervised and unsupervised learning methods for extractive summary generation. Table and description are taken from [24].

3 Concept

Contents

3.1 Outline

The general idea behind our prototype system is to combine basic tools of document retrieval systems (e.g., indexing documents, meta data information and query-based text search) with document visualizations and a novel, interactive multi-document visualization view, that is strongly inspired by the RadViz [27] visualization technique. The goal is to see how document retrieval systems, especially digital libraries, could benefit from integrating visualizations and image browsers. In order to aid the user while exploring a document collection, the prototype system consists of different views, which also support different levels of detail. When exploring the document corpus, the user can tag documents as interesting for further investigation at a later time. The user might consider different documents as important for various reasons (e.g., images, content or authors). Thus, the user can tag the documents by using a color tagging system (which supports three colors). Tagged documents are managed as a subset of the entire document corpus, which in turn can be further examined. Depending on the current state of the exploration process, the distinct views can be utilized for different tasks. The system aims to support the user in three ways:

1. Find documents of interest within the document corpus.

2. Discover new documents that have relations with previously found documents.

3. Find new relations and commonalities in the subset of interesting documents.

The following sections describe how the different views support the user in the aforementioned tasks. Note, at any time it is possible to switch between views. For illustration purposes, mock-up screenshots of previous design iterations are presented in this chapter.

3.2 Finding Relevant Documents

Since documents can be important to the user for various reasons, the system provides different options for querying the collection. The two main options for finding an initial set of interesting documents are explained in this section.

3.2.1 Text Queries

To find documents that are of interest to the user, Doxplorer provides the user with a text-based query system. Different term modifiers, such as wildcard searches and Boolean operators enable effective querying of large text corpora. Figure 3.1 shows an early sketch for the *SearchView*. Initially, the *SearchView* displays all documents for browsing and is usually the starting point of the exploration process. Within this view, the user can initiate text-based search queries and tag interesting documents. After a search query was entered, a full-text search starts and the top results for the specified query are presented. The user can then inspect single documents from the results in a different view (i.e., *InspectView*), open them in a Portable Document Format (PDF) viewer or directly tag a specific document as important.

3.2.2 Image Queries

As mentioned in previous chapters, images, especially in the field of data visualization, can be a critical factor when deciding the importance of a document. The design of the *ImageView* resembles the *SearchView* in structure. This means that an image list of all documents, which the user can browse through, is displayed. By selecting an image from the image browser or by uploading a local image, the user can initiate an image search. The 50 top most similar images and their respective similarity scores are returned and displayed in descending order. Of course, the list also indicates to which document a certain image belongs. Naturally, when considering an image as important the user can tag the associated document for further examination.

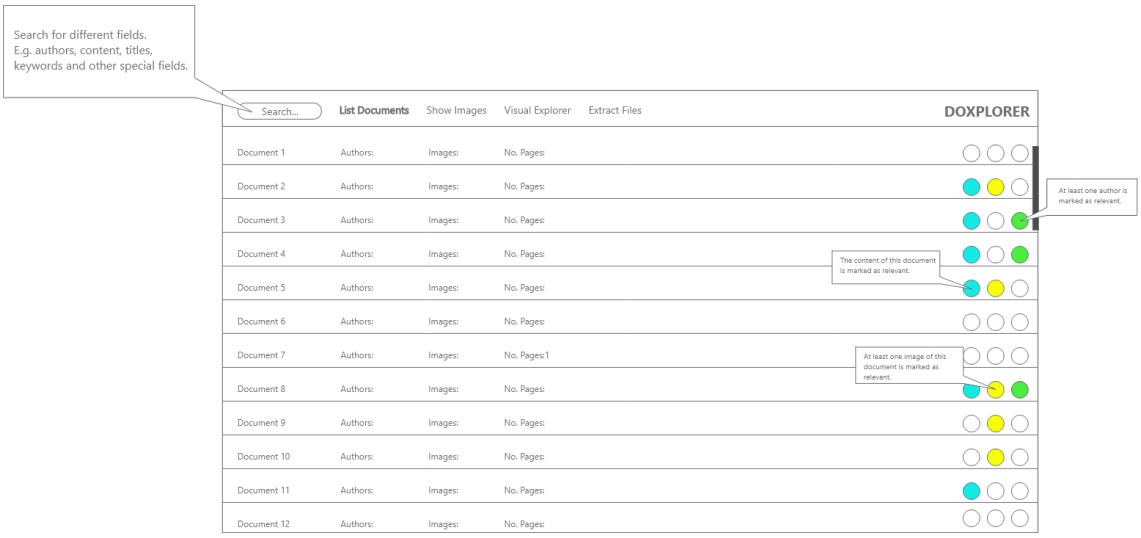


Figure 3.1: *SearchView* displaying a list of all documents from a collection. The color tagging system is shown next to each document. Search queries can be entered in the top left.

3.3 Discovering Similar Documents

Finding similar documents is an important task researchers are often confronted with. However, documents can resemble one another in various different ways (e.g., similar images, same authors or topics). Since the goal of this thesis is a proof-of-concept system that aims to combine text and image search with visualizations, we focus on basic similarities, such as authors, year of publication, text and images. There are different ways for the user to find similar documents when using the Doxplorer system.

3.3.1 Visual Explorer

The *VisualExplorerView*, see Figure 3.2, sorts the document corpus according to years of publication. Documents missing this information are put in a separate “unknown date” category. Documents are depicted by circles within this view. Documents previously tagged as important are visually emphasized and stored in a subset. When the user finds new interesting documents in this view, he/she can add them or remove previously tagged documents from the interest list. By selecting a circle, a document card for the corresponding document can be opened. The document card shows basic information about the selected document and provides search functionalities, see Figure 3.3. The basic information consists of author names, document title and images of the document. There are three different search options provided by document cards:

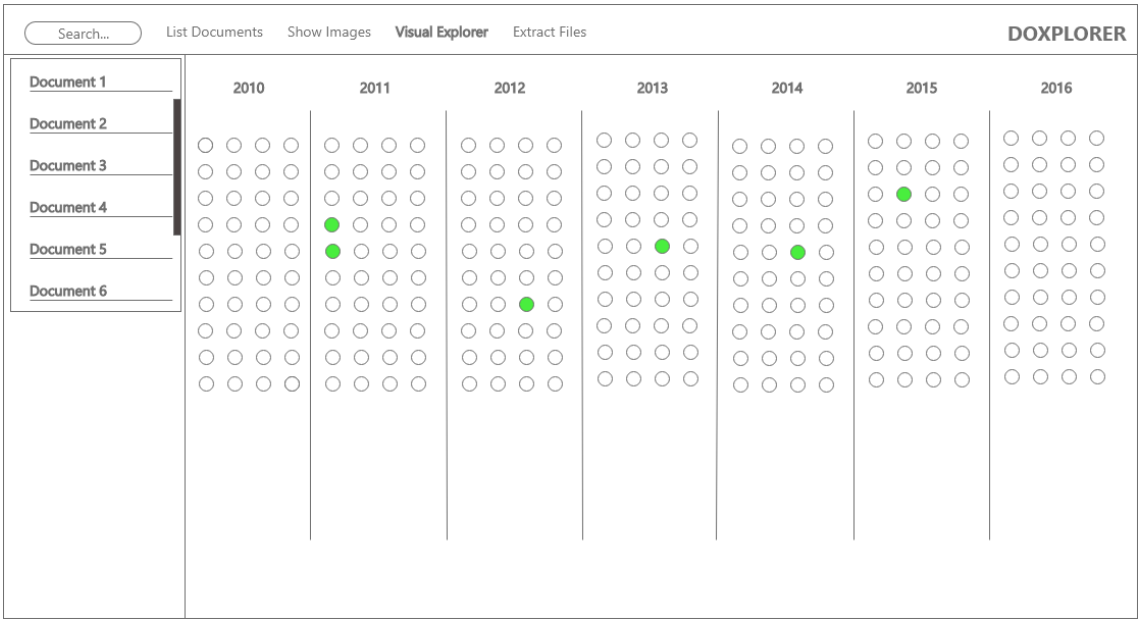


Figure 3.2: *VisualExplorerView* concept mock-up. Circles depict documents and are sorted according to their year of publication. A list view, shown on the left, displays previously tagged documents. Circles representing tagged documents are highlighted visually.

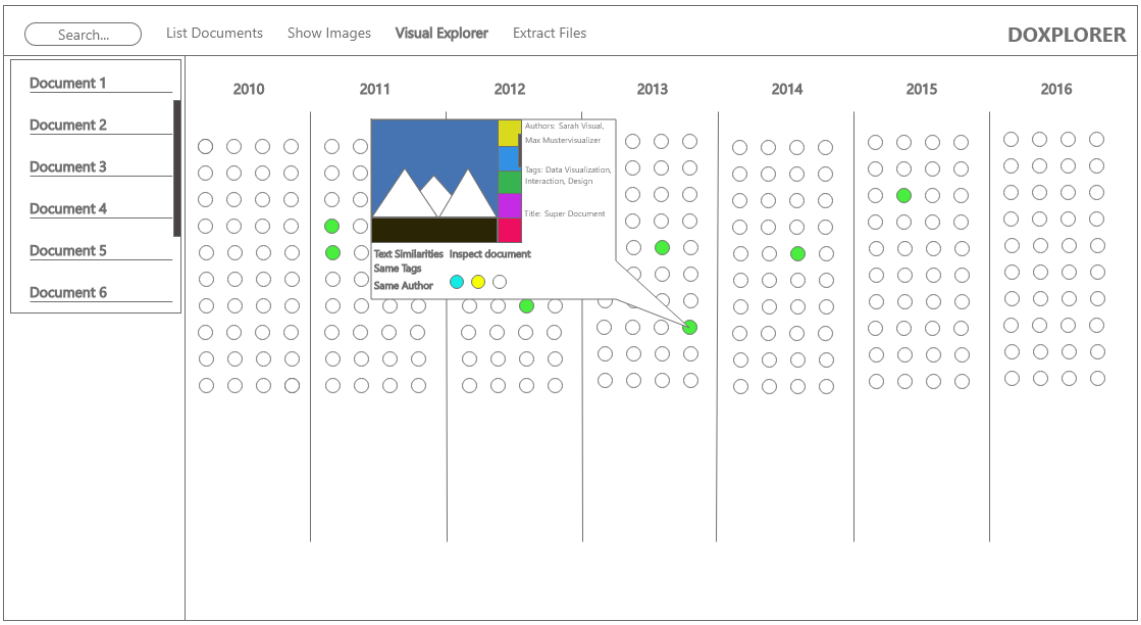


Figure 3.3: *VisualExplorerView* concept mock-up after selecting a circle. By selecting a circle, the user can open a document card that depicts core contents of the corresponding document including authors, title and images.

- Author: The user is able to see all authors from a document on the document card and can search for them within the *ExplorerView*. Documents written by the queried author are then visually emphasized and can be further investigated.
- Text: Another feature is the text-based similarity search for finding related documents. For a detailed explanation on the text-based search see Chapter 4.
- Image: Similarly, the user can initiate an image similarity search by using a document card. Since a document card displays a list of all images for a given document, the user can initiate the search by clicking at any image. The documents containing the most similar images (compared to the selected image) are, then again, visually highlighted within the *VisualExplorerView*.

Additionally, this view features a tf-idf based word distribution visualization. The visualization aims to show the most important key terms for the subset of previously tagged documents. This should help the user to better grasp the core contents of the selected subset. This knowledge can be further employed within the *RadVizDoc* visualization view.

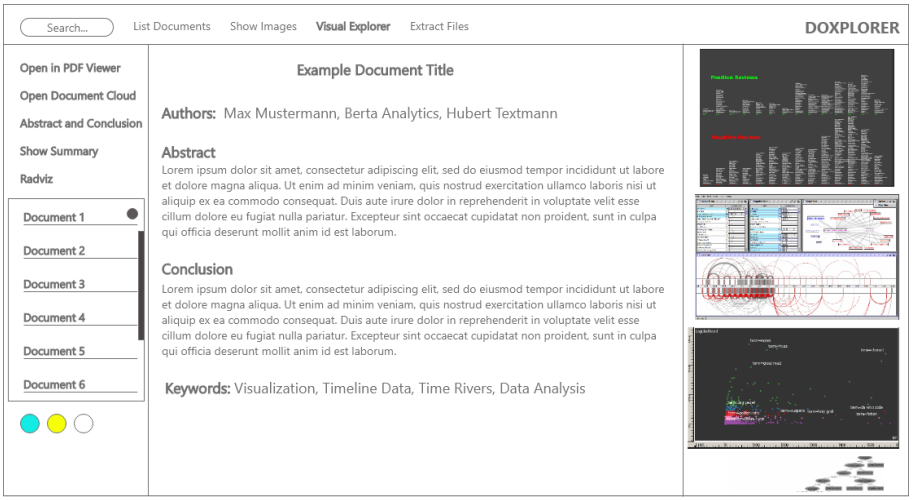
3.3.2 Inspect Documents

Another way to find similar documents is provided by the *InspectView*, which can be accessed from various other views (e.g., *VisualExplorerView* and *SearchView*). Figure 3.4 shows a design mock-up for the *InspectView*. The view contains different sub-panels to provide a concise overview of the different contents from the selected document.

First, there is a summary sub-panel in which summary reports with a predefined sentence number can be generated. The user has the option to choose the number of sentences for the final summary report and assign special weights to terms that are more relevant to him/her. Before the summary report creation, word statistics are presented to aid the user in assigning special weights to terms.

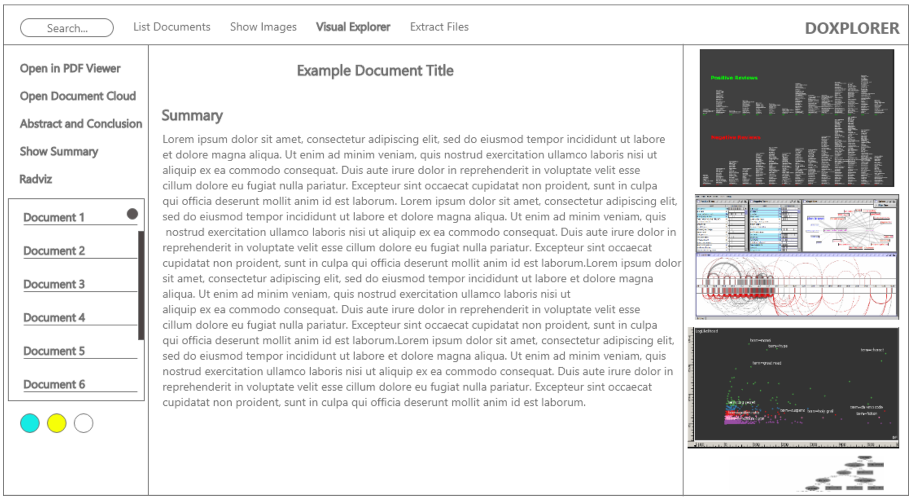
Second, a word cloud sub-panel provides the user with a visual representation of key terms for a specific document. The word cloud is solely term frequency based to provide a different perspective compared to the tf-idf based word statistics. In addition, a separate abstract and conclusion panel gives an overview of the core contents of a document by showing the authors, title, abstract and conclusion (if present). In addition, a list containing all images from the document is displayed at all times within this view.

After inspecting a document with the *InspectView*, further text-based search queries or image searches can be conducted in order to find similar documents with the newly gained knowledge.



(a) mock-up for concept of *InspectView* showing the abstract and conclusion panel.

(b) mock-up for concept of *InspectView* showing the tag cloud panel.



(c) mock-up for concept of *InspectView* showing a summary report of the document.

Figure 3.4: Different sub-panels of the *InspectView*. Previously tagged documents are shown in a list view on the left.

3.4 Similar Documents and Subset Exploration with RadVizDoc

Another option to find similar documents with our system is by using the *RadVizDocView*. Initially, this view was integrated in the *InspectView*. However, since the *RadVizDoc* visualization is applied to the entire subset of interesting documents rather than a single document, we decided to implement it in a separate view called *RadVizDocView*. This view utilizes a visualization technique that is inspired by the traditional RadViz layout which is a radial projection method [30]. Generally, in this visualization anchor points (representing dimensions) are positioned in a circular order and "pull" data points by using Hooke's Law of physics which allows for multi-dimensional data points to be mapped onto a plane [27]. Figure 3.5 shows the basic RadViz layout, with S_1 to S_5 as anchors depicting dimensions and point u representing a single data point being pulled by the anchor springs. However, in our visualization anchor points do not depict single dimensions but specific data items/queries. An example for such a data item would be an image or a specific text query.

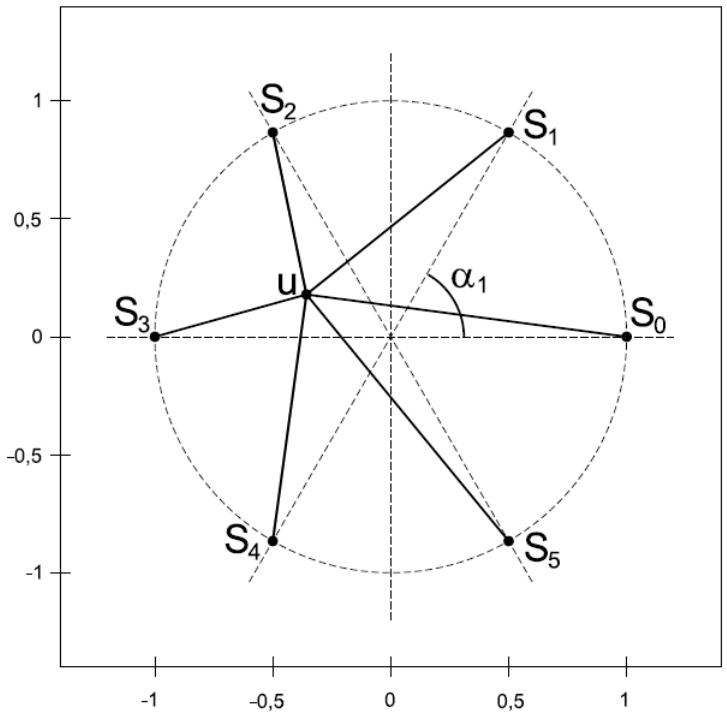


Figure 3.5: Definition of a basic RadViz layout taken from [27]. Anchor points S are positioned in a radial layout, pulling data point u with the attached strings by using their previously assigned forces.

3.4.1 RadVizDoc

In Figure 3.6, a draft illustrates the initial idea for the *RadVizDoc* visualization method. The visualization is supposed to provide the user with an interactive way to query the system and provide visual feedback on how queries are related to a subset of previously tagged documents. *RadVizDoc* enables the user to combine and use multiple queries at the same time on a given subset and visualize the results. For this objective, the visualization relies on three different objects. Those objects are called ***rings, anchors*** and ***elements***.

- ***Rings:*** There are four different types of rings; namely, image-rings, text-search-rings, author-rings and summary-rings. Anchors can be created and attached to rings. However, anchors must have the same data type as their parent ring, otherwise they can not be attached. Existing rings can be rotated, activated, deactivated and deleted.

- ***Anchors:*** An anchor basically represents a single query to the subset of interesting documents. There are four different query types which an anchor can represent. Those types are the same as for rings. Anchors can be freely moved around their parent ring. Of course, they can also be removed from a ring.

- ***Elements:*** Elements represent the documents from the subset and each element depicts a single document. At the beginning, elements are spawned on the inside of the most inner circle. When the querying process is initiated, the elements are pulled by the anchors of the rings. Hovering over an element displays the title of the document it is associated with. Moreover, it shows connections from the element to all active image anchors along with their forces since image anchors can be harder to interpret than other types of anchors.

When rings are rotated or anchors are moved, the elements get repositioned. The way that the elements travel is visually indicated by path lines. Rings in combination with their anchors describe the current configuration of the visualization. The subset can be stored together with different configurations as part of a history function. In order to do so, the user can create a *"history"* and take different visualization snapshots which can be saved and loaded at a later time. It is also possible to use rings of different data types at the same time. This, for instance, provides a way to combine full-text and author searches. The user also has the option to select how many anchors can simultaneously pull on a single element. For example, the user can set the option that only the top three highest scoring image anchors of an element influence it. Of course, the top three anchors differ for each element. This feature can provide useful insights for the subset and can help to separate the subset into different categories or clusters.

3.4.2 How to Use RadVizDoc for Finding Similar Documents

Generally, there are different ways how to find similar documents with this interactive visualization. One can, for example, select all documents from a certain year as subset. This subset then can be investigated within the *RadVizDocView* visualization. The user has the possibility to group different anchors together to find similar documents. Elements which are drawn to the same anchors have a good chance of being related with one another. Another example would be to use a summary or images from a certain document as anchors. Thus, elements drawn to those anchors can be considered similar to the document from which the information was taken.

3.5 Finding Relations

Relations among documents within subsets can also be found by using the *RadVizDocView*. By placing different anchors and starting the query process one can find relations among documents by evaluating the positions of the elements and how they are repositioned when certain configuration parameters are changed. Additionally, similar configurations can be used on different subsets to compare them. By limiting the amount of anchors that simultaneously pull on elements it is also possible to divide the subset of documents in different content categories. For example, if elements are set to be only pulled by the top two highest scoring search queries, every element ends up on a line between two anchors. Consequentially, documents which are on the same line can be considered as being related with respect to the existing queries. This means that we can visualize similarity categories or clusters. Elements which stay at the origin might be related in other ways but have nothing in common with anchors in the current setting.

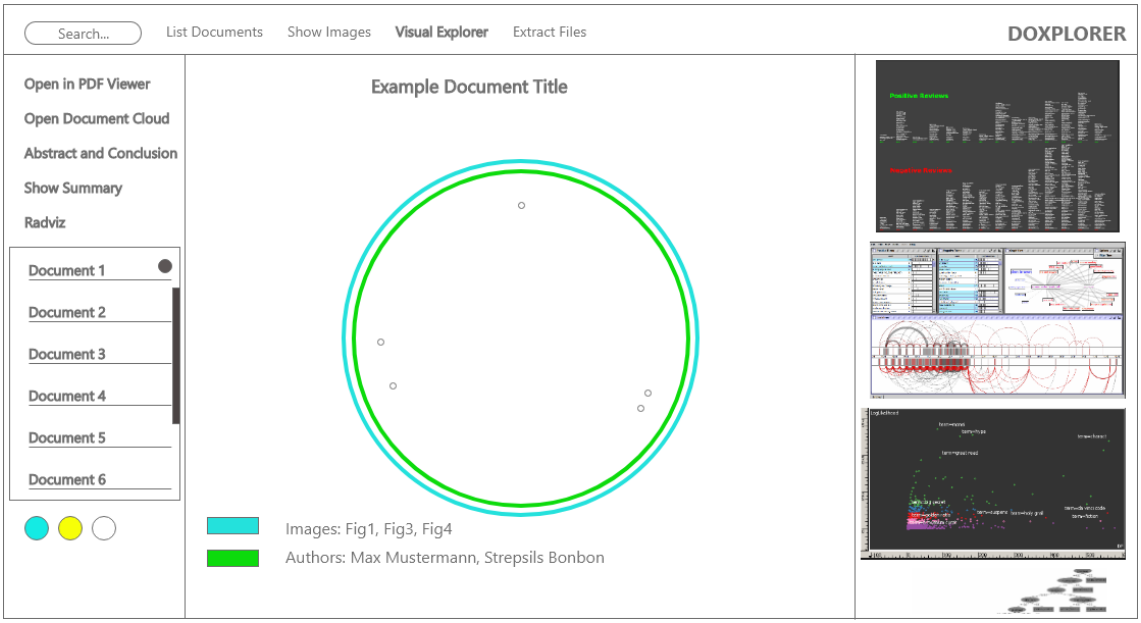


Figure 3.6: Intermediate mock-up of the basic *RadVizDoc* visualization idea.

4 Implementation

Contents

4.1 Overview

This chapter is devoted to explaining the tools and algorithms that were used to realize the concepts described in Chapter 3. On top of that, screenshots of the implemented system are presented to provide the reader with a better understanding. Since the focus of this thesis lies on the combination of various tools, rather than the tools themselves, some tools are realized by basic methods and algorithms in order to stay within the scope of the thesis.

4.2 Java and Graphical User Interfaces with JavaFX

Due to the support of different Natural Language Processing (NLP) libraries and various frameworks, such as DKPro[1], Java[2] is a good choice for building a document retrieval system. Furthermore, Java is a compiled language which makes it faster compared to interpreted languages such as Python[3]. This is an additional advantage especially for digital libraries that deal with large data collections where performance is crucial. According to their official website, Java is the number one programming language used by millions of developers worldwide. Moreover, Java provides very sophisticated libraries and frameworks for implementing Graphical User Interfaces (GUIs). Consequently, Java promised to be a good choice for the implementation part of this thesis.

JavaFX[4] is an extensive framework for creating modern and rich *GUIs* for Java applications and was used for implementing the graphical interface of the system. The framework supports a variety of different graphical components (e.g., scroll-panes, buttons, radio-buttons, split-panes) and allows the installation of different community developed libraries and other frameworks. One of those libraries, which was also used for this thesis, is JFoenix[5] which provides components in material design for JavaFX applications.

4.3 Document Information Extraction

For this thesis we used the VAST paper collection [17] from past years (2006-2012, excluding 2011) as document corpus. The first step was to extract all the necessary data from each document. In order to extract text, title, year, authors and images of each Portable Document Format (PDF) paper, we used Apache PDFBox[6]. Apache PDFBox is an open source library that enables data extraction, manipulation and creation of *PDF* documents. Since *PDF* documents can differ strongly in format and may have corrupt formats depending on how they were created, in certain cases the image extraction can sometimes be faulty. Thus, detected images below a certain size (200 pixels in height or width) are discarded to keep the number of faulty images at a minimum. The extracted texts are automatically stored in plain text format as text files and images are stored in Portable Network Graphic (PNG) format in a separate folder.

[1] `https://dkpro.github.io/dkpro-core/`
[2] `https://www.oracle.com/java/`
[3] `https://www.python.org/`
[4] `https://openjfx.io/index.html`
[5] `http://www.jfoenix.com/`
[6] `https://pdfbox.apache.org/`

4.4 Indexing and Searching of Texts and Images

In order to efficiently search for texts and images, the system uses different libraries that allow us to index the extracted data. The following two sections describe how indexing was implemented within the system.

4.4.1 Text Search and Indexing

Querying a document corpus should be fast and efficient. Hence, searching through each text file separately when a text query is issued is not sufficient. For this purpose sophisticated and extensive libraries, such as Apache Lucene Core[7], were developed. Lucene is a well known search library that, according to their official website, is even used by sites like Wikipedia or Twitter. Of course, many other applications use Lucene Core as well. It is easy to implement, has an extensive documentation and provides a lot of features. Lucene uses inverted indexes and provides different compression modes for fast and efficient data storage. As a result, using Lucene Core for indexing and searching texts was a very promising option.
Therefore, after text extraction we use Lucene to index the texts along with other meta data that was extracted by using PDFBox. In order to provide the system with as much meta data as possible, we additionally used the extracted meta data from [17] for our data set. Lucene is able to store different fields when writing an index and the user can issue queries to specified fields. Hence, Doxplorer uses five different fields, namely, content fields, author fields, title fields, document title fields and date fields. Author fields store the authors' names of a document, while title fields and document title fields are concerned with the file names and document titles. Date fields are responsible for storing the year of publication of documents.

4.4.2 Image Search and Indexing

When dealing with document collections, it is especially important for the image comparison to not take too much time. For this reason, we decided to use LIRe[8] (Lucene Image Retrieval) for Content Based Image Retrieval (CBIR) within the system. LIRe is a widely used open source Java library for image retrieval based on Lucene. Since it uses a light weight text search engine there is no need for a dedicated database server which makes it easy to implement LIRe in applications, especially in prototype applications [21]. The performance and ease of integration make LIRe to one of the best *CBIR* libraries for Java. Therefore, LIRe is used by many enterprise applications and according to their official website:

[7] `https://lucene.apache.org/`
[8] `http://www.lire-project.net`

"LIRE is successfully used at the WIPO, a United Nations Agency, to search in millions of trademark images and the Danish National Police to find similar scenes and to detect near duplicates" [20].

The library uses Lucene indexes for storing image features to enable fast and efficient content based image retrieval. LIRe offers a variety of state of the art image descriptors, for example, Fuzzy Color and Texture Histogram (FCTH), MPEG-7 (color layout, edge histogram, scalable color), Tamura texture features, Color and Edge Directivity Descriptor (CEDD) and more as described by M. Lux et al. [21]. After trying various descriptors, we decided to use *FCTH* for our system. Note, that depending on the domain and visual information, other descriptors could perform better on other data sets. As described by Chatzichristofis and Boutalis [5], *FCTH* descriptors are low level features limited to 72 bytes that combine texture and color information of an image in a single histogram. The limitation in bytes makes *FCTH* descriptors well suited for large image collections as described by Chatzichristofis and Boutalis.

4.5 Text-Based Similar Document Search

Initially, we implemented this part by creating term frequency vectors of documents and calculated the cosine angles in order to receive similarity scores. After the calculation, the ten most similar documents to a given query document were highlighted in the *ExplorerView*. However, this implementation did not aid the exploration process the way we hoped. This method had a too strong emphasis on writing style as compared to semantic content. Thus, we came up with new ideas on how to find similar documents which led to the current implementation. At the start, the system calculates the tf-idf scores for all terms of the given query document. Afterwards, it takes the ten highest scoring terms which are then passed on to Lucene as a standard text query. Then, we take the top five hits returned by Lucene and highlight them in the *ExplorerView*. This implementation has more focus on core information which helps to find documents with similar content.

4.6 Document Summary Creation

As described in Chapter 2, there are many different types of algorithms and approaches for creating summaries. After researching different algorithms we decided to use SumBasic as described in [47] because summaries are only a small part of our system and not the main focus. This decision was made because SumBasic has a good performance due to its simplicity while it still provides good results.

L. Vanderwerde et al. [47] created enhanced versions of the SumBasic algorithm with sentence simplification and lexical expansion; yet, the Recall-Oriented

Understudy for Gisting Evaluation (ROUGE) results they achieved on the Document Understanding Conference (DUC)[9] data sets of 2005/2006 did not significantly outperform the basic algorithm (5% in one case). Nevertheless, according to their paper, the submitted system was ranked first against 21 other systems when comparing user summaries against system summaries. Furthermore, they competed against 34 systems and made the third place for the topic receptivity of their system [47].
Those results also indicate how well the basic version of the algorithm already works compared to other systems. T. Uçka et al. [46] compared state of the art summarization methods with their own system, where we can also see the performance of the SumBasic algorithm as shown in Figure 4.1.

ROUGE evaluation methods		Summary for 200-words (average) (DUC-2002)						
		Methods						
		Random	Luhn [17]	LSA [49,50]	TextRank [52,51]	LexRank [13]	SumBasic [53]	KLSum [54]
Rouge-1	Rec.	0.40932	0.47503	0.37723	0.48417	0.48124	0.46128	0.37464
	Prec.	0.38148	0.44489	0.37012	0.43685	0.45984	0.45114	0.36803
	F-Sco.	0.39470	0.45924	0.37342	0.45868	0.46997	0.45597	0.37104
Rouge-2	Rec.	0.10562	0.20820	0.08950	0.16921	0.18507	0.15947	0.12113
	Prec.	0.09868	0.19558	0.08910	0.15195	0.17566	0.15584	0.11938
	F-Sco.	0.10199	0.20160	0.08921	0.16000	0.18010	0.15757	0.12018
Rouge-L	Rec.	0.36328	0.44490	0.33477	0.46748	0.43813	0.39661	0.34066
	Prec.	0.33880	0.41687	0.32892	0.42183	0.41867	0.38768	0.33470
	F-Sco.	0.35043	0.43022	0.33162	0.44289	0.42787	0.39194	0.33741
Rouge-W-1.2	Rec.	0.12324	0.15858	0.11295	0.15655	0.15201	0.13782	0.11839
	Prec.	0.18074	0.23417	0.17475	0.22316	0.22825	0.21231	0.18321
	F-Sco.	0.14647	0.18902	0.13715	0.18382	0.18237	0.16710	0.14375
Rouge-SU*	Rec.	0.15112	0.19901	0.12783	0.21541	0.20321	0.17981	0.13361
	Prec.	0.13124	0.17444	0.12342	0.17468	0.18494	0.17190	0.12913
	F-Sco.	0.14020	0.18556	0.12527	0.19200	0.19309	0.17548	0.13098

Figure 4.1: Comparison of different summary methods on the *DUC* data set of 2002 by T. Uçka and Ali Karcı [46].

4.6.1 SumBase Algorithm

As described in [47], the algorithm has four different steps. Steps 2-4 are repeated until the desired summary length is reached.

1. Compute the probability distribution for each different word w_i of the input text which is done by:
$$p(w_i) = \frac{n}{N}, \tag{4.1}$$
where N is the total number of words contained in an input text, and n is the number of times word w_i occurs.

[9] https://duc.nist.gov/

2. Calculate a relevance score for each sentence S_j with:

$$R(S_j) = \sum_{w_i \in S_j} \frac{p(w_i)}{|\{w_i | w_i \in S_j\}|} \tag{4.2}$$

3. Choose the highest scoring sentence which also includes the word with the highest probability.

4. For each word contained in the previously chosen sentence update its probability with:

$$p_{new}(w_i) = p_{old}(w_i) p_{old}(w_i) \tag{4.3}$$

Before applying the algorithm we use lemmatization in order to map different forms of each word to single tokens. We realize this task with the LanguageTool Lemmatizer provided by the DKPro [10] framework. DKPro[10] provides developers with a collection of different *NLP* software components that can be used together in pipelines. Since DKPro allows the user to create and integrate custom components, we decided to implement the SumBase algorithm as such. The final pipeline consists of four steps:

1. Reading the text file.

2. Segmenting the input text by using the LanguageTool Segmenter from DKPro.

3. Performing lemmatization on all segments.

4. Applying SumBase while using the lemma words as substitutes for the original terms.

Before initiating the summary creation process, the user is presented with a word distribution of the selected document where he/she can assign special weights to terms. Section 4.7 describes how those word distributions are created. This allows the user to alter how the summary report is generated. Furthermore, the user can choose how many sentences the final summary report contains and whether sentence scores are normalized during calculation or not. The final summary report displays the sentences in the same order they appear in the original document. Additionally, the report shows the corresponding sentence score next to each sentence.

4.7 Word Clouds and Word Distributions of Documents or Subsets

Word distributions are displayed in only two cases: **1)** Before creating a summary of a single document in order to modify term weights. **2)** To help the user find key terms of the current subset in the *VisualExplorerView* when pressing the "Show Word Dist."

[10] https://dkpro.github.io/dkpro-core/

button. Words are presented in descending order with respect to their tf-idf weights. The word distribution calculation handles the set of tagged documents like a single document which means we add the tf-idf weights of equal terms from different documents.

We also decided to provide a word cloud generation that is based on Term Frequency (TF) for single documents. This can be of further help to identify topics or terms of a document. This task was implemented by using the Kumo[11] library for Java which provides an extensive, easy to use word cloud creation API. Before creating the word cloud, we remove basic stop words from the term list. Afterwards, we use Kumo to generate a word cloud and save the result as an image file. The image file is then displayed within the Doxplorer system.

4.8 Abstract and Conclusion Extraction

The VAST data set that we are using includes XML file versions of all *PDF* documents. Therefore, we parse the XML files with the Java Document Object Model (DOM) parser in order to extract and show the abstracts and conclusions of documents if the respective tags are present. In case there is no abstract or conclusion tag contained in the XML file, a message is shown instead which informs the user about the absence of the specific text passage.

4.9 RadVizDoc

The next two sections are about how the different components of the *RadVizDoc* visualization were implemented. Furthermore, the implementation of the general algorithm and component interaction is described. The third section provides a description of the history function implementation. All graphical and visual parts of this visualization were realized by using basic JavaFX components.

4.9.1 Components

As previously mentioned in Chapter 3, there are four different types of rings and anchors. First, we are going to have a look at the different rings. Basically, rings can be created or deleted at any time while using the visualization. When creating a new ring its radius is always larger than the radius of already existing rings. Consequentially, rings are layered on one another. Additionally, all rings have the origin of the visualization pane as center. The visualization origin (0,0) denotes the center of the visualization pane if not stated otherwise. Each ring contains a list of all anchors attached to it. An image ring, for instance, can only have image anchors attached to it. Furthermore, a ring comprises a unique numeric identifier and a graphical circle which is used for visualizing the ring. A

[11] https://github.com/kennycason/kumo

ring keeps track of its identifier and radius which is changed when inner rings are deleted. Associated anchors can be moved freely on the respective ring by dragging the anchors with the mouse. Additionally, rings can be rotated which also repositions the anchors of the rotated ring. Further, it is possible to distribute the anchors around their parent rings by using the "Distribute Points" button. This function uses the following formula for anchor distribution:

$$R_i = i \cdot \frac{360}{N} \quad \text{for} \quad i = 0, ..., |C_j| \tag{4.4}$$

where:

1. R_i represents how many degrees anchor i needs to be rotated around the center of its parent ring.

2. C_j denotes the anchor set for a specific ring.

3. N depicts the maximum number of anchors any ring has in the current configuration with $N = \max\{|C_j| : C_j \in V\}$ where V denotes the set of rings.

Furthermore, rings can be activated or deactivated, which influences if the anchors of a ring are taken into account when positioning elements. Naturally, rings can also be deleted if they are not needed any longer. If a ring is deleted, all associated anchors are removed as well. However, a ring has to be selected in order to perform those interactions. At any time there is always at most one *"currently selected"* ring. Users can select a ring by simply clicking on it. The *"currently selected"* ring is always visually highlighted. Further, rings of the same type also share the same color.

Next, we are going to explain anchors in detail. The specific implementation differs slightly depending on the anchor type. However, for the most part the distinct anchors are implemented in a similar way. Basically, an anchor includes a graphical circle, a reference to the parent ring and an event handler. The last important information held by an anchor is the associated query data which is used for pulling the elements when computing their positions later. This query data can be of type string (for summaries, authors and queries) or *FCTH* (for images). The event handler ensures that the anchor always sticks to its parent ring when being dragged by the user. This is achieved by calculating

$$x_i = x \cdot \frac{r}{\sqrt{x^2 + y^2}} \tag{4.5}$$

and

$$y_i = y \cdot \frac{r}{\sqrt{x^2 + y^2}} \tag{4.6}$$

where:

1. x and y denote the current mouse coordinates.

2. x_i and y_i are the new coordinates of anchor i.

3. r is the radius of the parent ring.

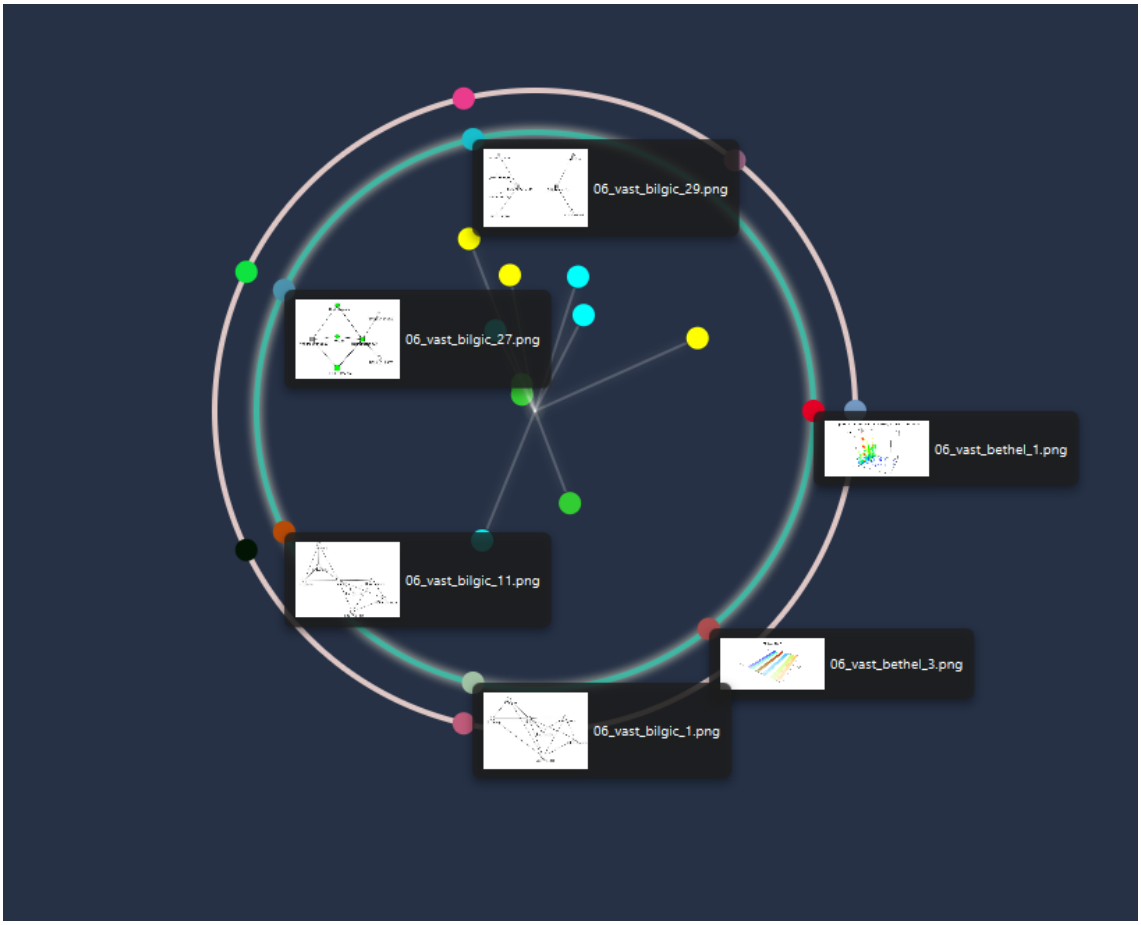


Figure 4.2: *RadVizDoc* showing images of image anchors.

Author anchors and text search anchors can be created by pressing the "Add Point" button. The user is requested to enter author names or search terms depending on the selected anchor type. However, summary anchors and image anchors are created in a different way. Depending on the currently selected ring, the system displays either images and relevant terms of tagged documents or previously created summary reports in a list view. By clicking on summaries, terms or images, the user can generate different anchors. Note, that anchors for search terms are the only ones that can be created in both ways. When an author ring is selected, the user still has the option to create search term anchors, image anchors or summary anchors in this way, depending on which type of ring was selected last. The system displays query information (texts or images) of anchors when the user hovers over them. Additionally, the user can display

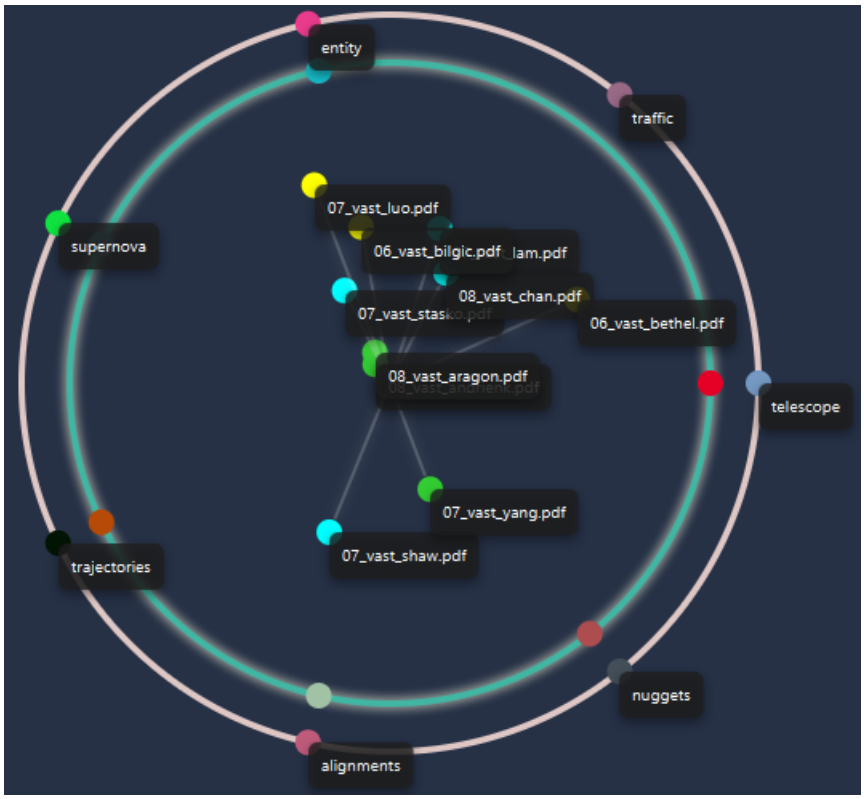


Figure 4.3: *RadVizDoc* showing texts of search query anchors.

this information for all anchors at the same time, for example, by pressing the "Images" button to show all anchor images (see Fig. 4.2 and Fig. 4.3).

Elements are the last major component used for the *RadVizDoc* visualization. Each element in the visualization represents a single document contained in the subset of tagged documents. Elements hold the most information out of all three main components including:

- A graphical circle element representing the document within the visualization.
- A reference to the indexed document itself.
- Four different maps for each anchor type (author, summary, image, search query). The maps store pairs of vectors and influences. Each vector describes the position of a single anchor in relation to the origin/center of the visualization. The influence value of a vector describes how strong an anchor "pulls" on a specific element.

- A color reference. Since the user can tag documents as important for different reasons they can be marked in three different colors. Furthermore, the user can assign a special color to elements in order to track them during the visualization.

The elements can not be directly repositioned by the user since the position of elements is calculated by the algorithm and depends on the configuration of the rings and anchors. However, elements can be marked with a different color in order to track them when changing the ring and anchor configuration. In addition, the user can toggle tooltips that display the document titles for all elements by pressing the "Elements" button (see Figure 4.3).

4.9.2 Algorithm

As an overview, the algorithm can be described in three steps:

1. Spawning elements representing tagged documents at the origin.
2. Calculating vectors from elements to anchors and their respective influences.
3. Moving elements to calculated positions according to vectors and their influences.

Before initiating the algorithm, a ring and anchor configuration has to be created. After establishing a satisfying configuration, the user can select the number of top scoring anchors simultaneously pulling on elements. Moreover, the user can set an *"average image comparison"* option for the algorithm. This option is turned off by default. A detailed explanation of what this option changes is discussed later in this section.

Now, let us see how the algorithm works. At first, the algorithm spawns an element for each tagged document in its respective color (blue, yellow, green). Subsequently, the algorithm iterates over all elements to calculate their correct positions within the visualization for the given configuration. Therefore, for each element we iterate over all rings of a certain type, for example, image rings and their corresponding anchors. Then, an influence vector pair is created specifically for each anchor. Each of those vectors describes the direction from the origin to a specific anchor. Since each ring has a different radius to support clarity within the visualization, the vectors of anchors of distinct rings have different lengths. Thus, all vectors are adjusted to have the same length. The influence calculation for a specific anchor naturally depends on its type. Since the user is provided with different configuration options for the algorithm, the influence calculation varies based on them. For image vectors the default influence calculation looks as follows:

$$I_{ij} = 1 - \frac{d_{ij} - min_{ik}}{max_{ik} - min_{ik}} \tag{4.7}$$

where:

1. I_{ij} is the influence of image anchor j for element i.
2. d_{ij} is the Euclidean distance of the FCTH image feature vectors of anchor j and element i.
3. min_{ik} is the minimum Euclidean distance out of all image anchors for a certain element i given by: $min_{ik} = \min_{k' \in A} d_{ik'}$, where A is the set of all image anchors.
4. max_{ik} is the maximum Euclidean distance out of all image anchors for a certain element i given by: $max_{ik} = \max_{k' \in A} d_{ik'}$, where A is the set of all image anchors.

Note, that a document can comprise more than one image. Thus, in the default version d_{ij} describes the smallest Euclidean distance found when comparing every image of a document with the anchor image. In case that *"average image comparison"* is selected, the algorithm does not take the smallest distance found but averages all image distances of a single document to a specific image anchor. The rest of the formula stays the same. Generally, the influence formula for the other types of rings is similar with:

$$T_{ij} = \frac{r_{ij} - min_{ik}}{max_{ik} - min_{ik}}$$

where

1. T_{ij} is the influence of a text-based anchor j for element i.
2. r_{ij} is a specific relevance function for an anchor j and element i and depends on the type (summary, author or query).
3. min_{ik} is the lowest relevance score out of all anchors of the same type for a certain element i given by: $min_{ik} = \min_{k' \in A} r_{ik'}$, where A is the set of all anchors of a certain type.
4. max_{ik} is the highest relevance score of out of all anchors of the same type for a certain element i given by: $max_{ik} = \max_{k' \in A} r_{ik'}$, where A is the set of all anchors of a certain type.

As we can see, the only difference compared to image rings is that we have a relevance function instead of a distance function, which, of course, needs to be taken into account in the influence formula. The reason for this distinction is that we use LIRe for fast state of the art image retrieval, which always returns distance scores between the query image and other images. However, text-based queries are handled by Lucene which returns relevance scores instead of distances. Now, let us have a look at the distance and relevance functions for the different anchor types.

- Image anchor: Here we have two options, as described before: (1) Smallest Euclidean distance found between query image and all document images. (2) Average Euclidean distance between query image and all document images.

- Summary anchor: Lucene score for an entire summary report that is converted into a query. Lucene calculates the score for the query to a single document which belongs to the corresponding element. Stop words are removed before issuing the summary report to Lucene.
- Author anchor: Authors are indexed in their own field. If this field contains the queried author name the system returns 1, otherwise 0 is returned.
- Text search anchor: Lucene score for a specific query to a single document which belongs to the respective element.

We apply this process to calculate vector influence pairs for each element with all anchor types. Now we need to calculate how to position the elements according to each anchor type. Let $\overrightarrow{A}$ be the anchor vector set of a single element for a certain type of query. Then, with I as the influence set of those anchors we get the vector $\overrightarrow{u}$ for positioning the element by:

$$\overrightarrow{u} = \frac{\sum_i^n \overrightarrow{A_i} I_i}{\sum_i^n I_i}$$

After calculating the position of an element for all anchor types, we need to merge the different positions together in order to obtain the final position. Therefore, depending on the number x of distinct types of activated rings, we need to scalar multiply each vector by $1/x$ before adding them up. This means, for example, that if two different ring types are active each of them has 50% influence on elements. Element positions are always calculated and updated in the following cases:

- A ring was rotated.
- A ring was deactivated/activated.
- The “Reposition Elements” button was pressed.
- The “Spawn Elements” button was pressed.

4.9.3 History and Snapshot Function

The history function stores a snapshot collection of previously created *RadVizDoc* configurations. A history comprises different *“snapshots”* where each snapshot contains a single XML file and an image. The image is a screenshot that shows how the visualization looked during the snapshot creation time. In addition, the XML file contains all information needed to restore that specific state. This information includes all rings, anchors, and documents that were tagged at that time. Furthermore, ring rotations and anchor positions are stored in order to recreate the exact configuration when loading a snapshot. The XML files are created by using the Java *DOM* parser. The user can select different

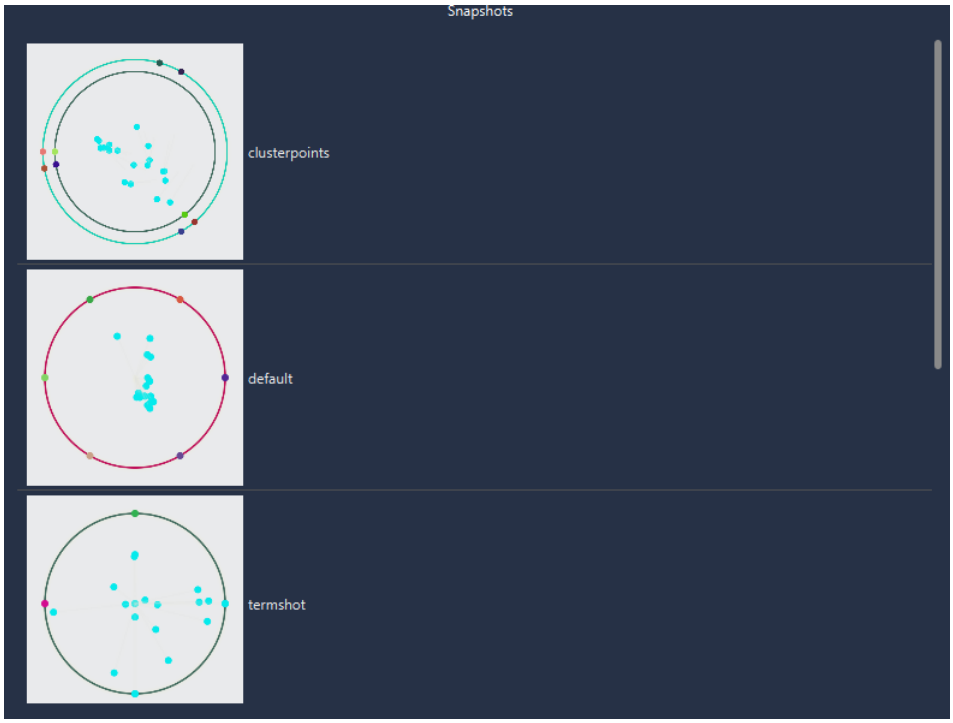


Figure 4.4: Snapshot selection menu after choosing a history.

histories and load a specific snapshots of any history. After selecting a history, the system shows a list which contains all snapshots of that history along with their corresponding images. When the user creates a history, a default snapshot of the current visualization state is created. Afterwards, the user can create new snapshots that are automatically stored for the current history. The user can create a new history at any time. Figure 4.4 shows an example of the snapshot selection menu.

4.10 Final Doxplorer Interface

This section provides figures depicting the final state of the system.

Figure 4.5: *SearchView* of the Doxplorer system.

Figure 4.6: *ImageSearchView* of the Doxplorer system.

Figure 4.7: *VisualExplorerView* of the Doxplorer system.

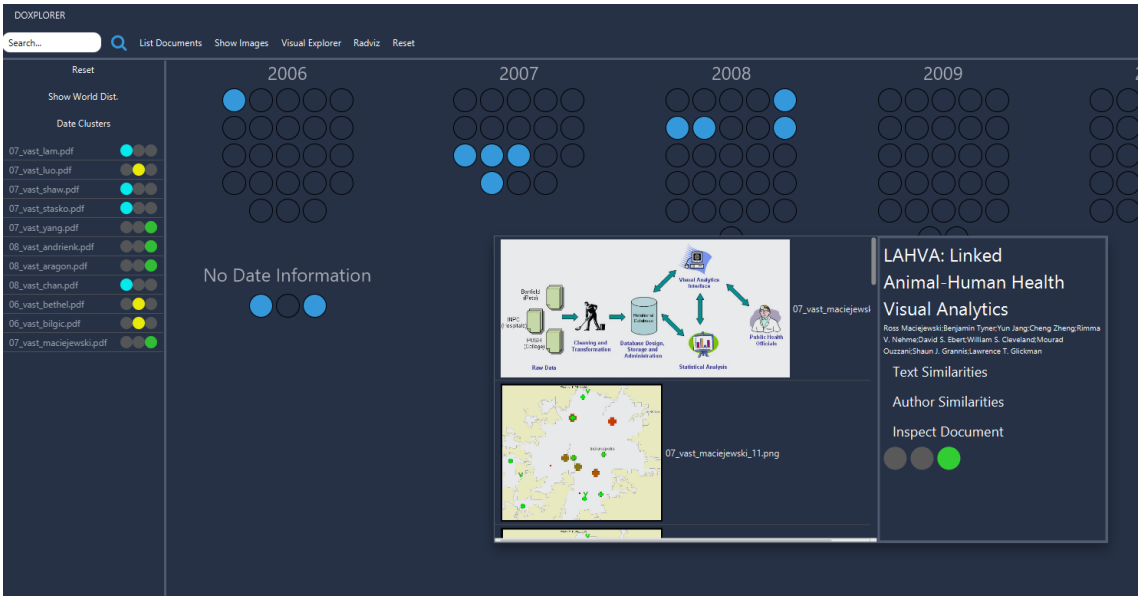


Figure 4.8: *VisualExplorerView* of the Doxplorer system after opening a document card.

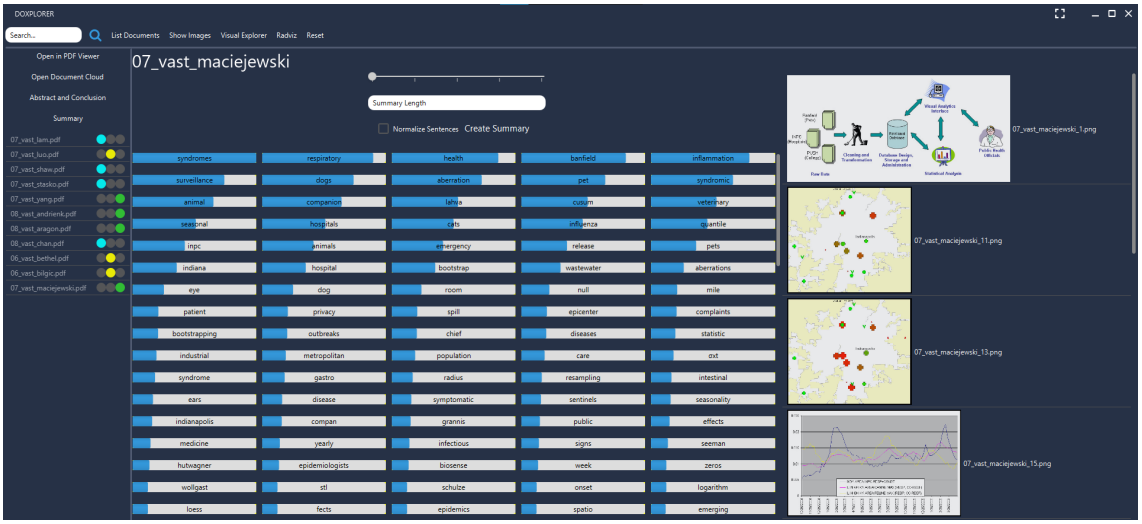


Figure 4.9: *InspectView* of Doxplorer showing summary creation menu.

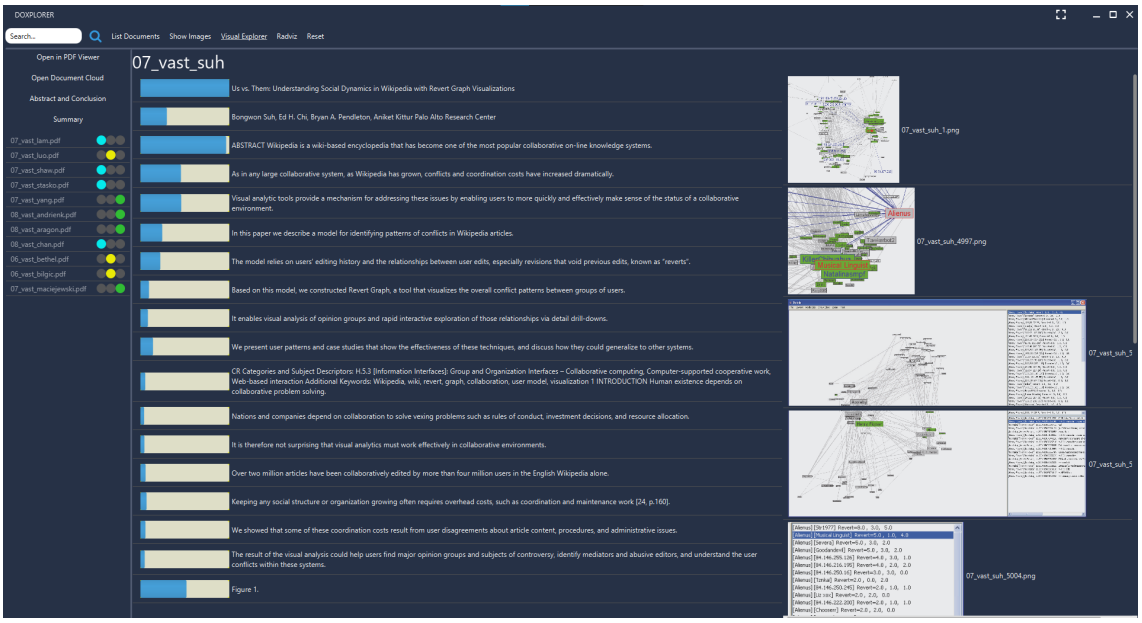


Figure 4.10: *InspectView* of Doxplorer showing generated summary report.

Figure 4.11: *InspectView* of Doxplorer showing a word cloud.

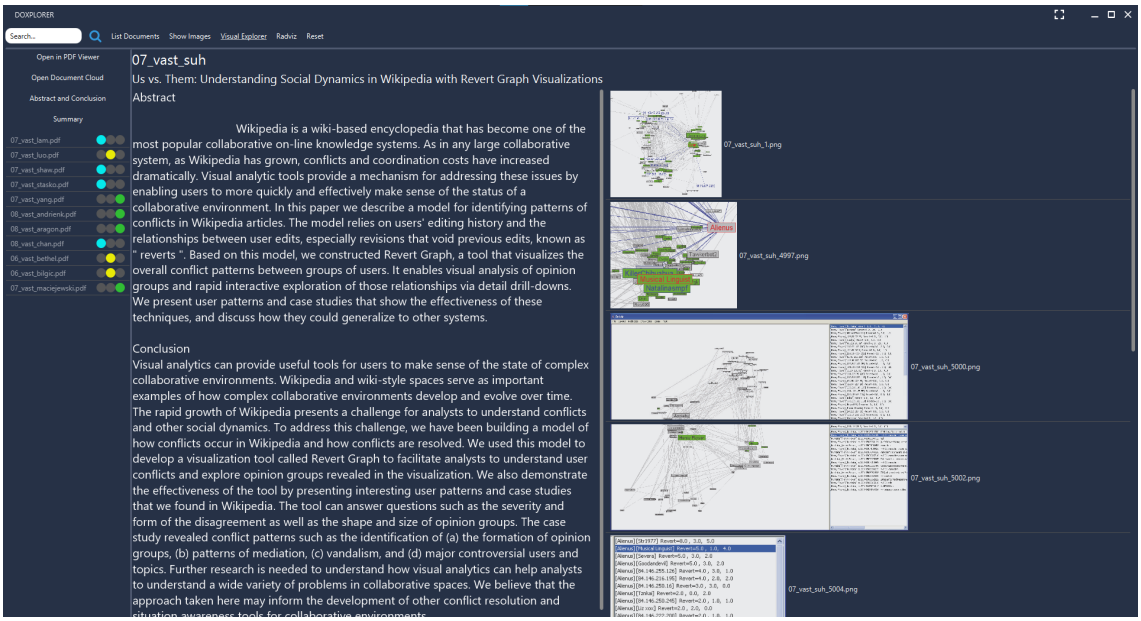


Figure 4.12: *InspectView* of Doxplorer showing abstract and conclusion of a given document.

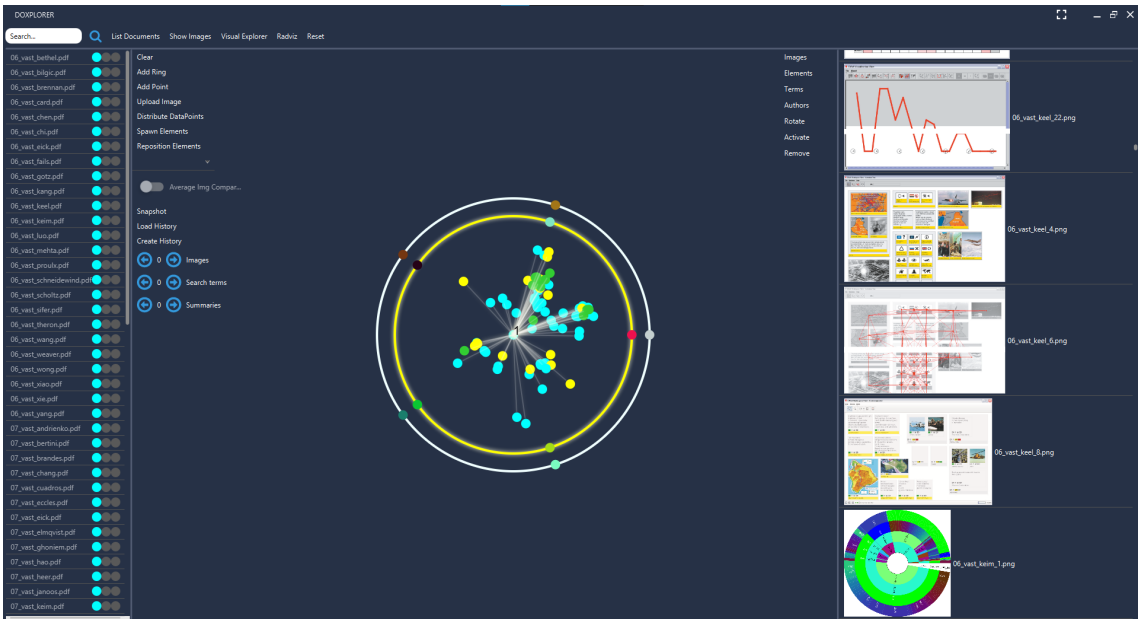


Figure 4.13: *RadVizDocView* of *Doxplorer* with example configuration.

5 Application

Contents

This chapter comprises two different sections. The first section of this chapter is dedicated to illustrate general use cases for *Doxplorer*. Afterwards, application scenarios of *Doxplorer* on the VAST data set are demonstrated and evaluated.

5.1 Use Cases

This section describes different use cases for *Doxplorer*. In total, there are five different use cases which are listed within this section. Note, that the first two use cases are small but crucial for the exploration process. They are also part of the larger use cases following them. In order to avoid too many different use cases, less important ones, such as summary creation, are comprised in the larger use cases which are more general. Before the first use case, we give an overview of the workflow how the different views are indented to work together.

5.1.1 Workflow

First, the *SearchView* and the *ImageSearchView* are used to create an initial subset of interesting documents. Further, the *VisualExplorerView* can be used to find documents that are similar to documents of the current subset by using document cards and similarity searches. The aforementioned similarities can be image-based, text-based, year-based or author-based. Document cards of this view also enable the user to swiftly grasp the core contents of documents. A term statistics view within this visualization also provides the user with the most important terms from the current subset. After generating a subset, the user can investigate it by using the *RadVizDocView*. This view allows the user to find relations among documents and created anchors. Furthermore, it can help the user to cluster or categorize documents according to different queries (which can depict different information). Lastly, the user can use the *InspectView* which can be opened from within any other view. This view has a supporting role and should always be used if the user needs more information about a document in order to decide if it is relevant.

5.1.2 Keyword Search

The user wants to query the system for relevant documents from a given collection by entering search terms. The results should be returned in real time, even for large document collections. Search terms can be entered in form of single words or phrases. Boolean operators, proximity searches, range searches, term boosting and wildcard searches (term modifiers) are supported. At the beginning of the exploration process, this functionality is crucial for finding relevant documents in a collection.

Main Scenario (see Figure 5.1)
The user opens the *SearchView* which shows a list of all documents from the collection. Next, the user enters search terms of interest. In succession, the system returns the top results for the provided query and updates the list. The items are shown in descending order according to their calculated scores. Afterwards, the user can select and mark documents with a color, which adds them to the current subset of interesting documents. Of course, this process can be repeated for as many queries as needed. Queries can be nested, connected with Boolean logic and more. Figure 5.1 lists the different query features provided by Lucene.

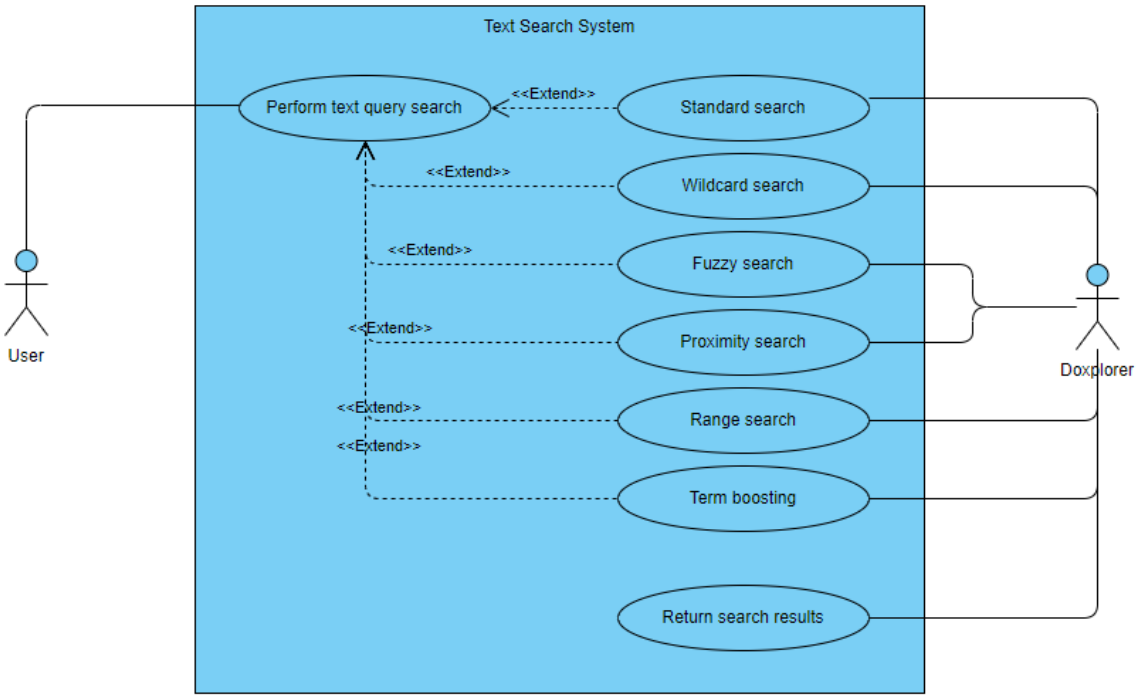


Figure 5.1: Use Case Diagram: Keyword Search.

5.1.3 Image Search

The user wants to query the system for relevant documents of a given collection based on images. Results should be returned in real time, even for large document collections. Search images can be uploaded by the user or selected from the document collection. This functionality is important to find documents that might include visual information that is relevant to the user. Further, this can help the user to find documents that use the same frameworks or interfaces in figures.

Main Scenario

The user opens the *ImageSearchView* which allows him/her to browse all images from the collection. He/she can initiate an image search by clicking on an image or by uploading one. Afterwards, the system updates the list by showing the top results in descending order. Furthermore, the calculated image distances are shown next to the images. In the top left, the user can see the query image. Next, the user can mark documents that contain relevant images within this view which adds them to the subset of interesting documents. Naturally, the user can perform as many image searches as needed.

5.1.4 Finding Similar Documents with "Inspect" and "Search"

The user has a document and wants to search the collection for similar documents. This goal can be achieved by using the *InspectView* combined with the *SearchView* and *ImageSearchView*. The *InspectView* offers a variety of different tools. For instance, the user can generate summary reports, word clouds, word statistics or even extract abstracts and conclusions from documents. These features help to extract knowledge for finding similar documents.

Main Scenario (see Figure 5.2)
The user can open the *InspectView* from any other view. This can be done by selecting a document from any list and pressing the "Inspect Document" button. In the *VisualExplorerView*, the user can also switch to the *InspectView* by opening a document card and pressing the "Inspect Document" button from there. At first, this view displays the summary generation menu or a previously created summary report if there exists one. The summary report can help the user to understand the core contents of the document. Furthermore, in this view, the user can also generate a word cloud, or read the extracted abstract and conclusion from a document if they exist. Lastly, the user has the option to open the full document from within this view, if he/she has a Portable Document Format (PDF) viewer installed. By using these features, the user can gain new insights into document contents. This enables the user to discover relevant key terms, that can be used to find similar documents by using the *SearchView*. Moreover, the *InspectView* also provides the user with all images from the inspected document. This may help to find interesting images which can in turn be issued to the *ImageSearchView* to find documents with similar images. The user can iteratively inspect documents and create queries with the newly acquired knowledge. This helps to further refine the subset of interesting documents.

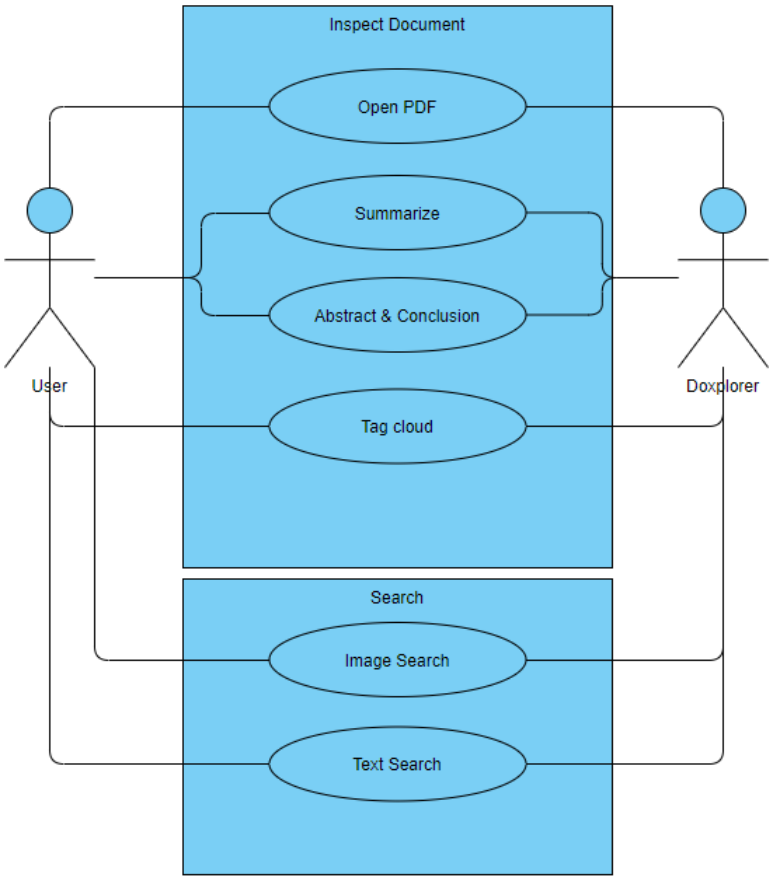


Figure 5.2: Use Case Diagram: Find Similar Documents with Inspect and Search.

5.1.5 Finding Similar Documents with the *VisualExplorerView*

The user wants to find similar documents of a collection by using the *VisualExplorerView* of *Doxplorer*. Of course, the *VisualExplorerView* can switch to the *InspectView* in order to analyze single documents, if needed. The *VisualExplorerView* enables the user to find documents with similar year of publication, images, texts and authors. The combination of the aforementioned tools aids the user in refining the subset of interesting documents even further.

Main Scenario (see Figure 5.3)
After the user has established an initial subset of interesting documents by using the *SearchView* and the *ImageSearchView*, he/she switches to this view in order to find similar documents. If the user is interested in a document, he/she can select the corresponding circle in the visualization and open a document card. From there, the user has the option to perform the use case we described previously. Yet, there are three other options. First,

we have the image similarity search. The user initiates it by selecting an image from a document card. As a result, circles depicting the top documents with similar images are highlighted in yellow. The user has then the option to inspect those documents by opening their document cards. In case the user decides that a document is important, he/she can mark it with the corresponding color in the document card which adds the document to the current subset. The user can also initiate an author search from document cards, where he/she can enter names of authors. Circles of the resulting documents are highlighted in green. The final option is to select the text similarity button to find the top related documents in terms of core content. Then again, resulting documents can be marked as important. This process can be repeated to refine the subset until a satisfactory result is achieved.

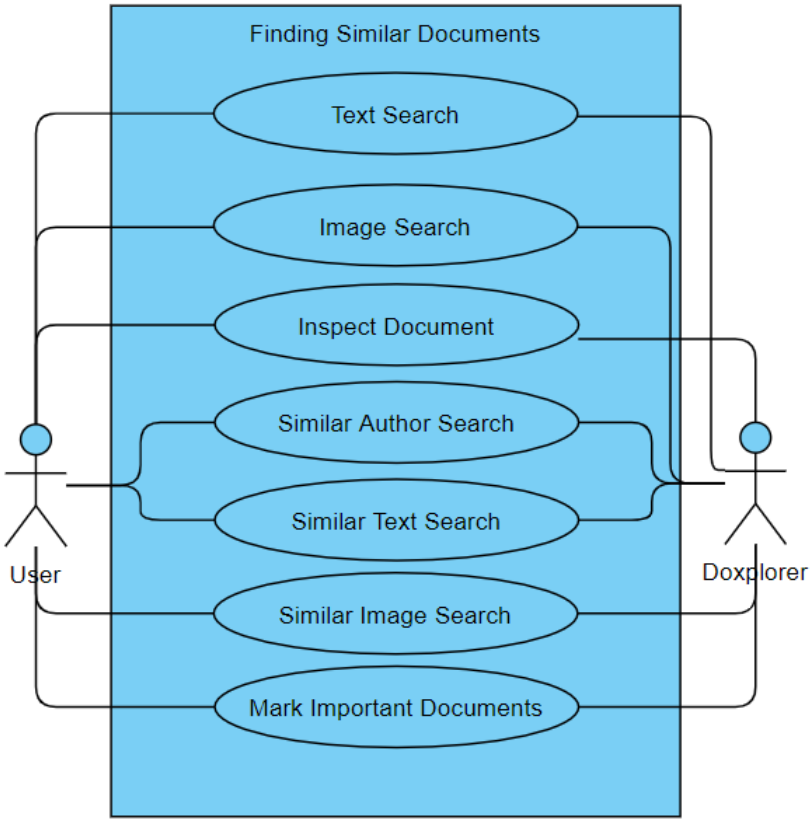


Figure 5.3: Use Case Diagram: Finding Relevant Documents.

5.1.6 Evaluating Subsets with the *RadVizDocView*

The user wants to find additional relations between previously tagged documents. In order to find such similarities, the user can use the *RadVizDocView*. By projecting the documents on a plane as two-dimensional points that are being pulled by anchor data points, the user can observe common behaviors among different documents. Those anchor points can represent different types of data, e.g., images, keywords, authors and summary reports. As a consequence, the user can identify new coherences between documents and anchors. Hence, he/she acquires a better understanding of the selected document subset.

Main Scenario (see Figure 5.4)
The user opens the *RadVizDocView* to investigate a previously created subset of documents. Here, the user has the option to create four different types of rings. The user can add anchors to rings and change the configuration by rotating rings, repositioning anchors, removing anchors, limiting the amount of top anchors simultaneously pulling and activating/deactivating rings. Further, the user has the option to manually highlight certain elements in red which enables him/her to better follow their movement when changing the configuration. The user evaluates the positions of elements according to anchors and further observes how those positions change when the configuration is altered. By doing this, the user can gain new knowledge about document relations within his/her current subset.

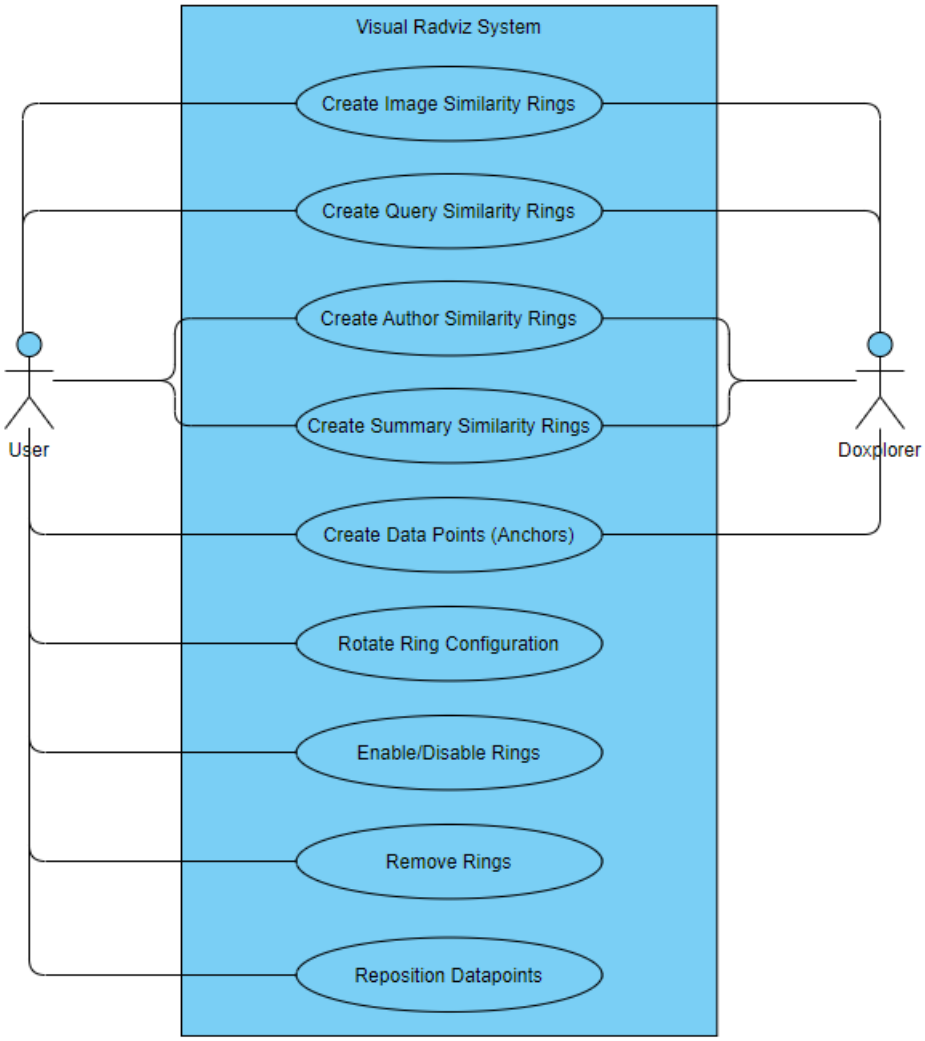


Figure 5.4: Use Case Diagram: *RadVizDoc*.

5.2 Scenarios

The application of *Doxplorer* is demonstrated by using it on example scenarios. All scenarios are conducted by a fictional character called Alice. In each scenario, Alice has a specific starting position and goal in mind which she tries to accomplish by using *Doxplorer*. All scenarios were applied on the combined VAST data sets from the years 2006, 2007, 2008, 2009, 2010 and 2012 which sums up to a total of 153 distinct documents.

Disclaimer

Some of the scenarios show image-based or contextual results retrieved by using the system on the previously defined data set. For pragmatic reasons, we refrain from providing specific references for every paper where the scenario results stem from. All scenario results were retrieved from the IEEE Conference on Visual Analytics Science and Technology (VAST) papers submitted in the years 2006, 2007, 2008, 2009, 2010 and 2012.

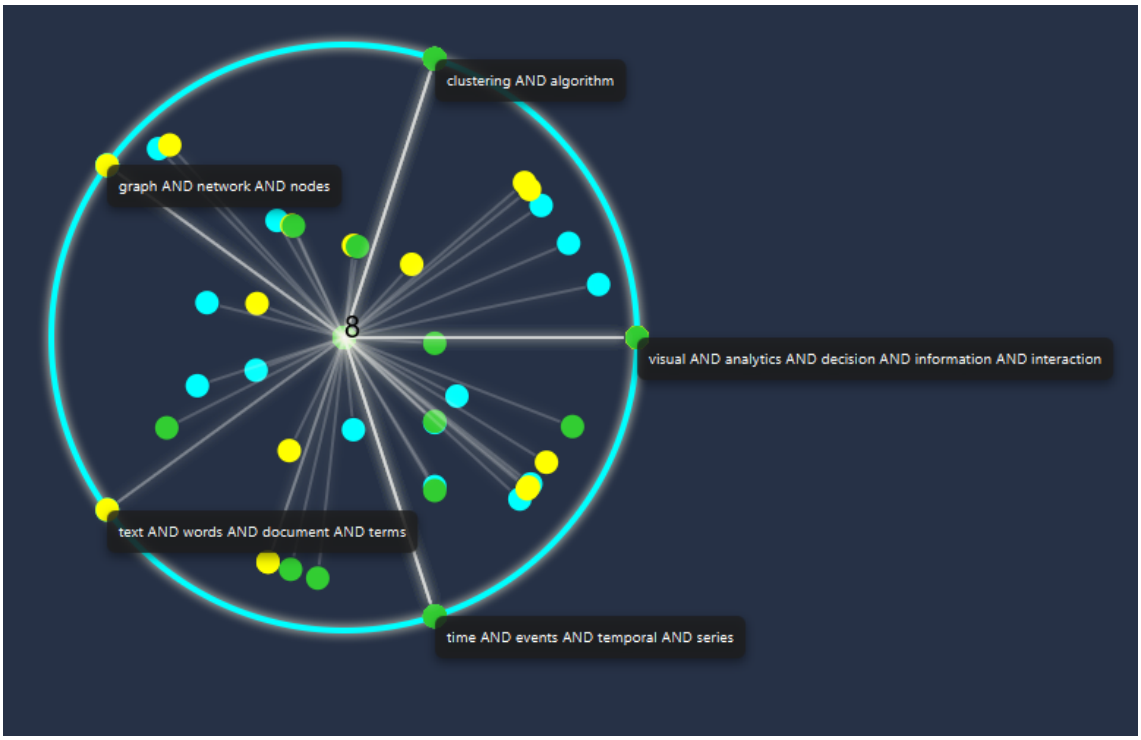
5.2.1 Scenario 1

Alice is interested in different visualization topics and wants to investigate if she can find trends across different years. In this scenario, Alice decides to compare the VAST paper data sets from 2006, 2007 and 2008. Instead of conducting a basic text search, Alice decides to use *RadVizDoc* for the text search to perform multiple queries simultaneously. She uses the coloring system to visually differentiate papers according to their year of publication (2006 blue, 2007 yellow and 2008 green). Alice uses Boolean logic in order to create anchors resembling different topics. Some of the terms Alice comes up with by herself and the rest she finds by using the word statistics and word clouds provided by the other views of Doxplorer. Alice decides on the following five anchors depicting topics:

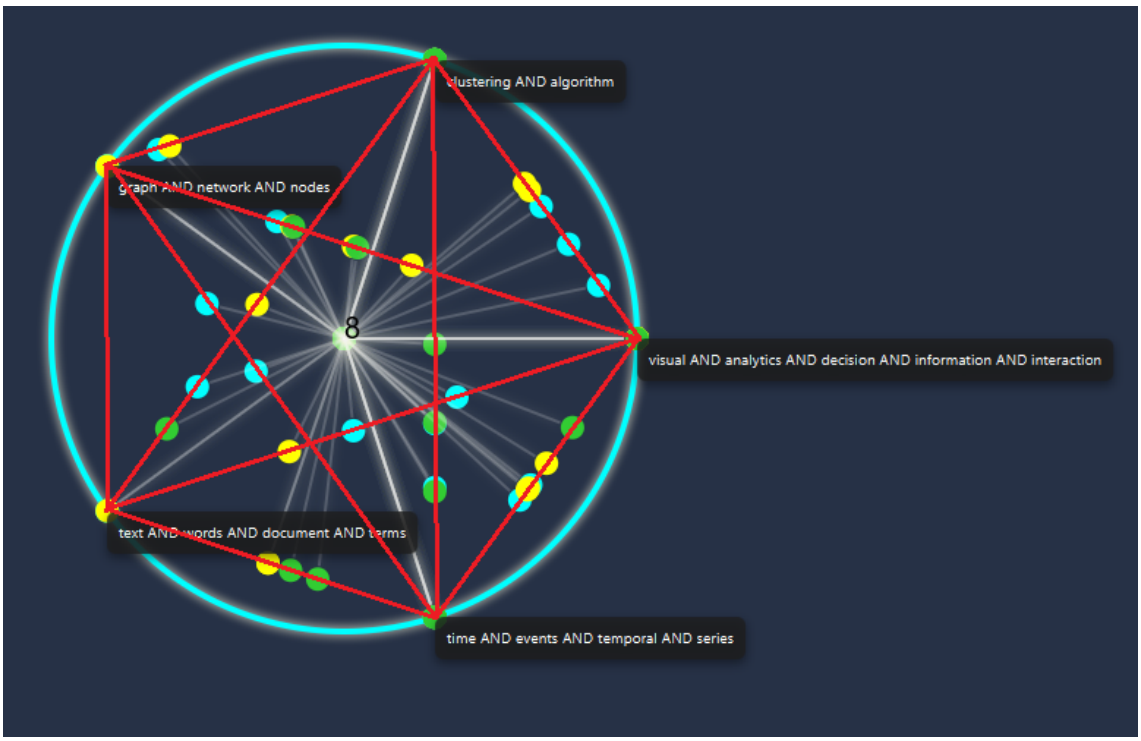
- visual AND analytics AND decision AND information AND interaction
- time AND events AND temporal AND series
- text AND words AND document AND terms
- graph AND network AND nodes
- clustering AND algorithm

By using Boolean logic, Alice ensures that documents that are being pulled by an anchor contain all of its topic key words. Alice finds out that only 13 documents include all terms of three distinct topics simultaneously. Since most of these documents are barely pulled by their third anchor, Alice decides to limit the amount of simultaneously pulling anchors to only two. Figure 5.5 shows the corresponding visualization. In Figure 5.6 we can see the observation Alice previously made (all documents, except for 13 documents, contain at most two of the anchor topics). Documents that are on a line between two distinct anchors can be considered as a cluster formed by two anchors. From the presented visualizations, Alice gains further insights. For instance, by looking at the elements between two different anchors, Alice concludes that some topics of the selected subset seem to be more related than others. For example, on the line between anchor six (*"graph AND network AND nodes"*) and anchor one (*"clustering AND algorithm"*) only two elements are present. In addition, they are very close to one of the anchors and far away from the other one. However, both of these anchors seem to be strongly related with anchor

two (*"visual AND analytics AND decision AND information AND interaction"*). From the visualizations, Alice can also directly tell that eight documents do not relate to her custom topics. At the end, Alice decides to remove the Boolean *"AND"* operators from the anchor queries in order to receive an overview of the general element distribution over her handcrafted topics (see Fig. 5.7). This allows Alice, for example, to find out that the documents from 2006 and 2007 (blue and yellow) tend to be more related to graphs, nodes and networks than VAST documents published in 2008 (green).



(a) Topics with Boolean logic and only two anchors simultaneously pulling. The number in the middle shows how many documents are not related to any anchors. Thus, they remain in the center.



(b) Lines illustrating the anchor connections of the visualization.

Figure 5.5: Example topic relations that Alice found by using the *RadVizDocView*.

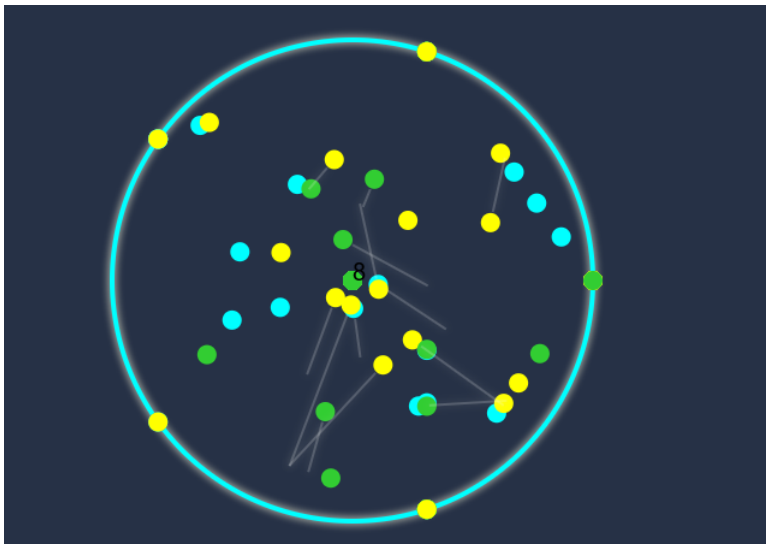


Figure 5.6: Example topic relations that Alice found by using the *RadVizDocView*. Element movement is visualized after allowing all anchors to pull simultaneously.

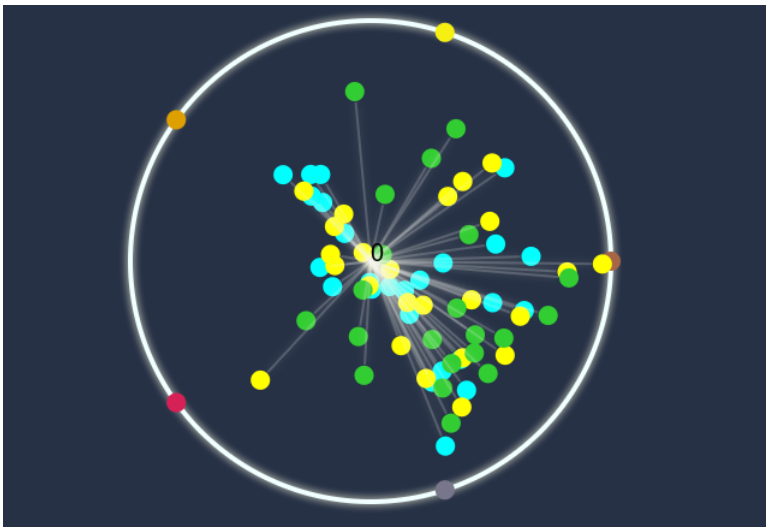


Figure 5.7: Example topic relations that Alice found by using the *RadVizDocView*. Terms are used without Boolean operator *"AND"*.

5.2.2 Scenario 2

After browsing the collection with different search features, Alice ends up with a subset of 36 tagged documents. However, when tagging documents she did not consider visual information. Nonetheless, there are three images Alice is now particularly interested in after inspecting some documents more thoroughly (see Fig. 5.8). One of these documents is about a multi-screen visualization and Alice wants to know if there are other papers with similar visualizations or multi-screen setups. Another document is about visualizing

high-dimensional data. Alice likes the visualization very much since she prefers dark mode visualizations. Hence, this image sparks Alice's interest in finding other dark mode visualizations. Although Alice prefers flashy and visually appealing visualizations, she likes how graph-based charts convey information in a very simple, yet effective, way. Thus, she also wants to find papers that use line-based charts as visualization method to know in which domains this technique is applied. For the aforementioned reasons, Alice decides on the goal to see if there are documents with similar visual information contained within her previously established subset. Further, she wants to know if there are other documents that could be of interest which are currently not contained within the subset.

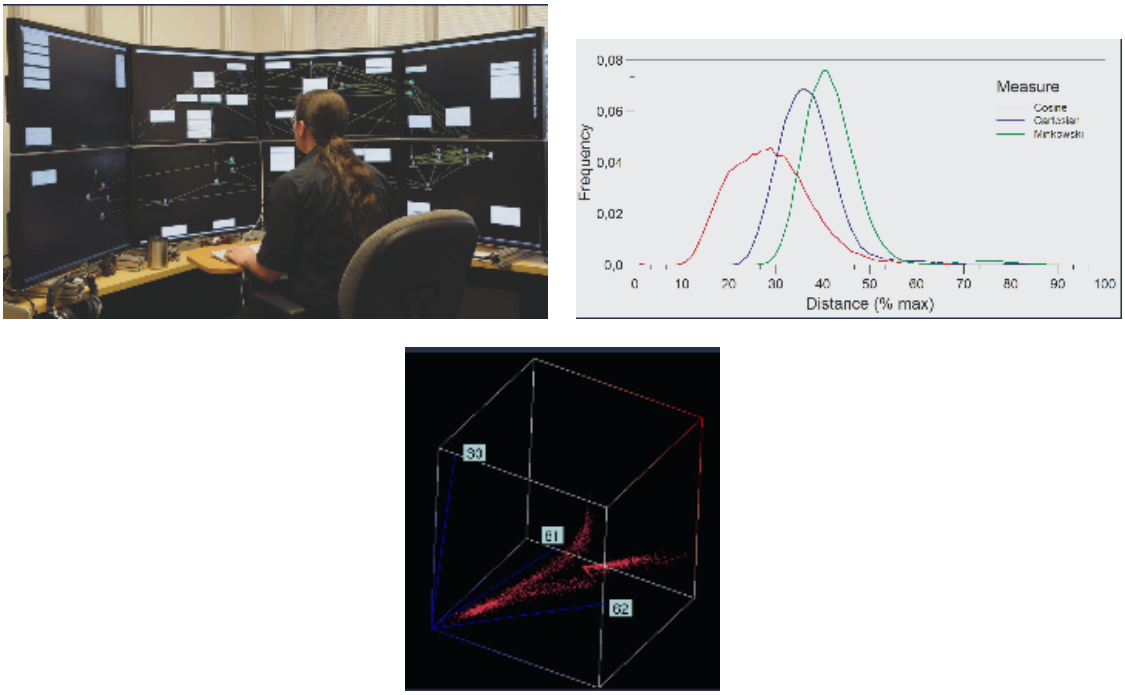


Figure 5.8: Images of interest to Alice taken from the VAST data set.

As a result, Alice starts by using the *RadVizDocView* (see Figure 5.9). When inspecting a few of the elements closest to the most right image anchor, Alice finds documents with similar visual information. The respective elements are highlighted in red as can be seen in Figure 5.9.

Alice also finds documents with visual information similar to the top left image anchor, which are highlighted in red again (see Figure 5.10). In order to find all of those documents, Alice uses the toggle option for "Average Image Comparison" and observes the movement of the elements which can be seen in Figure 5.11. Alice concludes that elements drawn closer to the "darker" image anchors are worth inspecting for both comparison options

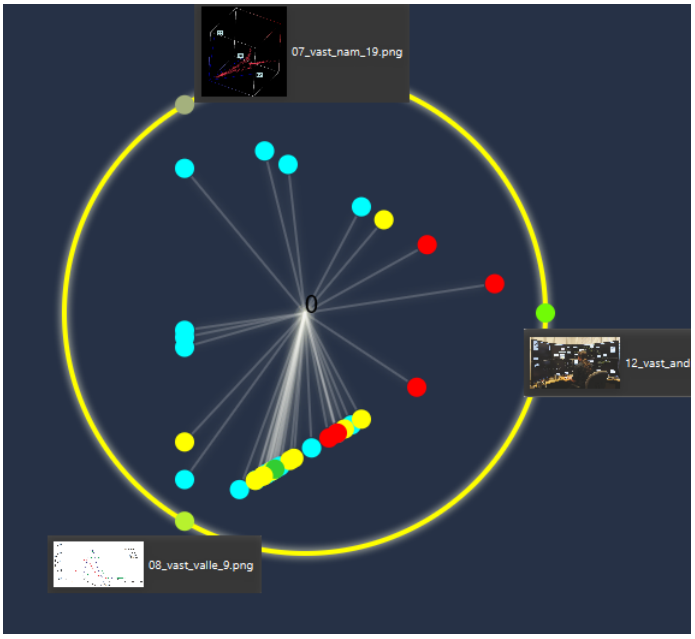


Figure 5.9: *RadVizDoc* configuration with three images. Elements with images similar to the most right image anchor are highlighted in red.

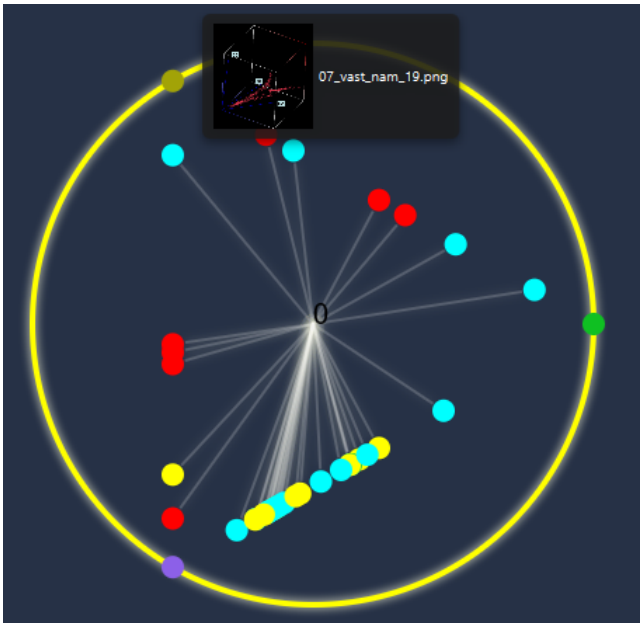


Figure 5.10: *RadVizDoc* configuration with three images. Elements with images similar to the top left image anchor are highlighted in red.

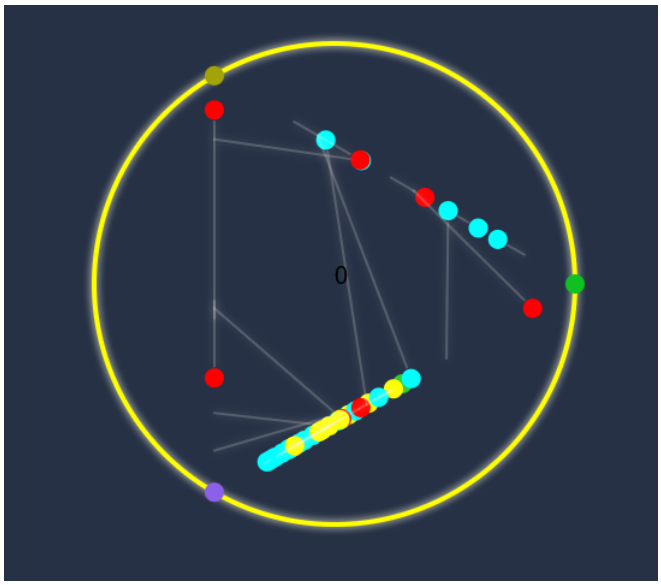


Figure 5.11: RadVizDoc configuration with three images and *"Average Image Comparison"* toggled on. Elements with images similar to the top left image anchor are highlighted in red.

(especially when their position changes dramatically after toggling the comparison option). This is because a document can have multiple images. In single image comparison mode, the best image for each anchor is considered when calculating its force. Consequentially, if a document has, for example, ten images it could be that nine of them are very similar to a specific anchor but one image is extremely similar to another anchor. When calculating the position for this element, this leads to a large disparity depending on which option is selected. As a result, Alice finds out that some of the documents drawn to this anchor also include additional images that are similar to the third anchor. Next, the same strategy is used to find similar documents for the third image anchor. For this last anchor Alice finds the most examples. This is of no surprise after seeing that most of the elements are drawn to this anchor. Alice already figured out that some elements are positioned in the middle between two anchors, because they contain multiple visually different images. Furthermore, Alice wants to know if there are documents in the collection, which are not part of her current subset, that contain similar visual information. Those two reasons lead Alice to use the *ImageSearchView*. She initiates an image similarity search for each of the three images that she is interested in. For each of the three images, Alice is presented with a list of the 50 top most similar images from the whole collection. Because of the color tagging system, Alice can see if the document belonging to a specific image is already in her subset. This allows Alice to swiftly browse through the result images and tag new documents if necessary. By using the *RadVizDocView* and the *ImageSearchView*, Alice found documents that contain visual information that is related to her interests. While the *ImageSearchView* allowed her to find documents that contain at least one similar image, the *RadVizDocView* enabled her to find documents that contain images similar to

two different anchors at the same time. Furthermore, the *RadVizDocView* provided Alice with knowledge about the average image distances from all images of a document to a certain anchor. Alice also realized that elements with positions close to each other can contain similar images (even though, they might not have similar images to the current anchors). Some example results that Alice found can be seen in Figure 5.12, Figure 5.13 and Figure 5.14.

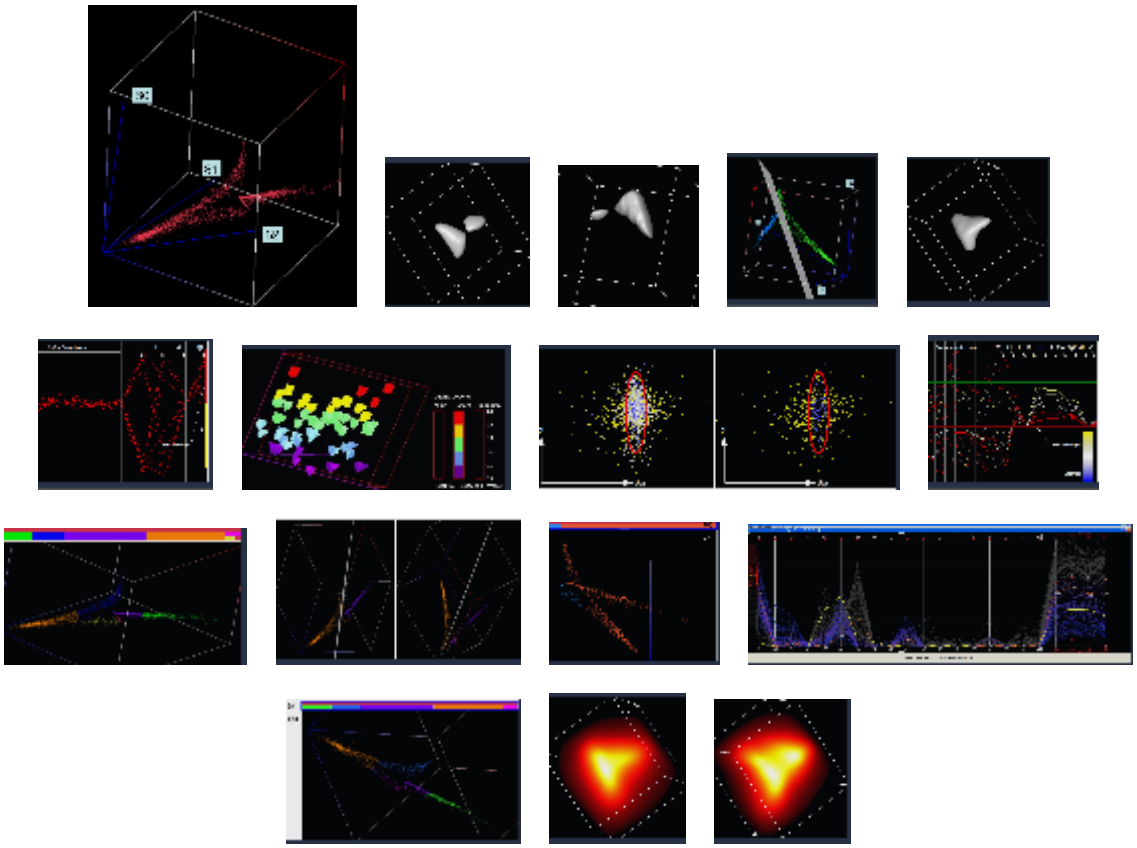


Figure 5.12: Similar images that Alice found in various documents by using Doxplorer. The first image, which is emphasized by size, depicts the image of interest. The query image shows a dark mode 3d visualization for high-dimensional data. The system finds images that are visually similar to the query image.

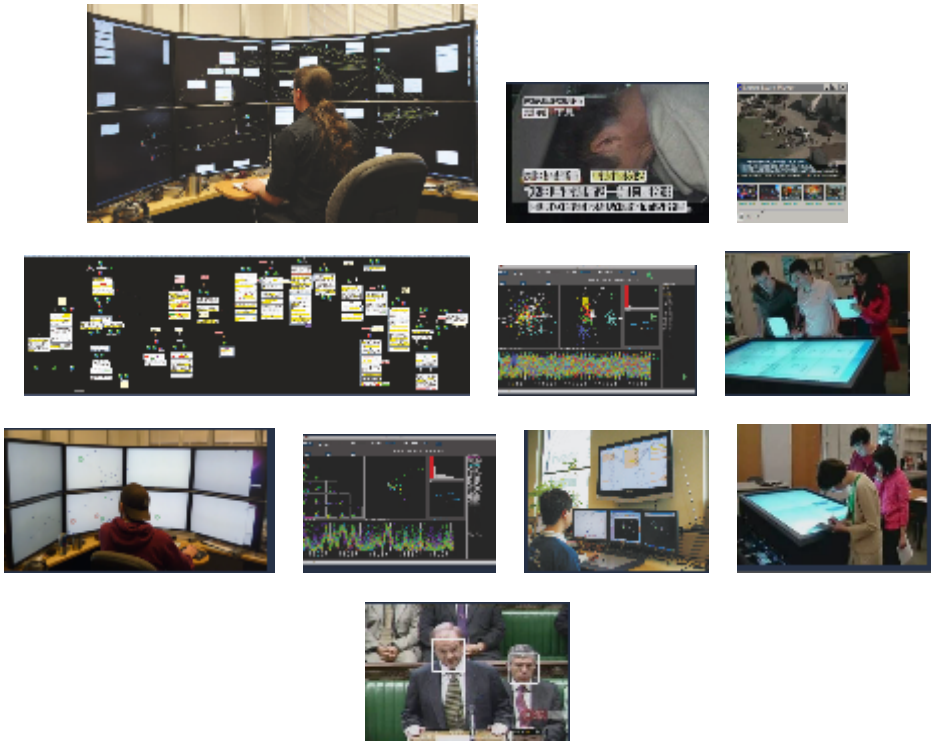


Figure 5.13: Similar images that Alice found in various documents by using Doxplorer. The first image, which is emphasized by size, depicts the image of interest. As we can see, the image comparison managed to find images that contain interfaces which are similar to the query image. Interestingly, the comparison also yielded images showing humans in front of displays which is in strong correlation with the query image.

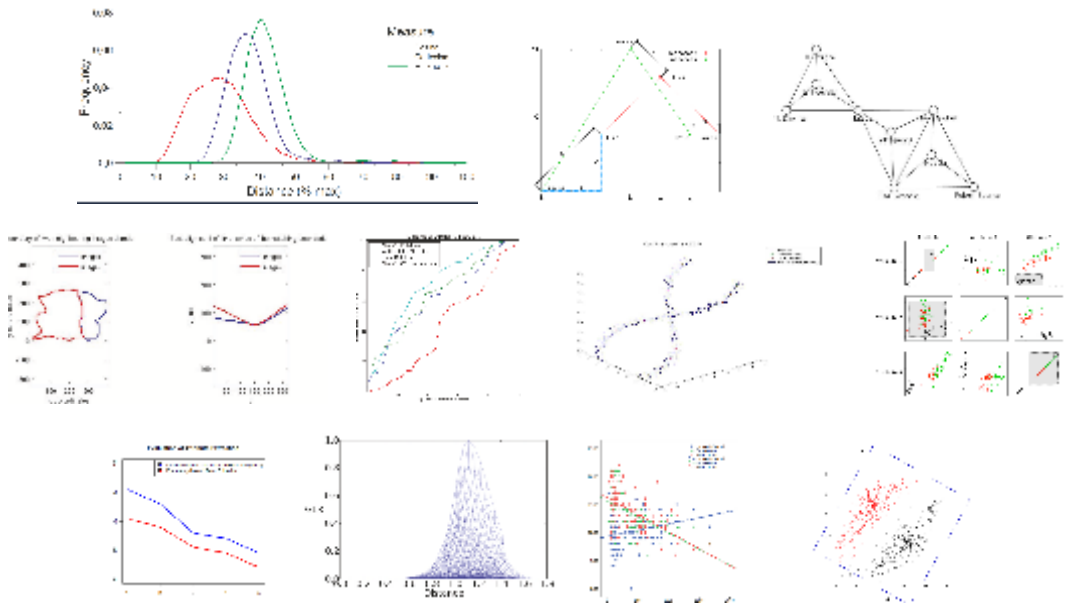


Figure 5.14: Similar images that Alice found in various documents by using Doxplorer. The first image, which is emphasized by size, depicts the image of interest. As we can see, Alice also managed to find similar images in this case. In this example, we can also see that the descriptors find different visualizations that use colors similar to the query image.

5.2.3 Scenario 3

In this scenario, Alice is interested in graph-based visualization types with nodes and edges. She wants to find out how this visualization method is used in different areas in order to get some new ideas for her own research. Alice decides to solve this task by using the image search functionality of *Doxplorer*. She starts by using the *ImageSearchView* in order to browse the images of the document collection. Then, Alice selects the first image which she considers relevant and tags the corresponding document. Afterwards, she initiates a similarity search for this specific image. In the results, Alice finds other visualizations that fit her criteria. However, Alice notices that in the results are also a lot of images that do not depict graphs with edges and nodes. This is because other images can still be visually similar in terms of color and edge connections without them being graphs. Furthermore, Alice realizes that the graphs can differ strongly depending on the data type they represent. For example, in one visualization the nodes of the graphs are depicted by images. Consequentially, Alice initiates new similarity searches for the graphs she previously found. She continues this process for a few iterations. This way, Alice manages to find various different kinds of graphs that all fit her criteria. Soon, Alice is in a position where she has marked various documents that seem relevant to her. She switches to the *VisualExplorerView* and inspects the document cards of the previously tagged documents. Hence, Alice can filter out documents that are not relevant by viewing all their contained images and other data, such as title. Furthermore, she can also use the image similarity function from the document cards. This enables Alice to find further relevant graph visualizations in case she missed any during her initial search. Example images of graphs that Alice found are shown in Figure 5.19. Further, the *VisualExplorerView* provides Alice with an overview of her subset according to year of publication (see Figure 5.15). Note that the two documents in the "No Date Information" category are also from 2006 even though they do not contain the specific meta data information (Alice knows this from inspecting the documents in a PDF viewer).

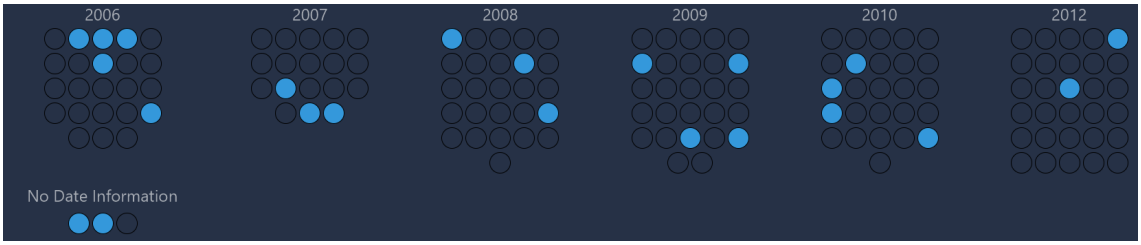
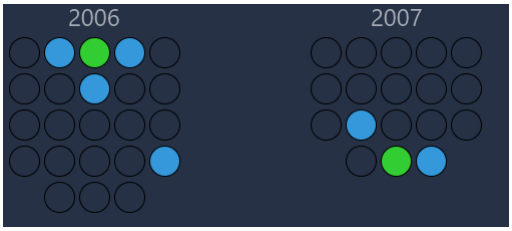


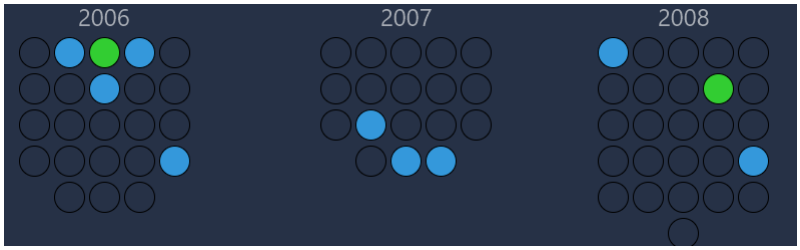
Figure 5.15: This figure shows the distribution for graph related documents according to year of publication that Alice previously found by using the image similarity functionality of *Doxplorer*.

Alice finds out that she discovered various papers with graph-based visual information that were published in 2006; whereas for the year 2012 she found only two. Alice also starts to investigate some authors. She notices that a lot of authors appear only once in the collection. Nonetheless, Alice can still use the authors' names that she finds to search

for them in other collections. However, Alice also finds authors who seem to be related to graph visualizations. For example, Alice finds Bongwon Suh and Stuart Card, who worked together on a paper about a graph-based visualization method. Alice decides to use the author similarity search in order to find out if they contributed to another paper from the collection (see Figure 5.16).



(a) Author similarity search for Bongwon Suh.



(b) Author similarity search for Stuart Card.

Figure 5.16: This figure shows the results of the author similarity search for two authors from the same paper.

Alice notices, that the resulting documents are already in her subset and inspects the corresponding document cards. As a result, she finds out that Bongwon Suh also worked on a paper about understanding the social dynamics on Wikipedia by using revert graph visualizations. Further, Alice discovers that Stuart Card worked on another paper about entity-based collaboration tools which utilizes graph visualizations. Another example for an author, that Alice finds, is Pak Chung Wong, who also appears in two papers that are already in the subset. One of those papers is about a zooming approach for large graph analytics and the other one describes an analytics framework for large semantic graphs. Alice realizes, that by using the author similarity search, she often finds papers that are already in her subset. This way, she can find authors who focus on graph-based visualizations. This indicates a correlation between certain visualization types and authors. As shown in this examples, Alice found overlapping documents by using the image search functionality and the author similarity search. Furthermore, Alice makes an additional observation when she continues using the author and image similarity searches. For example, she opens the document card of a paper that uses a graph-based visualization technique for large-scale news video databases which can be seen in Figure 5.17.

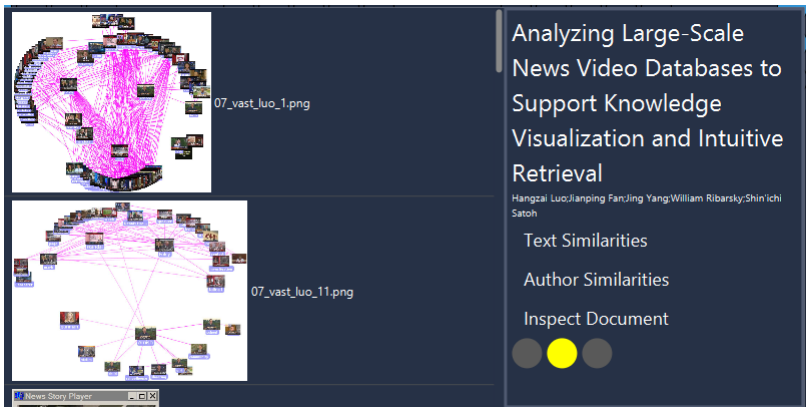
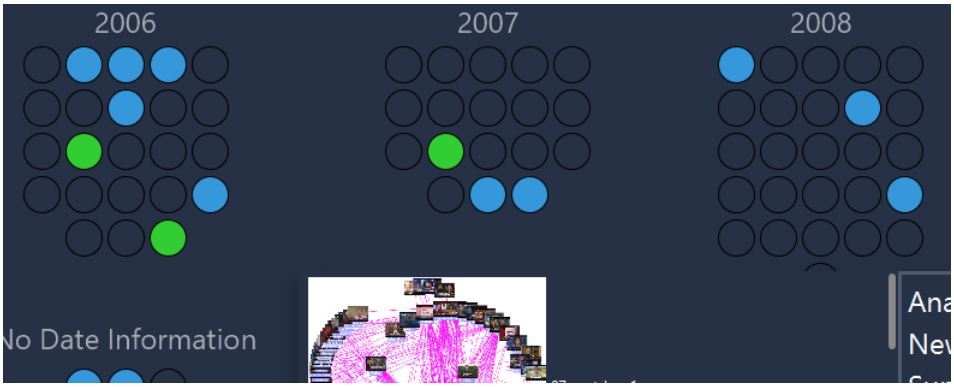
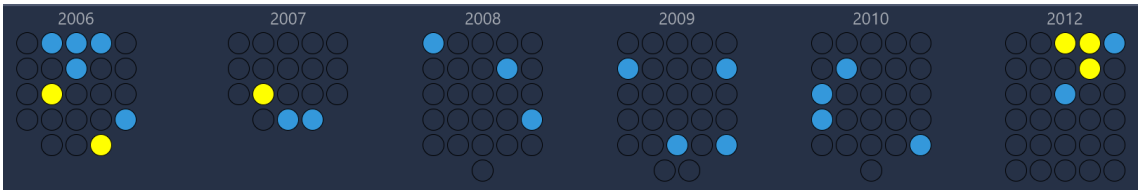


Figure 5.17: Document card for a paper about visualizing large-scale news video databases.

Afterwards, Alice initiates an author similarity search for this document. She discovers that the collection contains three documents from this author. However, this time only one of the papers is contained in her subset of graph related documents. Alice takes a look at the other two documents. She discovers a relation between all three documents; they are about visualizing image collections or news video databases. Thus, Alice sees that the author is not focused on graph-based visualization methods but is interested in the field of exploring visual information. She learns that the author uses different visualization approaches for similar types of data in the distinct papers. Alice becomes curious about one of the papers from this author and initiates an image similarity. To Alice's surprise, the image similarity search returns all papers from this author and overlaps with the author search results (see Figure 5.18). This further indicates a correlation between visual information and authors. In the end, Alice finds most papers from the collection that are visually related to graph-based visualizations and also authors that focus on this type of visualization. Later, by inspecting the documents from this subset, Alice can investigate in which domains graph-based visualizations are used. Most of the documents that Alice found covered topics or visualization ideas related to graphs. From her findings, Alice concludes that, depending on the domain, it is possible to find correlations between visual information, content and authors.



(a) Results of the author similarity search for Hangzai.



(b) Image similarity search for the first image from a paper by Hangzai, which is about exploring large-scale video news with an interactive visualization method.

Figure 5.18: This figure shows a correlation between the image similarity search and author similarity search for papers by Hangzai.

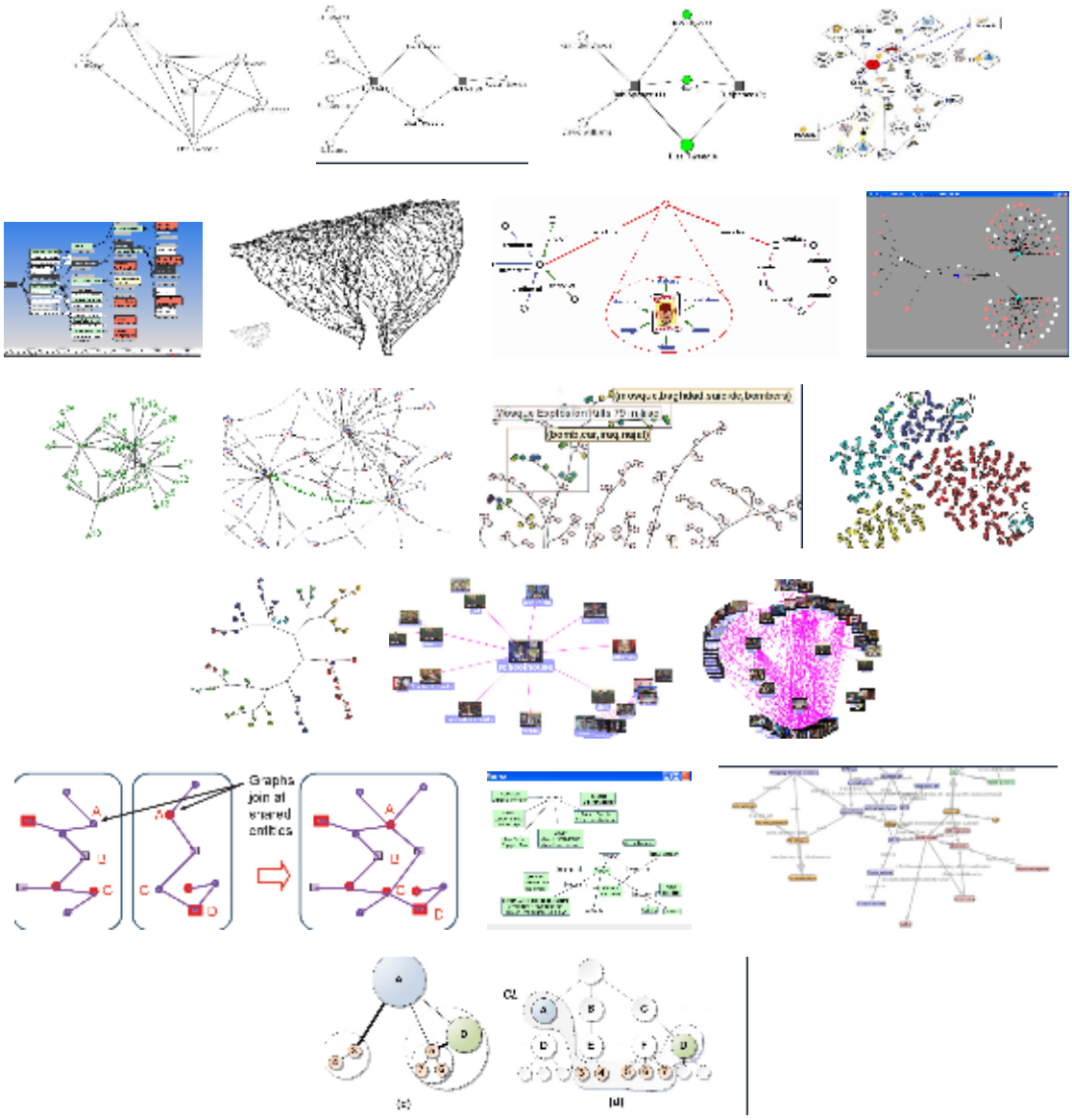


Figure 5.19: Different kinds of graphs contained in the collection, that Alice found by iteratively using the image similarity search functionality of *Doxplorer*.

5.2.4 Scenario 4

Alice is provided with a document collection which she has no information about (still the VAST data set with 153 documents) and decides to find out if there are any documents that are interesting to her. Alice is interested in topics related to health and wants to know how visual analytics can be applied in this area. Her goal is to establish a subset which could be analyzed with *RadVizDoc* at a later point time. Thus, she uses the *SearchView* to query the collection. Alice uses the following query: (health^4 AND "visual analytics"). This query boosts the term "health" and ensures that the phrase "visual analytics" also appears in the same document. Alice ends up with a result list comprising 27 documents and decides to further explore the collection within the *VisualExplorerView*. She notices that the found documents are spread across all publication years. However, especially in the years 2007 and 2010 but also in 2012 there were more documents related to that query than in other years (see Fig. 5.20). Now, Alice selects one of the highest ranking documents and opens its document card as shown in Figure 5.21. As a result, Alice can directly perceive that the document is about visualizing links between human and animal health. For the first two images from the document card, she initiates an image similarity search (see Fig. 5.22). For the second image, half of the found documents are already contained in the subset which indicates correlations between content and images. By using the image similarity search from this document card, Alice finds a document from 2009 which is about genetic mechanisms of complex human disorders that is not part of the current subset and adds it. When searching for the second image of the document card, the most interesting document was about a use case study on spatiotemporal data associated with the Avian flu. After inspecting document cards from multiple documents of the current subset, Alice notices that some of the higher scoring health related visualizations use high-dimensional data (e.g., see Fig. 5.23). By searching for similar images she finds other documents that are, for example, from the bioinformatics field and deal with visualizing gene expressions. Since Alice is interested in the paper about linked human and animal health she initiates an author similarity search for the first two authors. For example, she finds papers about mobile devices for emergency responses (e.g., health status of agent), and visualization methods for syndromic hot spots by using the author search. As a consequence of using the author search for this paper, Alice also finds out that Ross Maciejewski appears in two high scoring documents related to health. In both papers, spatiotemporal visualizations are used. Alice realizes that Maciejewski published two more papers, one about visual analytics for human decision making and one which describes a novel visual analytics approach for spatiotemporal and multivariate data. Alice realizes that the methods proposed in the second document could again be applied in fields related to health. Alice continues to refine the set of interesting documents by using the author and image similarity functionalities. In addition, Alice inspects document cards from documents that were previously tagged, after using the *SearchView*, and removes

them from the subset if they are not of interest to her. After establishing a base of interesting documents, Alice initiates a text similarity search for the document which is about the Avian flu. Most of the returned results are already in the current subsets, for example, the document about syndromic hot spots. However, Alice finds a new interesting document by using the text similarity search which is about collaborative synthesis of visual analytics conducted in collaboration with disease biologists where the task of finding the origin of an influenza outbreak is simulated. Moreover, the similarity search helps Alice to find a document which is about understanding multi faceted text corpora of, for example, patient records. Alice continues to use text similarity searches on the documents which are most relevant to her. She also obtains an understanding of correlations between documents in the subset when using the text similarity function.

In the end, Alice establishes a subset which is about visualization techniques for multivariate data, spatiotemporal data and collaborative visualization of analytic results which can all be applied to the field of human health or human rescue. While creating this subset, Alice already acquired a good overview of the documents which are relevant to her from this collection. While at the beginning it looked like there are not many papers related to health, Alice found out that multiple interesting documents are related to this topic in some way even though they do not directly include the term “health”.

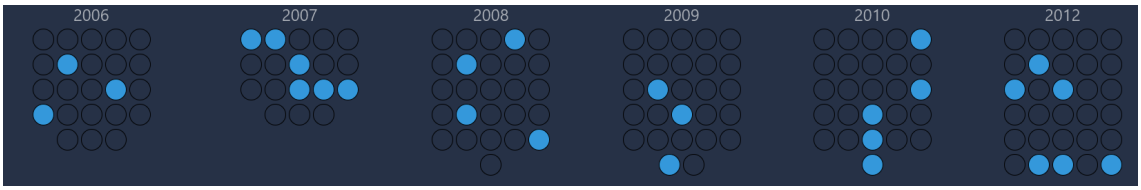


Figure 5.20: *VisualExplorerView* showing the distribution of documents related to query: health^4 AND “visual analytics” over the year of publication. Particularly high percentage of documents related to the provided query in the year 2007.

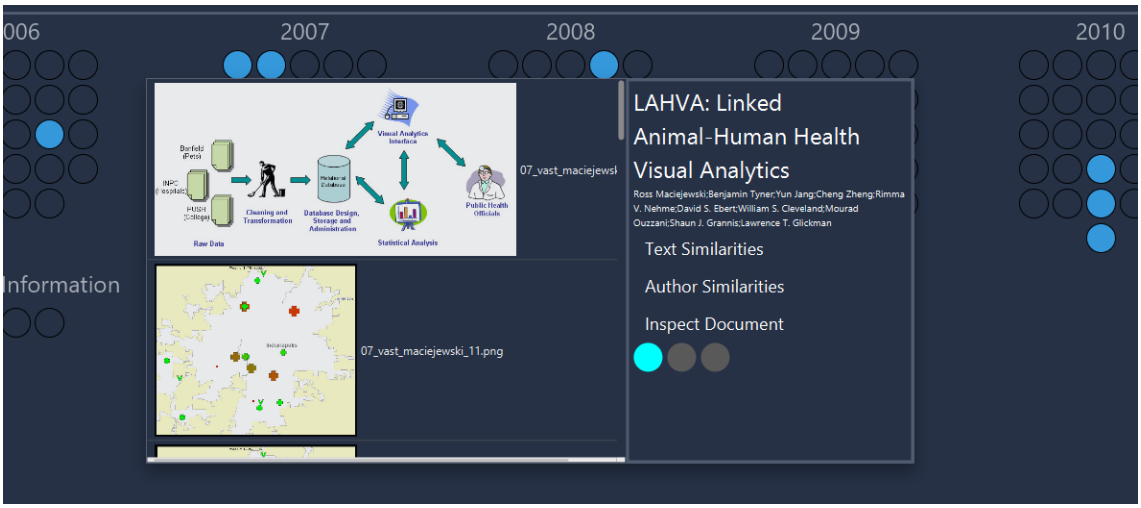


Figure 5.21: *VisualExplorerView* showing an overview for a previously generated subset. Additionally, a document card of a high ranking document is opened.

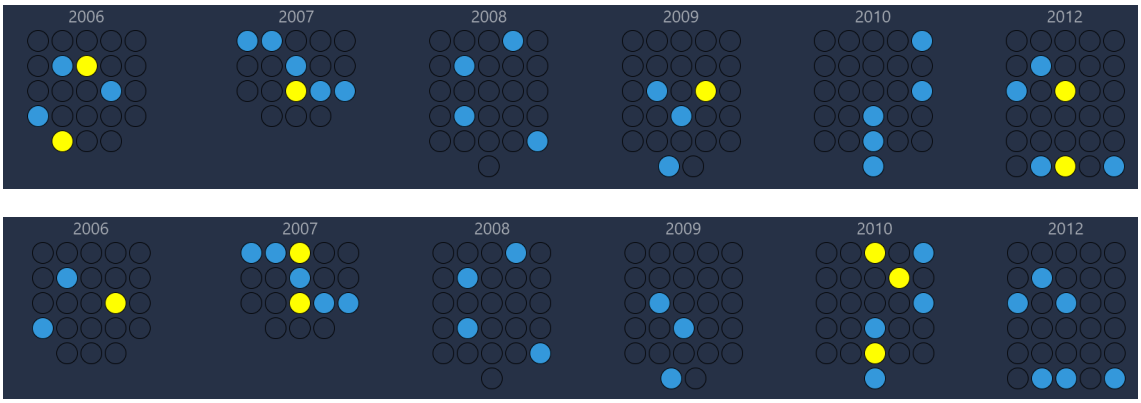


Figure 5.22: *VisualExplorerView* showing results for two different image similarity searches for a specific scenario. Results are marked in yellow and discussed in the respective scenario.

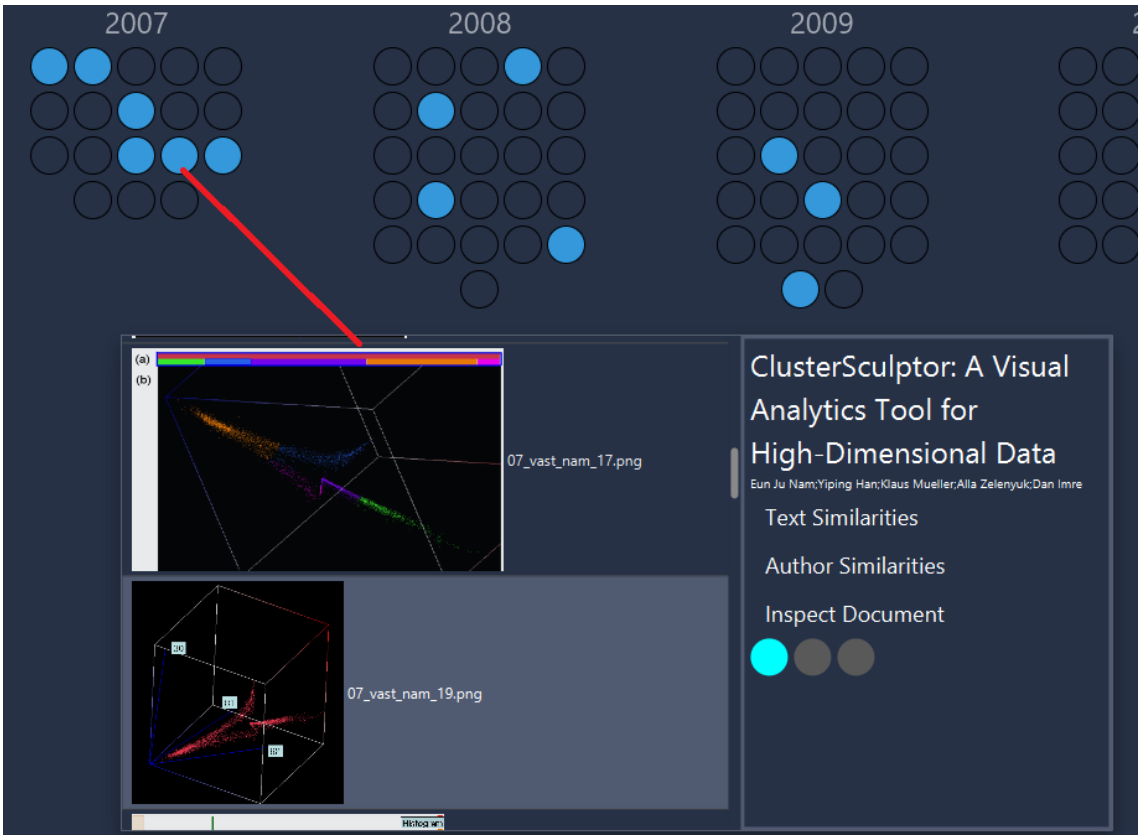
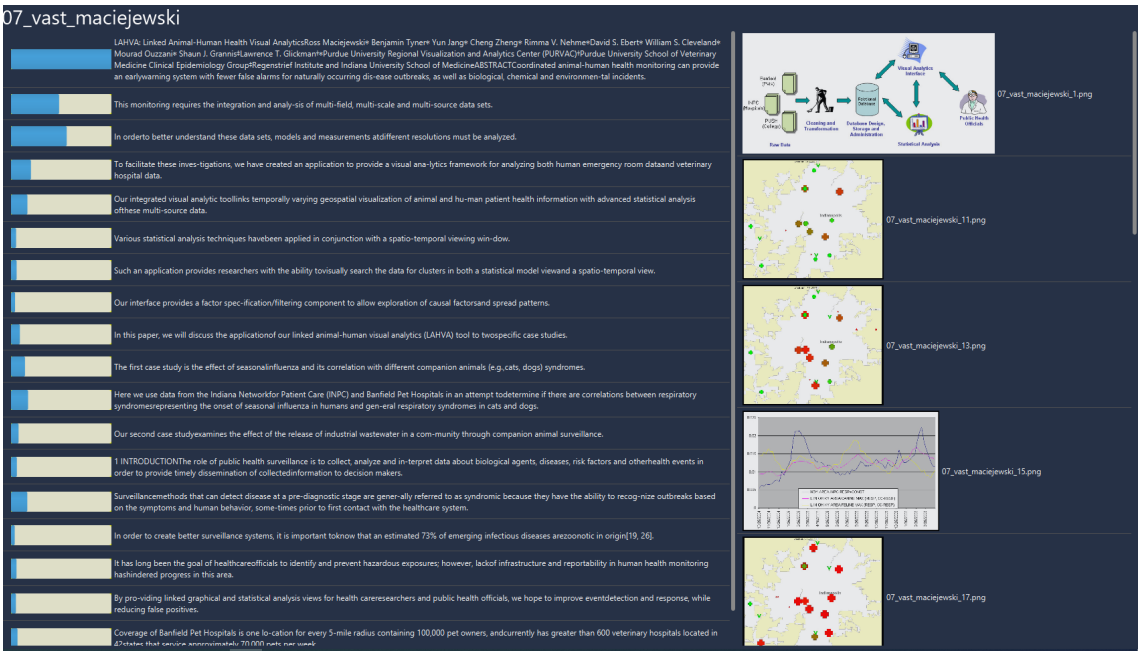


Figure 5.23: *VisualExplorerView* showing documents related to query: health^4 AND "visual analytics". Additionally, a document card is opened for further inspection.

5.2.5 Scenario 5

In this scenario, Alice is in the middle of an exploration process and has already established an intermediate subset of interesting documents. Due to *RadVizDoc* visualizations and similarity searchers from the *ExplorerView* Alice found documents that could be of special interest to her. Since document cards do not provide enough information in this case, Alice decides to use the *InspectView*. Hence, we accompany Alice when she is inspecting a document within this view. When Alice inspects a document for the first time within this view, she sees the summary generation menu. Here, she can already see important key terms from the document. Figure 5.24 shows the summary generation menu and a resulting summary report. The summary report, which Alice receives, provides a basic overview of the document content in just a few sentences. Since Alice selected options for a short summary report, it contains most of the basic information from the abstract and snippets from the middle and end sections of the paper. Now, that Alice has an understanding of the basic document content, she wants to know the final conclusion of the paper. Thus, she opens the abstract and conclusion sub-panel (see Fig. 5.25). Afterwards, Alice decides that the document is critical to her research. Thus, she chooses to display a tag cloud in order to find important terms which can be used to cluster documents in the *RadVizDocView* (see Fig. 5.26).

(a) Shows the summary generation menu. The top 400 terms are listed according to their tf-idf scores in descending order.



(b) Displays the summary report that was created beforehand.

Figure 5.24: *InspectView* showing the summary generation menu and the final summary report.

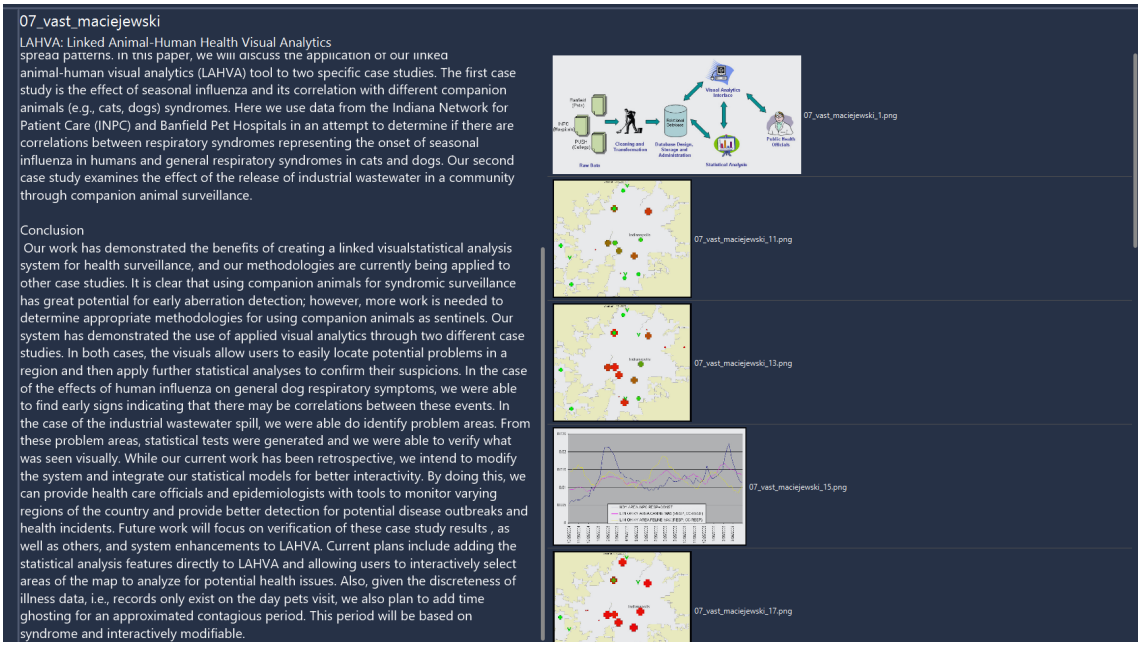


Figure 5.25: *InspectView* showing extracted abstract and conclusion of a document.

Figure 5.26: *InspectView* displaying tag cloud for a specifc document.

5.2.6 Scenario 6

In this scenario, Alice is provided with a subset of documents about different domains. She is interested in various fields, such as finances, health and network traffic. Within the subset, Alice knows three documents, one for each topic, that fit those criteria. Alice already knows how she can find similar documents by using the text similarity and image similarity features of the *VisualExplorerView*. However, Alice is interested if there is a way to find relations between documents by using summary reports. She thinks, that this could complement the other features; especially, because summary reports are usually longer than the automatically generated queries for the text similarity search. Consequentially, Alice creates summary reports for the three aforementioned documents. Next, she uses those summary reports as anchors within the *RadVizDocView* (see Figure 5.27).

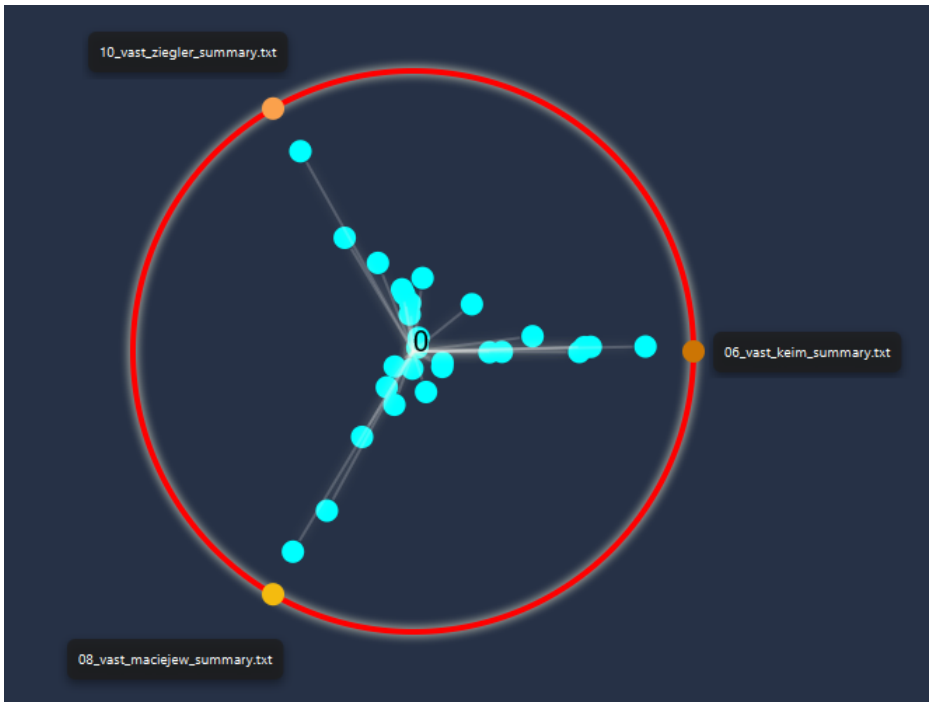


Figure 5.27: *RadVizDocView* showing the distribution of a subset according to summary report anchors.

The anchor on the right depicts a summary of a paper related to network traffic. When inspecting documents drawn to that anchor, Alice sees that most of them are very related in terms of content. Those contents range from wormhole detection in wireless networks over to accelerating network traffic analytics and other network traffic related papers. Further, she notices that most of those documents contain colorful figures. Alice concludes, that network traffic related visualizations seem to rely on colors to convey information (see Figure 5.28).

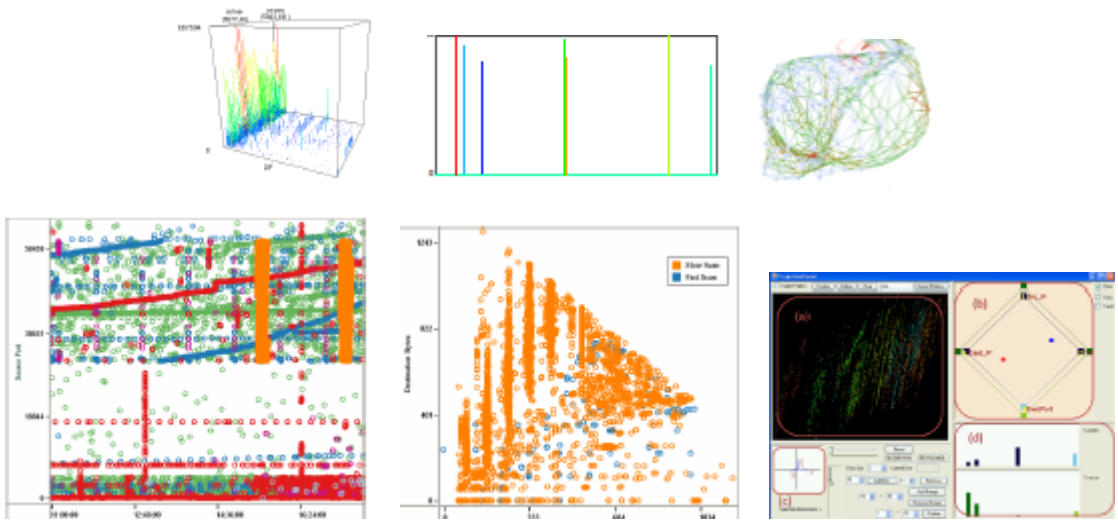


Figure 5.28: Images from papers that are related to network traffic and pulled by the same summary anchor.

The bottom left anchor represents a document related to syndromic hotspots. The visualization shows that only a small part of the subset correlates strongly with this anchor. Yet, Alice finds three documents that are strongly related in terms of content (flus, diseases and health). Moreover, she discovers that some documents, which tend towards this anchor, are related in other ways. For example, they use spatiotemporal visualizations for different domains, such as syndromic hotspots, abnormal event detection, outbreaks of flus, surveillance applications and more. By using this summary report as an anchor, Alice finds different papers that are related to spatiotemporal data visualizations and contain similar visual information. A few image examples of those papers are shown in Figure 5.30.

The third anchor depicts a summary report about analysis for financial time series data. Again, Alice finds that most documents drawn to this anchor are related content-wise in some way. Moreover, she notices something very interesting. Three documents from the subset, that are drawn to this anchor, contain images from the same visual analytics interface called WireVis [4] (see Figure 5.29). Another interesting observation is, that the elements depicting those documents are positioned very closely to one another (even though, the search was only text-based). This indicates a relation between textual and visual information. Alice learns that WireWis can be used for visual financial analysis. The three papers containing figures of the WireVis interface are *07_vast_chang.pdf*, *08_vast_jeong.pdf* and *10_vast_lipford.pdf*. Further, Alice discovers that documents drawn to this anchor tend to contain figures of interfaces, bar charts and line-based charts. Figure 5.31 shows example images. The first two images illustrate the WireVis interface. However, Alice becomes very interested in WireVis and wants to know if there are other documents that are related to it which are currently not in the subset. Thus, Alice decides to initiate an image similarity search for WireVis in the *VisualExplorerView*. To

her surprise, Alice does not only find the documents that she already knows but also another paper including an image of WireVis which is currently not part of the subset (*08_vast_green.pdf*). Finally, Alice concludes that summary anchors can be of great use to find related papers and that it is also possible to find visual information commonalities between those documents. Since summary queries contain a lot more words than text similarity queries, they can help discover different relations, even if the domains of the papers differ.

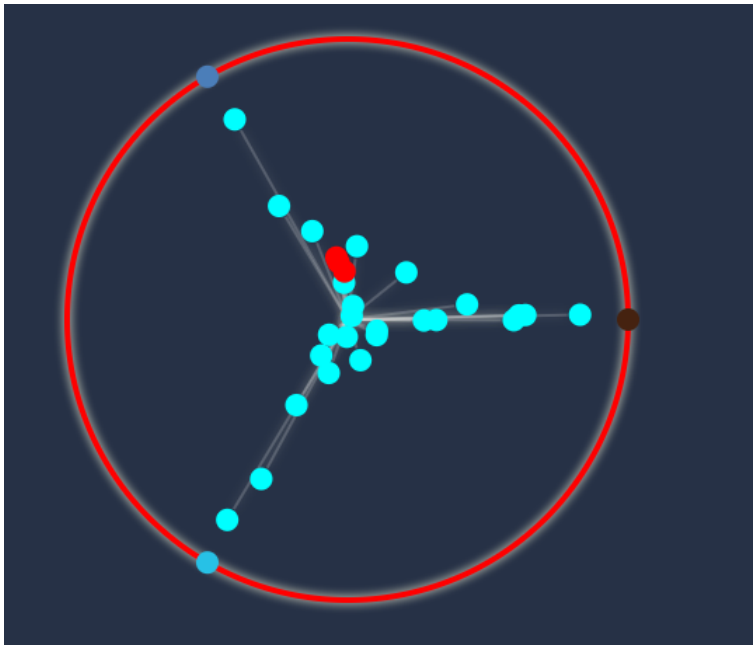


Figure 5.29: Distribution of documents according to summary anchors, where documents including images of WireVis are marked in red. All three documents are placed very close to each other, even though, the anchors in this scenario are only text-based.

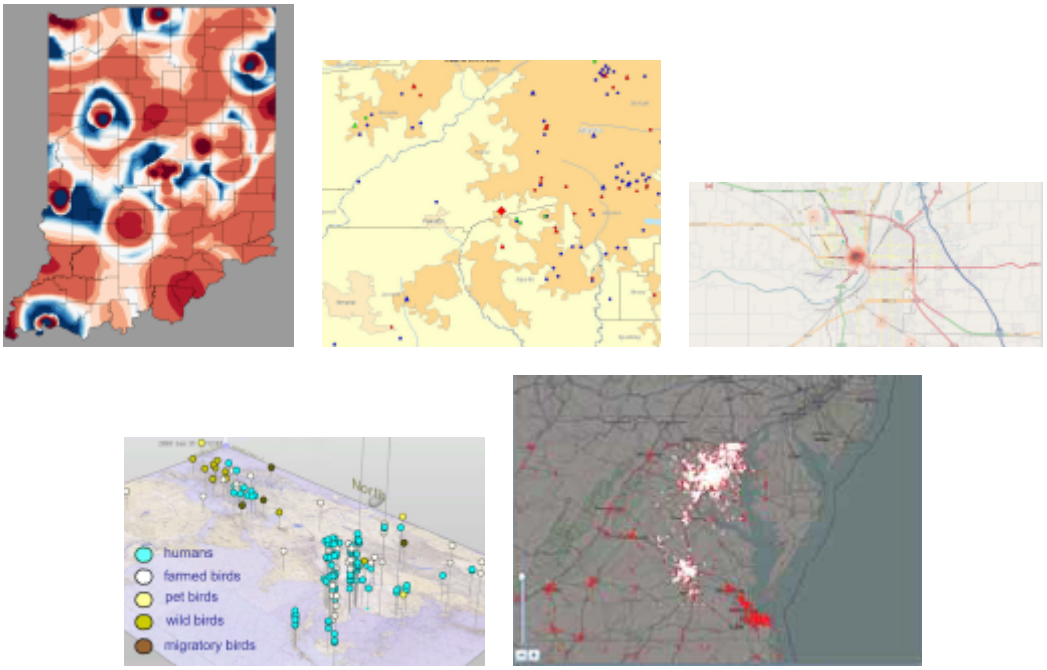


Figure 5.30: Images from different papers that are related to spatiotemporal data and pulled by the same summary anchor.

Figure 5.31: Images from different papers that are related to the same summary anchor about financial time series data. The first two images show the WireVis interface.

6 Discussion and Concluding Remarks

6.1 General Discussion

As we could see in the example scenarios, there are many different ways documents can be related. While the basic *SearchView* and *ImageSearchView* are well suited for establishing a starting sample of documents, the *VisualExplorerView* and the *RadVizDocView* aid the user in finding related documents or clusters. The document cards and *InspectView* allow the system to show details-on-demand when needed. By combining those views, it is possible to swiftly acquire a good overview of a medium-sized collection. For example, in one scenario we could see how Alice did not only find documents to her queries, but also found documents related to the initial subset by using similarity searches. Further, she discovered relations between documents and gained a basic idea of visualizations that can be used in various domains. The scenario results also show that the system works well for finding certain image types. Results suggest that the FCTH descriptors work well for finding:

- Images showing humans
- Different charts
- Graphs based on nodes and edges
- Colorful visualizations
- Dark images
- Figures showing interfaces

A general problem that we faced when using the system was that for some images there are no other images in the collection that could be considered similar in a contextual way. This is due to the fact that our data set consists of only 153 documents and a lot of the visual information contained in the set is very unique (e.g., novel visualization techniques).

Nevertheless, the system still provides result images for those unique images which have similar colors. Thus, they can be considered slightly similar in a visual way. The text-based similarity function also provides satisfying results. The only minor problem is that the extraction of Portable Document Format (PDF) document texts is not a hundred percent reliable; meaning that some words are only extracted partially or headlines merge with texts due to missing full stops. However, this mainly influences summary reports. Due to the tf-idf nature of the similarity calculations and text search, the faulty extractions do not have a great impact on them. Since all similarity calculations are implemented by using Lucene and LIRe, all views except the *DocRadVizView* could potentially be applied on larger data sets as well. Although, the *VisualExplorerView* would need a redesign for displaying larger collections. Moreover, at the moment the system is designed to use provided XML files for abstract and conclusion extraction. Furthermore, it is not possible to index new documents while the system is running. Those things would need to be changed in order to use the system on other document collections.

6.2 *RadVizDocView* Discussion

The visualization proved to be useful in many different ways, as we could see in the scenarios. We have shown, that it is possible to find documents that are related to each other in different ways by using this visualization technique. The option to use multiple queries at the same time, allows us to find relations according to the positions of the elements. The visualization also provides a way to see document distributions over visual information or textual content. Additionally, we have shown example findings in the scenarios that indicate a correlation between textual and visual information. Further, by using the option to interactively change the amount of anchors simultaneously pulling on elements, we can cluster elements according to the anchors as shown in one scenario. This works well for all supported data types and can be of great use to find relations among documents or anchors, as we could see. Applying only rings of a single data type concurrently has proven to be the most effective and useful way to use the visualization. Note, that multiple rings of different data types can still be used, but they should be activated alternately. Activating rings of different data types with many anchors simultaneously has shown to be less effective. The reason for this is that it is very hard to interpret the element positions for such complex configurations (anchor forces are equally distributed among different data types). For instance, a document could be related to text query anchors on the right, while it is also related to image anchors on the left. This would result in the element staying in the middle (images and texts both have 50% of the available anchor force) of the visualization. However, this problem can be avoided to a certain degree by limiting the amount of concurrently pulling anchors for a specific data type and not simultaneously using more than two data types. Then, it is still possible to interpret the positions in a meaningful way. Nevertheless, we find it most useful to only activate rings of one data type simultaneously. However, multi-ring configurations can

still benefit the user. For example, he/she could create topic anchors (like in scenario 1 before) and afterwards also create an author ring. This way, the user could see how documents from different topic clusters are related to certain authors. Another possibility would be to cluster elements according to text queries. Afterwards, the user can mark the documents in red and activate another ring (perhaps an image ring) while deactivating the current ring. This procedure enables the user to see if documents, that are clustered according to one ring, are also positioned close to each other when using another ring. In the next section, about future work, we also provide an idea how the problem, that arises when simultaneously using multiple different data type rings, could be avoided.

Since the text search features are all implemented with Lucene, the respective anchors also work for larger collections in terms of performance. However, at the moment we extract the image features for this visualization when the view is opened. This, of course, takes time depending on how many documents and images are contained within the subset. This puts a limiting factor on the subset size. However, if the images of all documents from the subset were indexed beforehand in their own folder, it would work for larger collections as well.

6.3 Future Work

The system could be extended by finding a better way to extract the texts of *PDF* documents. Furthermore, the system can be enhanced by using a better summarization method; perhaps an extractive summarization method based on neural network models similar to the one proposed by Nallapati et al. [25]. An interesting idea for future research could also be to develop a method that automatically learns and generates *RadVizDoc* configurations. This could perhaps allow the user to use multiple different anchor types simultaneously by grouping anchors that draw similar elements. This would avoid the current problem with too many different data type anchors, where the user has to manually move anchors to get a feeling for how to position them. The implementation could be done by creating element clusters for anchors which are then compared to find intersections. Topic modeling would also be a well suited addition to the system. As a result, the system could automatically create topics for subsets and generate anchors depicting those topics. Semantics based state of the art text similarity techniques could also be added to retrieve even better results when looking for similar documents. Contextual information could be added as well. For example, by creating a log file of the entire exploration process while the system is learning from it.

6.4 Conclusion

In this thesis we set out to create a prototype document retrieval system called *Doxplorer* that uses Content Based Image Retrieval (CBIR), full-text search, summary reports, meta data information and custom visualizations for exploring medium-sized document collections. This chapter summarizes the presented work.

In Chapter 2 traditional Information Retrieval (IR) and its neglection of visual information was described. Additionally, the three basic models used in traditional *IR* were covered. Moreover, an overview of different *IR* fields of research (e.g., Private Information Retrieval (PIR)) was provided in this chapter. Afterwards, different single and multi document visualizations were presented. The chapter concluded with an overview of current state of the art document summarization methods.

Chapter 3 explained the basic concepts for the system. The initial ideas for the different system views (e.g., *ExplorerView*, *SearchView*, *ImageSearchView* and *RadVizDocView*) were explained and illustrated by mock-ups of previous design iterations. This chapter was dedicated to explain what the system should do and not how.

Next, we had Chapter 4 which was about the implementation of the concepts proposed in Chapter 3. In this chapter, the different algorithms and libraries which we used were described. Especially, the implementation of our custom visualization called *RadVizDoc* was explained thoroughly in this chapter.

Chapter 5 started by providing a set of different use cases for the system. Those use cases described how the system should be used. Afterwards, a fictional character called Alice performed multiple scenarios on the VAST data set of the years 2006-2012 (except for the 2011 set) by using our system. The scenarios have shown good results and indicated correlations between visual and textual information.

Finally, we can say that our prototype has provided promising results and shows that the principles behind the system work. Undoubtedly, the possibility to search for images and text within a single system is already powerful on its own. However, the application of the system illustrated how different document similarity searches and visualizations can further help to find relevant documents. We also provided an outlook showing that there are various options for additions to the system that could be made in future work to build upon those principles.

A

List of Acronyms

CBIR	Content Based Image Retrieval
CEDD	Color and Edge Directivity Descriptor
CLIR	Cross-Language Information Retrieval
DOM	Document Object Model
DUC	Document Understanding Conference
FCTH	Fuzzy Color and Texture Histogram
GUI	Graphical User Interface
IR	Information Retrieval
IRS	Information Retrieval System
NLP	Natural Language Processing
PDF	Portable Document Format
PIR	Private Information Retrieval
PNG	Portable Network Graphic
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
TF	Term Frequency

Bibliography

[1] K. Banawan and S. Ulukus. The capacity of private information retrieval from coded databases. *IEEE Transactions on Information Theory*, 64(3):1945–1956, 2018. (page 7)

[2] K. Banawan and S. Ulukus. Noisy private information retrieval: On separability of channel coding and information retrieval. *IEEE Transactions on Information Theory*, 65(12):8232–8249, 2019. (page 7)

[3] N. Cao, J. Sun, Y.-R. Lin, D. Gotz, S. Liu, and H. Qu. Facetatlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1172–1181, 2010. (page 15)

[4] R. Chang, M. Ghoniem, R. Kosara, W. Ribarsky, J. Yang, E. Suma, C. Ziemkiewicz, D. Kern, and A. Sudjianto. Wirevis: Visualization of categorical, time-varying data from financial transactions. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 155–162. IEEE, 2007. (page 82)

[5] S. A. Chatzichristofis and Y. S. Boutalis. Fcth: Fuzzy color and texture histogram - a low level feature for accurate image retrieval. In *2008 Ninth International Workshop on Image Analysis for Multimedia Interactive Services*, pages 191–196, 2008. (page 34)

[6] J. Cheng and M. Lapata. Neural summarization by extracting sentences and words. *ArXiv Preprint ArXiv:1603.07252*, 2016. (page 17)

[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50. IEEE, 1995. (page 6)

[8] V. Dalal and L. Malik. A survey of extractive and abstractive text summarization techniques. In *2013 6th International Conference on Emerging Trends in Engineering and Technology*, pages 109–110. IEEE, 2013. (page 16)

[9] M. Dörk, N. H. Riche, G. Ramos, and S. Dumais. Pivotpaths: Strolling through faceted information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2709–2718, 2012. (page 15, 16)

[10] R. Eckart de Castilho and I. Gurevych. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University, `https://www.aclweb.org/anthology/W14-5201`. (page 36)

[11] M. Fischer. The kwic index concept: A retrospective view. *American Documentation*, 17(2):57–70, 1966. (page 12)

[12] J. Foote. An overview of audio information retrieval. *Multimedia Systems*, 7(1):2–10, 1999. (page 6)

[13] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, and D. A. Karpuk. Private information retrieval from coded databases with colluding servers. *SIAM Journal on Applied Algebra and Geometry*, 1(1):647–664, 2017. (page 7)

[14] Q. Gan, M. Zhu, M. Li, T. Liang, Y. Cao, and B. Zhou. Document visualization: An overview of current research. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6, 01 2014. (page 3, 8, 9, 12)

[15] Y. Hassan-Montero and V. Herrero-Solana. Improving tag-clouds as visual information retrieval interfaces. In *International Conference on Multidisciplinary Information Sciences and Technologies*, pages 25–28, 2006. (page 9)

[16] S. Havre, B. Hetzler, and L. Nowell. Themeriver: Visualizing theme changes over time. In *IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings*, pages 115–123. IEEE, 2000. (page 13, 14, 15)

[17] P. Isenberg, F. Heimerl, S. Koch, T. Isenberg, P. Xu, C. D. Stolper, M. Sedlmair, J. Chen, T. Möller, and J. Stasko. Vispubdata.org: A metadata collection about ieee visualization (vis) publications. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2199–2206, 2017. (page 32, 33)

[18] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. (page 17)

[19] Y.-R. Lin, J. Sun, N. Cao, and S. Liu. Contextour: Contextual contour visual analysis on dynamic multi-relational clustering. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 418–429. SIAM, 2010. (page 15)

[20] D. M. Lux. Lire: Lucene image retrieval. `http://www.lire-project.net/`. Accessed: 2020-09-23. (page 34)

[21] M. Lux and S. A. Chatzichristofis. Lire: Lucene image retrieval: An extensible java cbir library. In *Proceedings of the 16th ACM International Conference on Multimedia*, pages 1085–1088, New York, NY, USA, 2008. Association for Computing Machinery, `https://doi.org/10.1145/1459359.1459577`. (page 33, 34)

[22] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244, 1960. (page 6)

[23] B. Mitra and N. Craswell. Neural models for information retrieval. *ArXiv Preprint ArXiv:1705.01509*, 2017. (page 7, 8)

[24] N. Moratanch and S. Chitrakala. A survey on extractive text summarization. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 1–6. IEEE, 2017. (page 16, 17)

[25] R. Nallapati, F. Zhai, and B. Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *ArXiv Preprint ArXiv:1611.04230*, 2016. (page 17, 89)

[26] J.-Y. Nie. Cross-language information retrieval. *Synthesis Lectures on Human Language Technologies*, 3(1):1–125, 2010. (page 6)

[27] L. Nováková and O. Štepánková. Multidimensional clusters in radviz. In *Proceedings of the 6th WSEAS International Conference on Simulation, Modelling and Optimization*, pages 470–475, 2006. (page 19, 26)

[28] P. Riehmann, D. Kiesel, M. Kohlhaas, and B. Froehlich. Visualizing a thinker's life. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1803–1816, 2018. (page 13)

[29] S. E. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 1977. (page 6)

[30] M. Rubio-Sánchez, L. Raya, F. Diaz, and A. Sanchez. A comparative study between radviz and star coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):619–628, 2015. (page 26)

[31] D. Rusu, B. Fortuna, D. Mladenic, M. Grobelnik, and R. Sipoš. Document visualization based on semantic graphs. In *2009 13th International Conference Information Visualisation*, pages 292–297. IEEE, 2009. (page 11)

[32] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. (page 5)

[33] T. Saracevic. Digital library evaluation: Toward evolution of concepts. *Library Trends*, 49(2), 09 2000. (page 2)

[34] S. P. Shariatpanahi, M. J. Siavoshani, and M. A. Maddah-Ali. Multi-message private information retrieval with private side information. In *2018 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2018. (page 7)

[35] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–50, 2005. (page 6)

[36] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996. (page 8)

[37] A. Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001. (page 5)

[38] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000. (page 6)

[39] H. Strobelt, D. Oelke, C. Rohrdantz, A. Stoffel, D. Keim, and O. Deussen. Document cards: A top trumps visualization for documents. *IEEE Transactions on Visualization and Computer Graphics*, 15:1145–52, 11 2009. (page 3, 9, 10)

[40] D. Suleiman and A. A. Awajan. Deep learning based extractive text summarization: Approaches, datasets and evaluation measures. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 204–210, 2019. (page 16)

[41] H. Sun and S. A. Jafar. The capacity of robust private information retrieval with colluding databases. *IEEE Transactions on Information Theory*, 64(4):2361–2370, 2017. (page 7)

[42] L. Tamine-Lechani, M. Boughanem, and M. Daoud. Evaluation of contextual information retrieval effectiveness: overview of issues and research. *Knowledge and Information Systems*, 24(1):1–34, 2010. (page 2, 6, 7)

[43] J. Tan, X. Wan, and J. Xiao. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, 2017. (page 16, 17)

[44] J. J. Thomas, P. J. Cowley, O. Kuchar, L. T. Nowell, J. Thompson, and P. C. Wong. Discovering knowledge through visual analysis. *Journal of Universal Computer Science*, 7(6):517–529, 2001. (page 14)

[45] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1–24, 1989. (page 5, 6)

[46] T. Uçkan and A. Karcı. Extractive multi-document text summarization based on graph independent sets. *Egyptian Informatics Journal*, 21(3):145 – 157, 2020, `http://www.sciencedirect.com/science/article/pii/S111086651930324X`. (page 35)

[47] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606 – 1618, 2007, `http://www.sciencedirect.com/science/article/pii/S0306457307000507`. Text Summarization. (page 34, 35)

[48] P. Venetis, G. Koutrika, and H. Garcia-Molina. On the selection of tags for tag clouds. In *Proceedings of the fourth ACM International Conference on Web Search and Data Mining*, pages 835–844, 2011. (page 11)

[49] F. B. Viégas and M. Wattenberg. Timelines tag clouds and the case for vernacular visualization. *Interactions*, 15(4):49–52, 2008. (page 9)

[50] M. Wattenberg and F. B. Viégas. The word tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1221–1228, 2008. (page 12)

[51] P. C. Wong, B. Hetzler, C. Posse, M. Whiting, S. Havre, N. Cramer, A. Shah, M. Singhal, A. Turner, and J. Thomas. In-spire infovis 2004 contest entry. In *IEEE Symposium on Information Visualization*. IEEE, 2004. (page 14)

[52] W. Zeng, A. Dong, X. Chen, and Z.-l. Cheng. Vistory: interactive storyboard for exploring visual information in scientific publications. *Journal of Visualization*, pages 1–16, 2020. (page 1, 13, 14)