TU WIEN Informatics

Concept Map Mining als Browserweiterung

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Mario Stoff
Matrikelnummer 11777706

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Ass. Dr.techn. Manuela Waldner, MSc

Wien, 22. Oktober 2021

Mario Stoff	Manuela Waldner

TU WIEN Informatics

Concept Map Mining as Browser Extension

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Mario Stoff
Registration Number 11777706

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Ass. Dr.techn. Manuela Waldner, MSc

Vienna, 22nd October, 2021

Mario Stoff	Manuela Waldner

Erklärung zur Verfassung der Arbeit

Mario Stoff

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 22. Oktober 2021

Mario Stoff

Kurzfassung

Concept Maps sind ein wohlbekanntes Mittel um vorhandenes Wissen strukturiert darzustellen. Sie werden durch ein Node-Link-Diagramm repräsentiert, welches verschiedene Konzepte und ihre Verbindungen zueinander aufzeigt. Diese Elemente werden meist aus unstrukturierten Texten gewonnen. Eine Concept Map händisch herzustellen kann mühsam sein, aber vollständig automatisierte Lösungen erlangen oftmals nicht die gewünschte Qualität. Halb-automatische Herangehensweisen hingegen haben sich oftmals als günstigen Kompromiss zwischen diesen beiden Ansätzen herausgestellt. Bei diesen halb-automatischen Ansätzen ist es besonders wichtig, dass der manuelle Aspekt der Erstellung möglichst einfach und intuitiv gestaltet ist, um den effizienten Arbeitsfluss der Benutzerinnen und Benutzer nicht zu unterbrechen. Aus diesem Grund spielt die grafische Nutzeroberfläche eine entscheidende Rolle wenn es darum geht eine effiziente Arbeitsweise und befriedigende Ergebnisse zu ermöglichen.
Es ist daher das Ziel dieser Arbeit, eine Umgebung zu erschaffen, in der Benutzerinnen und Benutzer die Möglichkeit haben, sinnvolle Konzepte aus beliebigen Webseiten zu extrahieren und diese Konzepte mit bestehenden Wissensstrukturen zu verbinden. Mit anderen Worten, es soll visuell hervorgehoben werden, wie neue, unbekannte Information mit einem bestehenden Wissensstand zusammenhängt. Zu diesem Zweck wird hier eine Browsererweiterung vorgestellt, welche Benutzerinnen und Benutzern ermöglicht den Textinhalt beliebiger Webseiten mit einer Software zur natürlichen Sprachverarbeitung zu analysieren. Konzepte und Relationen werden dann direkt im originalen Text hervorgehoben. Den Benutzerinnen und Benutzern steht es dann frei, diese neuen Konzepte und Relationen zu einer bestehenden Concept Map hinzuzufügen. Concept Maps können auch automatisch als Node-Link-Diagramm dargestellt werden. Mit der Hilfe einer kleinen Benutzeruntersuchung kommen wir zu dem Schluss, dass diese Anwendung definitives Potenzial aufweist, aber aufgrund von einigen Mängeln, vor Allem im Bereich der automatischen Textverarbeitung und der Konzept Erkennung, noch nicht für einen realen Anwendungsfall geeignet ist.

Abstract

Concept maps are a well-known method of structured representation of knowledge. They are represented as a node-link diagram that showcases different concepts and their relations to each other, which are often extracted from unstructured text. Manual generation of such concept maps can be a tedious task, but fully automated approaches are often not able to satisfy qualitative expectations. Semi-automated methods have shown to be a satisfying compromise between these two. It is especially important that the manual aspect of creating a concept map is as intuitive and easy as possible so that the user's workflow is not interrupted, and tasks can be completed efficiently. Therefore, it is the graphical user interface that plays a critical role in guaranteeing a satisfying experience and swift completion of tasks.

It is therefore the aim of this thesis to create an environment, in which users are able to extract meaningful concepts from arbitrary websites and connect these concepts to existing knowledge structures. In other words, it should visually convey how new, unseen information fits to the knowledge they already have. For this purpose, an extension to the Google Chrome browser is presented in this thesis, that allows the user to analyze the text on any website on the internet with a provided natural language processing software. Concepts and relations can then be highlighted in the original text to visualize their connection to existing knowledge. At the user's choice, new concepts, relations, and combinations of the two can be added to existing concept maps. These concept maps can be automatically visualized as a node-link diagram.

With the help of a small user evaluation, we conclude that the approach has definite potential but still lacks the reliability, especially with the automatic text processing and concept extraction, for real-world use-cases.

Contents

CHAPTER 1

Introduction

In many professions, workers are required to conduct thorough research on diverse topics - for example, investigative journalists, researching and fact-checking their latest piece, or criminal investigators, trying to find connections between evidence. Their goal is most of the time to gain new knowledge and connect the newly learned information to already known facts. These tasks often require reading many different sources, as the required knowledge is often scattered across many places on the Internet or other digital media. Sometimes, connections between ideas are not always obvious across different documents and they could easily be overlooked. When reading a lot of sources on one topic, researchers will often encounter redundant information which is time-consuming and does not yield any real benefit to the task of making sense of the topic. There exists a need to quickly and visually link information in an online text to already explored knowledge domains and to make permanent connections.

Concept maps have been proven many times to be very useful in representing someone's knowledge of a certain topic or focus question and to help with tying new evidence to existing knowledge structures. But the manual construction of a concept map can be tedious and is often not as easy as it might appear. Technical solutions have been developed to aid in the construction of these maps, but most of them only facilitate the mechanical part of creating the map. Only a few provide actual help by offering meaningful suggestions as to what further concepts might be included to extend the concept map. And if these functionalities are available, it is often only in dedicated software clients, where an input text needs to be fed to an application in order to be analyzed.

1.1 Aim of the Work

With this thesis, we aim to find out if we can apply a semi-automatic concept map mining approach on the text of arbitrary web pages in the user's browser. We want to

explore whether concept and relation suggestions received from a text processing tool can be easily highlighted within the source page itself, and whether the connections between them can be visualized in a satisfying manner. Users should be able to create sophisticated concept maps directly in their browser without the need to leave the website they are researching on and without the need of any external tool or software. We want to enable them to quickly identify redundant information, as well as any new information that relates to to their previously acquired knowledge. With this approach, we aim to make the process of analyzing source material and generating valuable suggestions fully automatic. Users should be able to make use of these suggestions and create concept maps without much manual interference.

1.2 Methodology

To achieve this aim, the contribution of this thesis is a browser-embedded User Interface that can analyze online text and extract concepts and relations through Natural Language Processing and Concept Map Mining. After the text analysis, the tool presents the user with the results. It shows the user what concepts within the text are already contained in their existing concept map. It can also highlight new concept suggestions that directly relate to the already known concepts, or present the user with entirely new concepts that it extracted from the text. All these possible indications are highlighted through differently colored background boxes around the concept words or phrases directly in the text. Through hover and click interaction, a user can discover new relations and make additions to their existing concept map.

1.3 Structure

In Chapters 2 and 3 we visit the theoretical background of concept maps and concept map mining, and related work on user interfaces and used techniques. Chapter 4 describes the user interface and functionalities developed for this application. Chapter 5 deals with the implementation of the suggested solution. In Chapter 6, we address the weaknesses and limitations of our approach. In Chapter 7, a use-case is described as a walkthrough example and a user evaluation is conducted and discussed. Finally, in Chapter 8, we summarize this thesis and discuss possible future work on this topic.

CHAPTER 2

Background

In the Background chapter, we explain the basic concepts that are essential foundations to this work but are not directly related to the work itself. Section 2.1 covers the definition of Concept Maps and their most prominent use-cases. Section 2.2 focuses on the process of Concept Map Mining (CMM), with a special focus on the semi-automatic CMM software developed by Presch [Pre20].

2.1 Concept Map

The idea of a concept map originated in a 1972 research program by J.D. Novak and Dismas Musonda [NM91]. In this 12-year program, the researchers tried to follow children's understanding and knowledge of science. The program was based on the cognitive learning theory by David Ausubel [Aus63], which describes the structure of knowledge in human brains as a combination of concepts and propositions. New concepts and propositions need to have connections to the learner's existing knowledge framework in order to be learned efficiently. To better represent the children's knowledge in his program, Novak developed concept maps.

In a later report [NC06], Novak and Cañas defined a set of guidelines on "Constructing Good Concept Maps".

1. They start the process of constructing a concept map by establishing a context. This is best accomplished by defining a *Focus Question*. The focus question should set clear boundaries for what the concept map contains and limit the scope to the problem or issue that the concept map should help to resolve [NC06].
2. Once the domain and focus question are established, about 15 - 25 key concepts for this question can be identified. These concepts are then ranked from the most general/inclusive at the top of the list to the most specific concept at the bottom.

This list should only function as an approximation, as concepts will or will not be added to the concept map later on.

3. After that, concepts can be gradually added to a preliminary concept map and relations can be formed. Concepts can be placed at the level of hierarchy where they seem to fit best in the current context. Novak and Cañas [NC06] recommend the use of post-it notes or a software tool, as that makes it easier to move concepts around.

4. The next step is to identify *cross links*. These connect different sub-domains of the concept map to each other and show the creator's understanding of the domain. This understanding is important to avoid unwanted phenomenons like full sentences as concepts or "string maps", i.e. long chains of concepts and relations where only one always follows another.

5. As a final step, it is important to realize that a good concept map needs multiple revisions. Concepts can be rearranged to increase clarity. New concepts can be added and unnecessary ones removed. Novak and Cañas write that a good concept map usually needs at least three revisions.

As we will be described later on, the process to construct a concept map used in this thesis diverts from this model method. For a start, this is due to the fact that the purpose of the map creation itself is a little different. Novak and Cañas want to visualize a fixed amount of knowledge in a well constructed concept map. The aim of our approach, on the other hand, is to expand existing knowledge through research and exploration of text. It is therefore not feasible for us to start out with defining a set amount of key concepts, as that would defeat the entire purpose of our application. Relations and cross links should also arise naturally throughout the research process with our procedure. We do, however, also strongly encourage to revise and edit the concept maps in progress, as the unstructured expansion that comes with our approach can easily lead to obsolete concepts or double mentions of relations.

2.1.1 Definition

Based on Novak's definition and quality measures [NGB84], Villalon and Calvo formally described a concept map as follows [VC08]:

Concept Maps are a triplet $CM = \{C, R, G\}$, where C is a set of concepts $C = \{c_1, c_2, ..., c_n\}$ each being a noun or phrase that is unique in C. R describes a set of relations $R = \{r_1, r_2, ..., r_n\}$ where each relation is again a triplet $r_i = \{c_p, c_q, l_i\}$ with c_p and c_q being concepts from C and l_i being a label phrase. The generalization levels in G each correspond to a distinct set of concepts from C. $g_i = \{c_1, c_2, ..., c_s\}$. Generalization levels are ordered in the way that for two levels g_i and g_j, g_i is more general than g_j only if $i < j$.

2.1.2 Applications of Concept Maps

In his Diploma Thesis "Semi-Automatic Creation of Concept Maps"[Pre20], Presch describes several different use cases for concept maps. He states that this list is *"not to be seen as a complete enumeration of all existing applications, but rather give a brief overview of what is possible."*[Pre20]

Education: As already mentioned, education is the field that concept maps originated from as a method of representing the knowledge of students over a certain topic [NM91]. They can be used as either material for studying or a means of evaluation of a student's knowledge, where a teacher can compare a student's concept map to one created by an expert [NC07].

Visualizing Expert Knowledge: In expert systems, concept maps can be a powerful tool to visualize the detailed knowledge an expert has on a certain topic. As one example of such expert level concept maps, Hoffman et al.[HCFC01] utilized concept maps to *"create a knowledge model of weather forecasts in the gulf coast region"* [Pre20].

Storytelling: Under the guise of "Storytelling" many applications can be summarized. These can include use cases, where a concept map is used to show certain connections in a topic in a visual manner [Pre20]. One such example is the use of concept maps by The International Consortium of Investigative Journalists to visualize the connections between leaked documents of offshore companies and the individuals involved in the affairs [PAN].

Organization and Planning: Another way to use concept maps is for planning and collaborating on shared knowledge [Pre20]. In the context of project planning in libraries, Colosimo and Fitzgibbons [CF12] use concept maps for a workshop where they organize files and access resources. Resources are directly attached to concepts and can be accessed from the concept map.

Information Retrieval and Summarization: Other than visually representing textual data, concept maps can also be used as a basis to retrieve information and sum up the results. Taking advantage of the propositional and hierarchical nature of concept maps, Carvalho et al. [CHC01] developed algorithms that could filter and rank the relevance of results of search engines based on the structure of the concept map.

The solution that we have developed could potentially be utilized in more than one of these application areas. Our tool could be used by students in the education field to create concept maps while they are exploring and gathering information on a subject. They could then show their teachers that they have dealt with different materials and understood the subject with their resulting concept map. For concept maps that classify as "storytelling", there is potentially a lot of research and exploration of unknown sources involved in the process. That is where the proposed application particularly shines. On the other hand, it would probably find less use for visualizing expert knowledge. That kind of task would not entail much gathering of new ideas, but rather portraying what

the expert already knows. While this would still be possible with our application, it is not tailored to that task and there would be other programs that better fit their needs.

2.2 Concept Map Mining

Concept Map Mining describes the extraction of concepts, relations, and levels of generality from text. There are different approaches and classifications of those approaches. The term "concept map mining" has first been introduced by Villalon and Calvo [VC08] in 2008 and further classified by Kowata et al.[KCB10] in 2010.

This thesis is built on the semi-automatic concept map mining application developed by Presch [Pre20] for his master thesis. His application uses a state-of-the-art Natural Language Processing pipeline to analyze an input text and derives possible concept and relation suggestions. In his work, he simplify the mentioned definition of a concept map by omitting the generalization level requirement and declaring a concept map $CM = \{C, R\}$, consisting only of concepts and relations, as a valid result. Therefore, the resulting concept maps of this thesis' application also do not provide a hierarchical structure.

Presch's semi-automatic concept map mining process [Pre20] consists of two distinct parts:

1. **Automatic text processing** takes care of generating concept and relation suggestions. It takes a simple, clean text string (i.e. without any special characters) as input and through the subsequent execution of multiple algorithms, it reaches the desired result.

   It starts with a "Linguistic Pre-Processing" step, where typical steps of a Natural Language Processing pipeline are applied to annotate the input text. Figure 2.1 shows the structure of such a typical pipeline.

   The process starts with "tokenization" of the text, which is the process of splitting a text into its most basic components, like words, punctuation marks, numbers and other symbols [HHL19]. After that, "sentence segmentation" is applied, where a text is split into individual sentences or chunks of information [HHL19]. These sentences are usually sufficiently detailed to form *subject-verb-object* triples [HHL19]. With "part-of-speech tagging" the previously segmented parts of sentences are annotated with their grammatical role in these sentences. Words are, for example, tagged as nouns, adjectives or verbs. This can be seen in Figure 2.2. An important part for this CMM approach is the processing step of "named entity recognition". With this step, previously annotated tokens are tagged with additional categories that relate them to things in the real world. These categories can classify a token as representing a person, location, organization, a date, or others. This extra classification helps with extracting concepts later on. In the next steps "Coreference Resolution" is applied, which is a process that detects anaphora (i.e. usage of the same phrase in subsequent sentences) (see Figure 2.3).

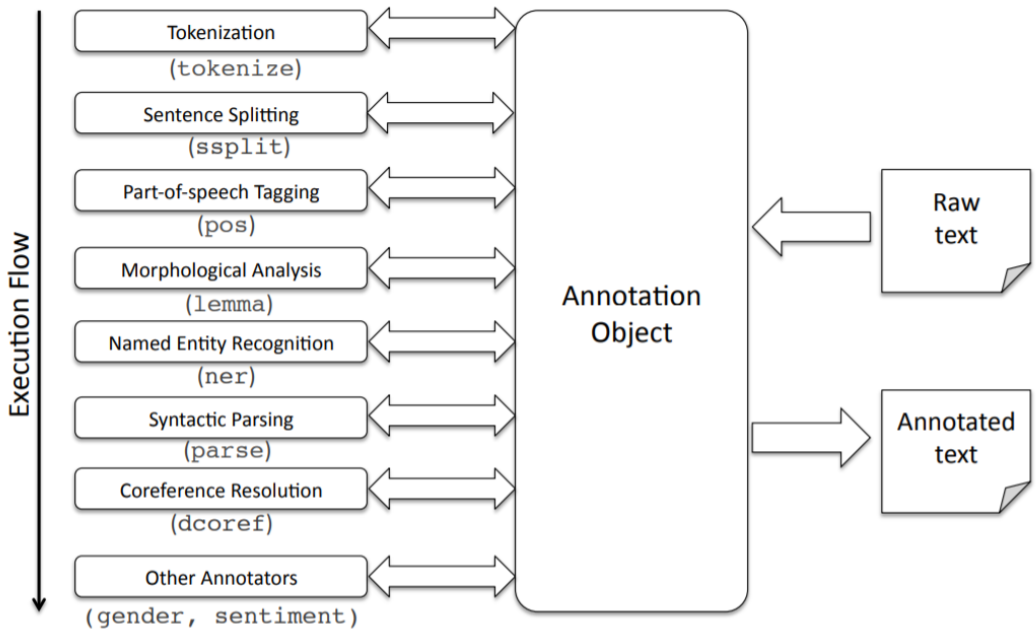


Figure 2.1: A state-of-the-art NLP pipeline from The Stanford CoreNLP [MSB+14].

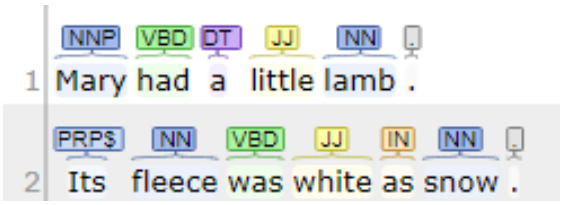


Figure 2.2: Part-of-speech tagged text with Stanford CoreNLP [MSB+14].

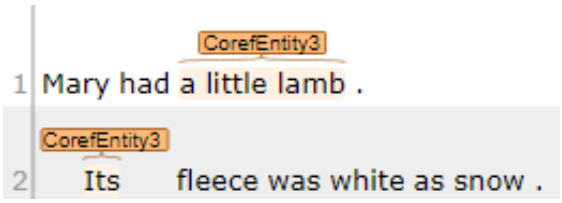


Figure 2.3: Coreference Resolution with Stanford CoreNLP [MSB+14].

This makes it easier to detect multiple occurrences of the same concept. After that, similar concept-mentions are put into groups. Finally, concepts are extracted from the previously generated concept mention groups and relations are generated from verb phrases identified in the text.

2. For his **Manual Concept Map Construction**, Presch created a prototype web application that displays the concept and relation suggestions from the previous step to the user. The user can then select desired concepts to add them to a visual graph representation of a concept map. Suggested relations and related concepts can be automatically shown by selecting any concept in the concept map and subsequently also included in the concept map graph. Used concepts are highlighted in the "text area", where the submitted text is presented.

The work in this thesis makes use of Presch's automatic text processing as a first step to analyze the page text of arbitrary sites on the World Wide Web before further processing the results.

CHAPTER 3

Related Work

The Related Work chapter focuses on other available concept map user interfaces and what they can or cannot do in comparison to our implementation. We also take a look at common highlighting techniques and visual linking and how they are used in visualization literature.

3.1 Concept Map User Interfaces

The construction of concept maps has, over the last couple of decades, evolved from simple pen and paper drawings to the use of elaborate digital tools. Today, there are many tools available that allow users to create concept maps from scratch, redesign and rearrange them, and also connect them to further resources to maximize the information they represent. Different tools have different capabilities. The following section gives an overview of different tools and what they are capable of.

3.1.1 CmapTools

In their guidelines for constructing good concept maps, Novak and Cañas [NC07] also recommend a software toolkit called CmapTools [CHC+04]. This program was designed as a client-server based software kit at the Institute for Human and Machine Cognition (IHMC) in Florida to create concept maps and share and collaborate during that process. It is designed with the use in education and the scientific field in mind. CmapTools is still being developed and extended to this day.

CmapTools aims to provide a low threshold for creating concept maps while at the same time maintaining a high ceiling of functionality. In other words, it should be easy for even a child to construct a basic concept map, but it should also be possible for experts to create elaborate and graphically more sophisticated maps. With CmapTools, users can connect their concept maps with many types of resources all across the internet, be it

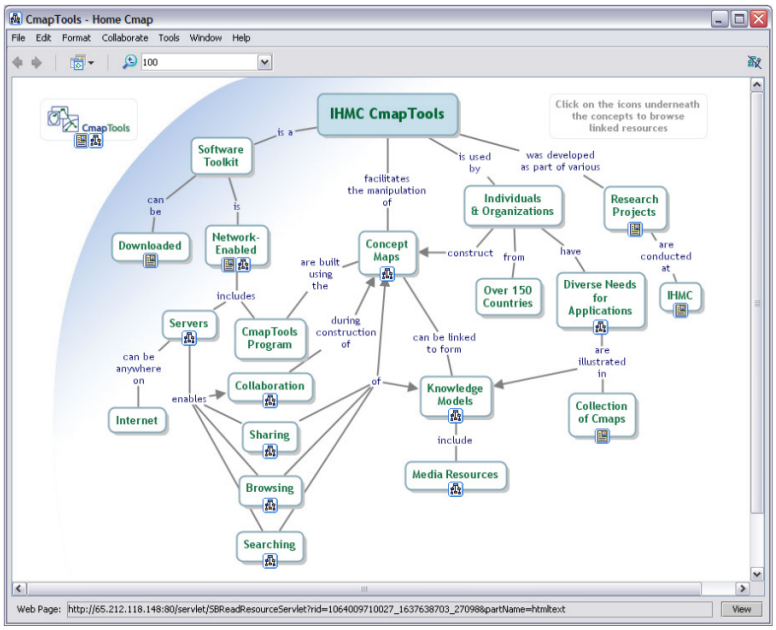


Figure 3.1: Concept Map about CmapTools in CmapTools [CHC+04].

images, videos, sound clips, texts, etc., or even other concept maps. Like this, it is possible to establish large knowledge models across many interlinked concept maps. Using this system, it has been found that concept maps are quite efficient for browsing large domains of knowledge compared to the page-based approach on the World Wide Web [CDC+01]. CMapTools allows its users to display multiple linked resources at the same time, making it easy to investigate related materials. Everything can be drag-and-dropped around, all the while maintaining or updating established links.

One very interesting feature the program offers is a concept suggester that mines the web for relevant concepts [LMR04]. It analyses the concept map under construction and provides the user with concept suggestions it deems relevant to the context of the map. Due to the fact that the tool is mining the Internet without an actual understanding of the map, it is understandable that not all suggestions are relevant, but the developers argue that even two or three meaningful suggestions can trigger further ideas in the creator of the map. A similar phenomenon can be seen with the concept map mining applied in our project. Although our approach extracts concepts from a single source at a time, it still is likely to produce irrelevant suggestions.

Another big aspect of CmapTools is the public sharing of knowledge. Everything can be published to public servers all across the world where maps and resources can be shared and linked to each other. The open public server approach of this toolkit was very unique at the time of its creation.

As part of the CmapTools software suite Cañas et al. [CCL18] have developed a web embeddable concept map editor that brings the functionality of CmapTools to the Browser. Apart from a number of advanced features, this new tool, called eCmap, works similar to the desktop client. In contrast to this web app approach, our work is implemented as a browser plugin and can therefore analyze any content the user is currently visiting.

Other than this very specific and sophisticated solution, there are numerous applications available for free on the Internet. These are, however, in most cases not tailored to creating concept maps, but rather general tools for drawing graphs and diagrams. Nonetheless, all upcoming tools have the capability to create concept maps that fulfill the definition's requirements.

3.1.2 Basic Editors

Some tools can be very simple, yet quite effective when it comes to creating concept maps. Such basic editors, like the Visual Paradigm Editor [VPE] or MindMup [MIN], do not offer many extras, but they are practical enough when a user only wants to create a simple concept map quickly and easily. Both these tools allow to place boxes in different shapes (Visual Paradigm) or just one default shape (MindMup) for concepts, and to link these concept boxes with labeled lines. That is enough to form a basic concept map. Visual Paradigm has a lot of options to style nodes and links.

3.1.3 Collaborative Tools

Apart from the bare basics that are needed to design a concept map it can be very useful in many use-cases to work together with other people on the same concept map. Fortunately, there are quite a few online tools available that fully operate on the cloud and are built for collaborative work. Most of the following pieces of software are not specifically developed for creating concept maps but rather for many different kinds of diagrams and charts. But they all offer decent capabilities to design diagrams that adhere to the definition of concept maps.

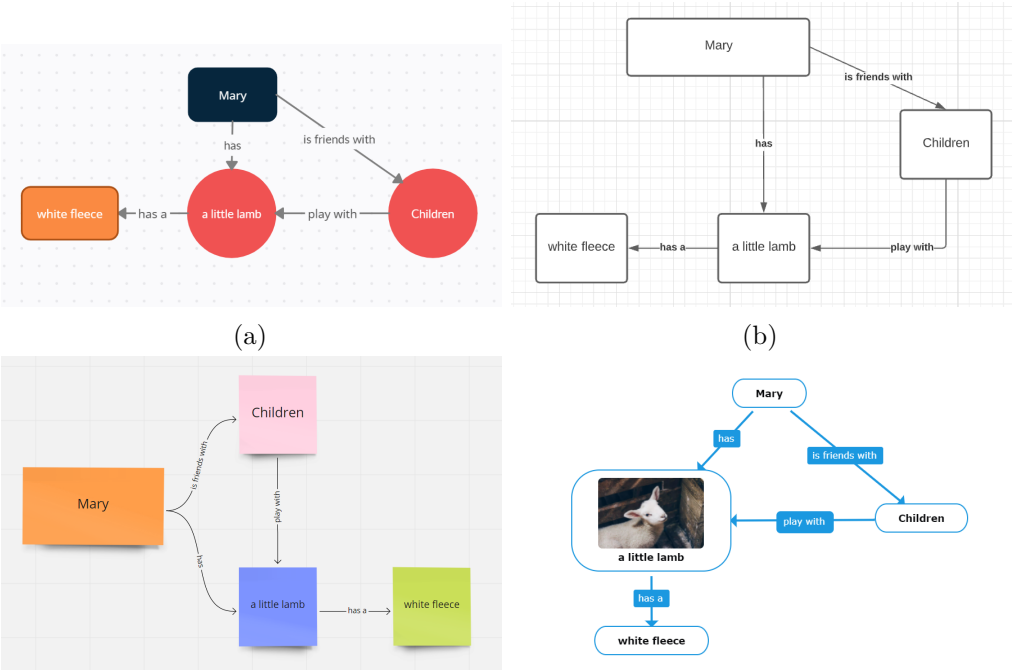
Creately, Lucidchart, Miro

Creately [CRE], Lucidchart [LUC], and Miro [MIR] are one step up of basic editors. For concept mapping, they offer everything the previous tools had as well as attaching hyperlinks to concepts. But the bigger distinction is the ability to invite other people to work on the same document in real-time (however, this is a premium feature with Lucidchart). With all three of them, users are able to post comments for collaborators to see and discuss. When using a Miro whiteboard, users can also communicate directly via an integrated text chat feature. For even better collaboration, Miro and also Creately (beta) offer live video conferences among team members.

Mindomo

Mindomo [DOM] is another application where users can create concept maps with little effort and work together (premium feature). But one thing distinguishes this tool from

the previous ones. Like CmapTools [CHC+04], Mindomo allows to attach a variety of media elements to the concept boxes. All these attachments (videos, images, audio) can either be embedded from other websites, or uploaded directly from the user's computer or their cloud storage like Google Drive or Dropbox. It does, however, not have any easy, built-in means of communication with other users.



(a) (b)

(c) (d)

Figure 3.2: Concept Maps created with (a) Creately (b) Lucidchart (c) Miro and (d) Mindomo.

CLIP Tool

The CLIP tool, described by Mahyar and Tory [MT14], is a Java-based tool that has its focus less on merely creating a visual graph, but is focused specifically on supporting a collaborative sensemaking task. With this tool, multiple users researching the same case can share and discuss their individual findings in a shared space. Each user can create their individual concept graph, where they visualize their own evidence as nodes and add relations between them. A user gets notified if another user has similar or related evidence in their concept graph. At any point of their investigation, users have the opportunity to merge other user's graphs with their own to incorporate their findings into their own. Any nodes or edges in these graphs can be attached with text notes, references to the original source, time stamps, and others. Nodes created by different collaborators appear in different colors to easily distinguish between who provided what evidence. This also visualizes what evidence has been found by multiple users. A time-line feature is also incorporated in the tool to make it possible to later retrace the expansion of the

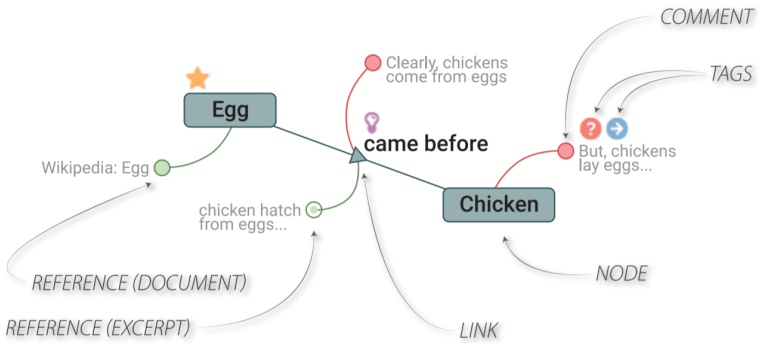


Figure 3.3: Example of annotated concepts and relation with KTGraph [ZGI+17].

concept graphs.

Knowledge-Transfer Graph (KTGraph)

The knowledge-transfer graph by Zhao et al [ZGI+17] is also a tool that is intended for collaborating in investigative sensemaking tasks. But in contrast to the CLIP tool, it is not designed for real-time collaboration and merging user's individual graphs, but rather handing off one user's partial findings to another person in an asynchronous type of collaboration. With this tool, a user can create a standard concept graph, where different concepts are connected with relation phrases. These concepts and relations can then be annotated with many different assets that should accentuate the user's reasoning, ideas, and train of thought. These include embedded references to source material, comments, and visual tags that signify different meanings like open questions, to-dos, importance, or posed hypotheses (Fig 3.3). Every step of the graph's creation is recorded to a timeline panel that can be used to play back the evolution of the graph step by step. All these accessories to the concept graph are intended to make it easier for subsequent investigators to follow the previous user's thoughts and findings and to quickly catch up to the current state of the investigation and to find anchor points where they themselves can push the investigation forward.

3.1.4 Browser-based Sensemaking Tools

The following tools are not equipped to create concept maps per se, but they are in their essence very similar to the application that was developed for this thesis. Both these tools aim to improve the sensemaking process for users by representing websites and web resources as elements in a graph and signifying relation between them with directed links.

ScratchPad

The "ScratchPad" by David Gotz [Got07] is also an extension to a browser interface. It is integrated as a sidebar next to the main browser display area (Fig. 3.4). With it, a user can "snapshot" the page currently active in the browser and put a graphical representation of that website into their ScratchPad. They can also drag-and-drop information of a finer

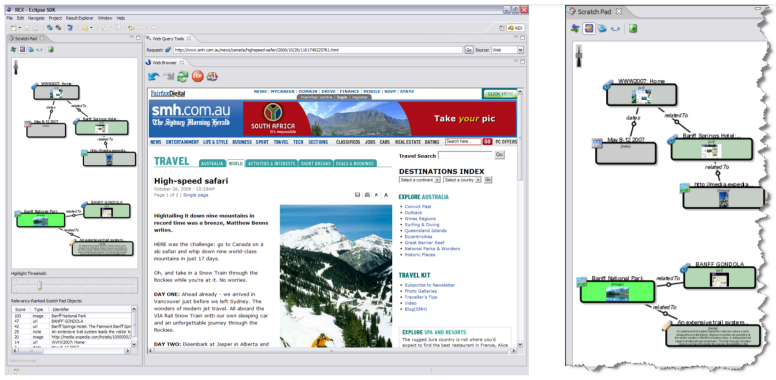
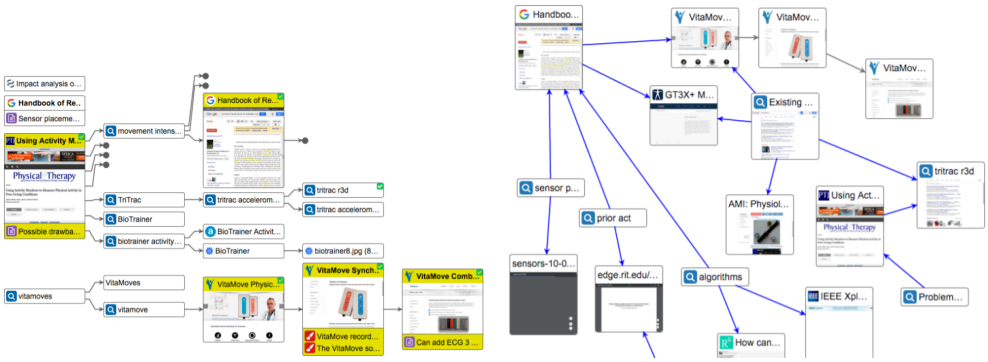


Figure 3.4: A ScratchPad-enhanced web browser and a close-up of the ScratchPad sidebar [Got07].

granularity from a web page onto the ScratchPad, or create new object nodes directly on the ScratchPad itself. These objects can be modified any time. Object nodes can also be linked together to express relationships. Unlike concept maps, these relations are not labeled and bear no distinct, additional meaning other than a relation of any kind between these objects.

SenseMap



(a) (b)

Figure 3.5: (a) History Map and (b) Knowledge Map with SenseMap [NXB+16].

The "SenseMap" application by Nguyen et al. [NXB+16] has its main use in visualizing the sequential steps taken throughout a sensemaking task. Browser actions, like entering search terms and following links, are automatically recorded to a "History Map" (Fig. 3.5a). While doing their research, users have the ability to add highlights and written notes to any website they visit. This additional information gets automatically attached to the page's representation in the history map. This history map can be viewed as a node-link graph, where search terms and visited pages are represented as nodes and the

links between them signify the path a user has taken to get to a certain page. The user can also access a different representation of this map, called a “Knowledge Map”, where the information from the history map can be curated to better suite the users needs (Fig. 3.5b). Nodes can be spatially organized and links can be manually added or removed.

3.2 Highlighting

There are many well-known ways to highlight certain words or phrases in a text e.g. bold typeface, underlining, italics, or background coloring [SOK+15]. Different highlighting techniques work better for certain types of data. For example, categorical data can be highlighted well with hues of text or background color, underline styles, borderline styles, or font families [SOK+15]. Quantitative data on the other hand is better emphasized through font size, luminance, thickness, or degree of letter spacing [SOK+15]. In their user studies, Strobelt et al. found out that highlighting with background color lies in the top three techniques in regard to detectability (or popping out) for all their scenarios.

Perception theory shows that multiple visual low-level features can interfere with each other [HE99]. This was also observed in the user studies [SOK+15]. The strongest distractors for background coloring were identified as underlines and italics.

Strobelt et al. [SOK+15] write in their paper that most text analysis systems employ background coloring as a highlighting technique and that this approach creates problems once multiple overlapping annotations occur. This can also be observed in this work and will be further addressed in a later chapter.

An example application that uses span elements with different background colors to mark concepts and relations is the web-based text annotation tool called “Anafora” [CS13]. With this tool, users can annotate a text with special highlights, whose properties are predefined in XML-schemas. These highlights can represent different types of entities (or concepts) and relations between those entities.

Another tool where concepts and relations can be highlighted in a text document is called “Egas” [CLN+13]. It is a web-based platform designed to annotate biomedical text. Concepts and relations can be manually annotated or received through calling an automatic document annotation service. With this tool, concepts are also highlighted as colored boxes and relations as lines between them, tagged with the type of relation (Fig. 3.6).

3.3 Visual Links

Visual links have been used in numerous applications and research papers in order to show that two or more entities share a connection. Studies have shown that connecting entities with visual links generates an even stronger grouping effect than other design principles like similarity, color, shape, or size. Ziemkiewicz and Kosara [ZK10] describe in their study the perceived attracting force that different grouping methods have on

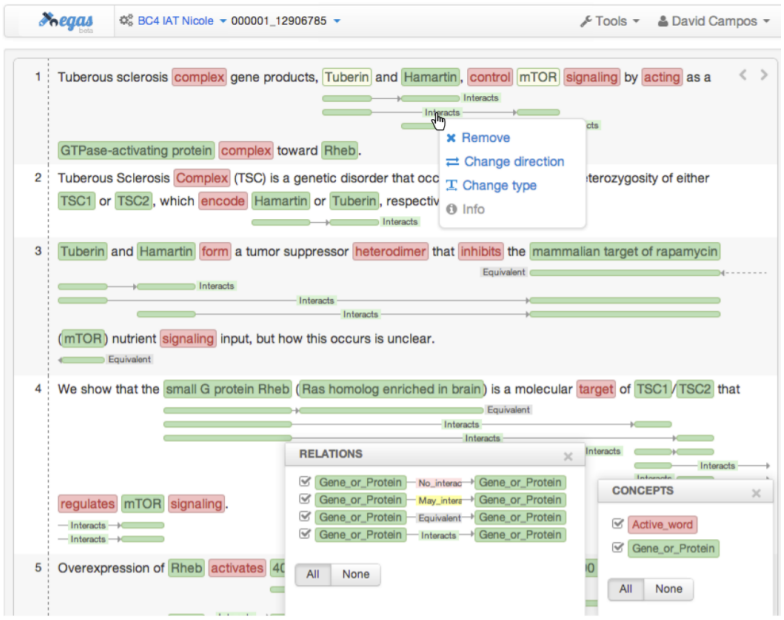


Figure 3.6: Egas main user interface [CLN+13].

the remembered position of elements. They found that connecting lines, along with outlines around elements, let test participants remember two elements closer together than they actually were. Palmer and Rock present in their study [PR94] the role of uniform connectedness. In one experiment they show how two targets that are connected with a straight line appear to be grouped together, even though each target is either closer in position, more similar in size, or a combination of both to another disconnected target.

Visual links show particular effectiveness over mere color highlighting when connected items are outside of the active visual field [HBW08]. That is why they are often used in applications that involve connecting information across multiple windows and large displays.

In a paper by Waldner et al. [WPL+10], visual links are utilised to connect pieces of information that are selected by the user across multiple windows with the help of a central window management application. This application is targeted to help information workers analyze information across multiple sources. A similar visualization, specifically created for collaborative information exploration, is presented in a paper titled "Collaborative information linking: Bridging knowledge gaps between users by linking across applications" [WS11]. Information is once again connected across multiple application windows with the additional feature of user-specific visual links. In "Context-preserving visual links" by Steinberger et al. [SWS+11], an improved process of linking related elements across windows is presented. Connecting lines are rendered in a way to minimize occlusion of relevant data and to make them very distinct from the elements

they cover. Analyst's Workspace [AN12] is a sensemaking environment which is designed to utilize the available space on large displays to perform an investigative data analysis. This application encourages an entity-centric approach to data analysis. Visual links are employed to connect entity occurrences in multiple displayed documents. An approach that is a little different is presented in a paper by Geymayer et al. [GSL+14]. Here, visual links are used not only to show connected information that is visible on large screens, but also related information that is occluded by other windows or otherwise outside the user's viewport. Finally, visual links are used in a 2021 article by Waldner et al. [WGSS21] to connect evidence in another visible window to concepts in an observation graph.

In our thesis, we do not have items of interest across multiple frames or applications. Our entities are most likely positioned in each others proximity, but we still want to make use of the strong connective property that visual links provide. The links that are generated with our tool represent relations between entities that were automatically extracted from the text and are therefore used to immediately draw the user's attention to related entities that might be of interest.

CHAPTER 4

Method

As mentioned before, this thesis uses the work by Presch [Pre20], specifically, the automatic text processing software that produces concept and relation suggestions from input text. The aim of this thesis is to create a Front-End Interface that can apply this automatic text processing on arbitrary text on the Internet and present the user with potentially meaningful extensions to their existing knowledge. We want to take the concept and relation suggestions from the server and display them right in the place where they originate from, on a website on the Internet. From there, the user should be able to pick the concepts and relations they like and permanently add these new propositions to an existing concept map. We want to highlight concepts that are known, i.e. contained in an existing concept map, as well as entirely new concepts contained in a website's text, and mark relations between known and new concepts in a prominent way.

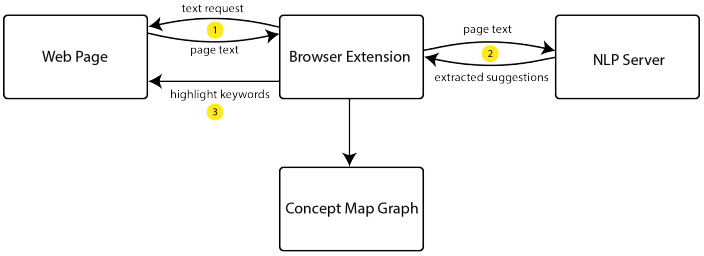


Figure 4.1: Schematic of the text analysis process.

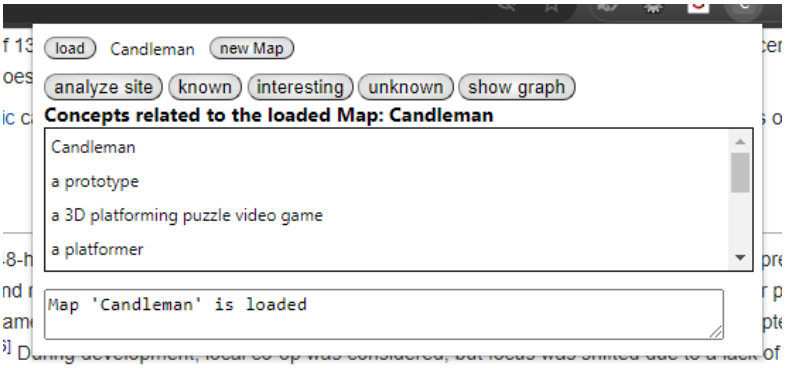


Figure 4.2: The extension popup window.

For this purpose, we have built an extension to the Google Chrome browser. This extension manages the connection to Presch's Concept Map Mining Back-End and all the highlighting that is produced through interaction (Fig. 4.1).

There are three distinct components, or areas of interaction, to this application:

- The Chrome extension popup,
- the target website, and
- the concept map graph.

The extension popup functions as the main hub from which the application is controlled (Fig. 4.2). The website that the application is used on works as a carrier for all the highlights and for interactions that extend the concept map. Finally, the concept map graph, available through the extension popup, visualizes the acquired knowledge in the form of a node-link-diagram and allows for manual adjustments to the concept map.

4.1 Interface

After the successful installation of the extension, an icon for the application becomes available in the browser's extension menu. Clicking this icon opens the extension's popup window (Fig. 4.2) from which all functionalities of the application are controlled. On the initial activation, there are seven buttons and an empty text field available. At startup, only three of the seven buttons can perform their action as no concept map is loaded into the application and the website's text has not yet been analyzed. The button labeled "load" reveals all the different concept maps that are available in the browser's storage. Selecting one of the available options will load that map into the application and all future interaction will affect this loaded concept map. The name of the loaded map appears next to the load button so that the user always knows which of their concept maps they are working with at the time.

If no existing concept maps are available, or if the user wants to create an entirely new map, the button labeled “new Map” produces an input field where a name for the new map needs to be entered. This creates a completely blank concept map ready to be filled with new concepts and relations.

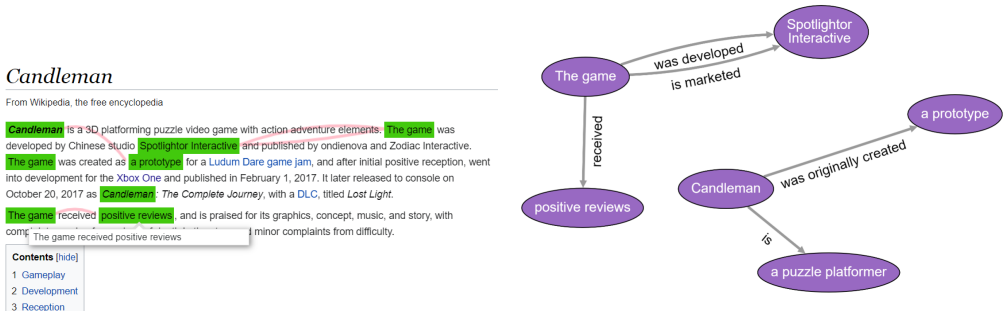
The third button that works right away is labeled “analyze site”. Clicking this button will take all the text paragraphs on the current web page and send them to the automatic text processing back-end server. After the back-end server finishes processing the text and returns the results, immediate change is visible. On the popup window, a list of concepts appears below the line of buttons (Fig. 4.2). It shows the concepts that are either contained in the loaded map itself or that are related within the text of this page to a concept contained in the loaded map. At the same time, concepts that are already contained in the loaded map are highlighted with a green background color within the website’s text. Hovering the mouse over these highlights reveals relations to target concepts that the highlighted concept might have. Clicking target concepts results in the relation being added to the loaded concept map. Otherwise, if no concepts from the analyzed text are contained in the loaded map, no concepts are immediately highlighted. The user could then switch to a different display mode, where new, unrelated concepts appear in the list on the popup window.

The buttons with the labels “known”, “interesting” and “unknown” let the user switch between different display modes. These control what kind of information is emphasized on the website and what concepts are shown in the list on the popup window. The different display modes are described in detail in the following Section 4.2.

The final button on the popup window, labeled “show graph”, redirects the user to a new tab in the browser. On this new tab, the concept map that has been loaded into the application is visualized as a node-link diagram. All the concepts in the map are represented by a purple ellipse with the concept’s label written inside. Relations are labeled and represented as directional lines between two concepts.

4.2 Interaction

Depending on the state of a concept map, whether it is rather new and comparatively empty or nearly complete and containing a large number of concepts, the requirements for this highlighting tool might be quite different. Whether the user wants to extend existing branches in their concept map or discover new ideas that relate to the overarching topic of the map, the information and suggestions they want from the text can vary. Because of that, we have contrived multiple display modes that all cater to different needs one might have for this application. They range from only showing what is already known, over emphasizing interesting and related bits, to presenting entirely new ideas contained within a text.



(a) (b)

Figure 4.3: (a) Highlights in the "known" display mode and (b) corresponding concept map.

4.2.1 Known

By clicking the button labeled "known" on the popup window, after loading a concept map and analyzing the text, the user gets presented with all the concepts that appear in both the loaded map and the text. These concepts are highlighted with a green background color. Should both the source and the target concept of a relation in the loaded map occur within the same paragraph of the analyzed text, the two highlights are visually connected with a pink line. Hovering the mouse over a highlighted concept that is the target of a relation reveals a tooltip box that emphasizes the relation by displaying the whole proposition (source concept - linking phrase - target concept) (Fig. 4.3).

This display mode is supposed to serve as an overview of what is already known. It shows only what is already contained in the loaded concept map. This can be helpful when establishing whether a text is related enough to the ideas that are addressed within the concept map to examine it in more detail. For this reason, this is the default display mode that is applied when a text is analyzed. As no new information is highlighted, and therefore nothing new can be added to the map, none of the highlights provide any click-interaction.

4.2.2 Interesting

The button labeled "interesting" activates the key feature intended to extend an existing base of knowledge captured in a concept map. With this selection, the concepts that match between the loaded map and the analyzed text get filtered so that only those concepts remain that provide further relations to other concepts. All concepts that fulfill this requirement are then highlighted with a green background color. Upon hovering a highlighted concept, all its outgoing relations in the same paragraph are revealed by highlighting them with a yellow background color (Fig. 4.4). All of these target concepts are visually connected to the source concept by a pink link. Hovering the yellow highlight of a target concept reveals a hidden tooltip that alerts the user of a possible interaction. It tells the user that clicking on this highlight results in permanently adding the targeted

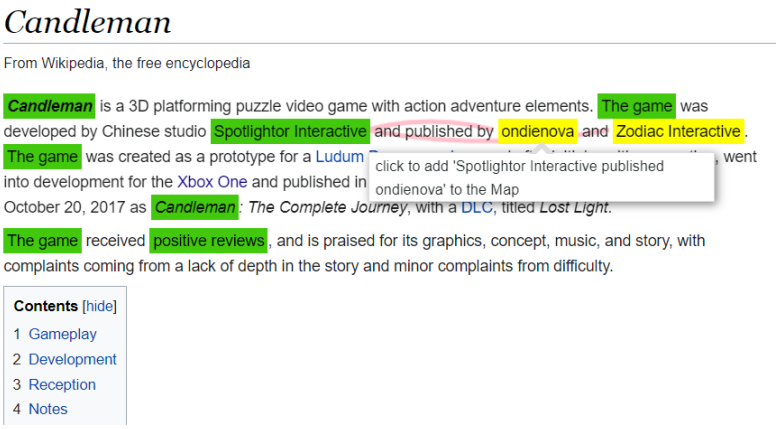


Figure 4.4: Example of the "interesting" display mode. Known concepts are highlighted in green and suggested relations for "Spotlightor Interactive" are shown in yellow.

relation to the concept map. This includes both the concepts themselves as well as the relation between them. When another green concept is hovered, the targets and relations of the previously hovered concept disappear and the new concept's relations are displayed.

This display mode is supposed to encourage extending the existing knowledge captured in the concept map. It shows where new ideas relate to known concepts and enables a quick and easy expansion of the map.

4.2.3 Unknown

The button labeled "unknown" will show the user new concepts that are not yet contained in their loaded concept map. Clicking this button will identify the ten concepts from the analyzed text that are not in the concept map and occur the most within the entire text. These ten concepts get highlighted with a yellow background color. Hovering the mouse over a highlighted concept once again shows a tooltip that indicates the click-interaction, which will add the targeted concept to the concept map.

This display mode is supposed to serve as an overview of the most important unknown concepts in the text (as these appear the most). They might or might not be useful for building the concept map since they only represent the context of the website but are not necessarily related to the topic of the concept map. It is still possible that some of these ten concepts are related to something in the concept map, or they might be entirely unrelated. These highlights only signify that these concepts themselves are not yet contained in the concept map. This selection can be useful if many other concepts on the website are already contained in the concept map, and therefore the context of the website has something in common with the topic of the map.

But showing the ten most occurring, unknown concepts is not the only way this display mode enables the user to explore unknown concepts. While the "unknown" display mode

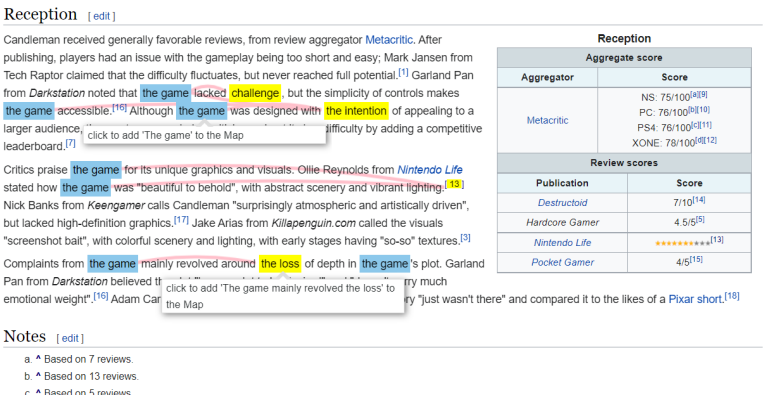


Figure 4.5: A concept (The game) is selected from the list (blue highlight). All its relations are shown in yellow. Tooltips that indicate interaction are revealed by hovering elements.

is selected, the list of concepts on the popup window changes from only showing concepts that are related to the loaded map to containing all concepts that were extracted from the web page. All the concepts in this list are sorted by occurrence so that the concepts that appear the most in the text show up at the top of the list. Clicking on an entry of this list results in the respective concept being highlighted in the text with a blue background color (Fig. 4.5). At the same time, should this concept have any outgoing relations, the targets of those relations are also highlighted with a yellow background color. All target concepts (yellow) are visually linked to the source concept (blue) by a pink line. The user can then either click the blue source concept to add only this one to the map, or they can click on any yellow target concept to include the entire relation in their concept map. Both these interactions are once again indicated by a tooltip box, revealed by hovering the mouse cursor over the respective highlight (Fig. 4.5).

Choosing concepts from the list can be a useful way to create new sub-branches for a concept map, taking concepts from a text on the same topic as the map, that are not yet directly connected to the previous contents of the concept map.

4.2.4 The Graph

Clicking the “show graph” button on the popup window opens a new tab in the browser. On this new tab, the currently loaded concept map automatically gets visualized as a node-link diagram (Fig. 4.6). On this page, concepts can be re-arranged by dragging and dropping them around. This might be necessary to bring the graph in a more appealing form, as the automatic layout might contain a few overlaps and edge crossings. In the top-left corner of the screen, an interaction area is located. There, manual changes to the concept map can be made. This area shows the name of the displayed concept map and multiple buttons. These buttons allow to create new concepts and relations that were not found through the extraction from a website. Clicking either the button for

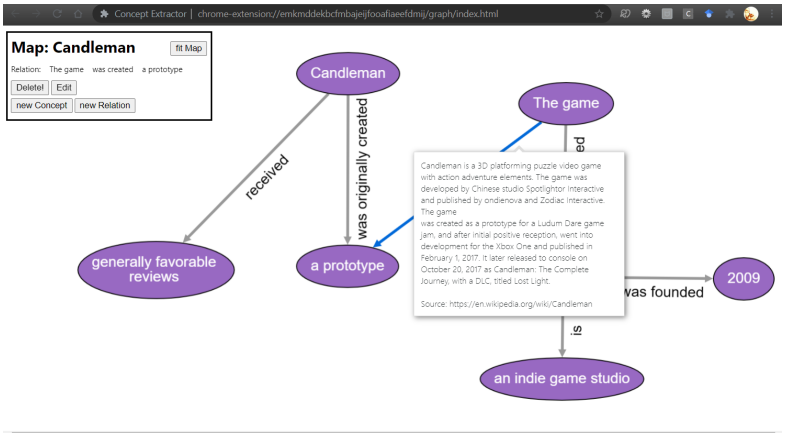


Figure 4.6: A concept map graph created with this application. A tooltip shows where a hovered edge originates from.

"new Concept" or "new Relation" reveals an additional area where the concept's name or the parts of a relation can be entered into text fields. Submitting these new additions automatically adds them to the graph and to the displayed concept map. Clicking a concept bubble makes that concept's name appear on the interaction area and two new buttons are revealed. Using those buttons, the user is able to edit the clicked concept's name or delete the concept from the display and permanently remove it from the concept map.

Relations can be edited or removed in the same way by selecting the linking line between two concepts. When a relation is removed, source and target concepts remain in the concept map, regardless of whether this creates disconnected components. Removing a concept, on the other hand, also removes all relations that originate from or lead to that concept. In order to provide more context, each relation link is equipped with a tooltip, revealed by hovering the relation (Fig. 4.6), which shows a paragraph of the original source text from which the proposition has been added. A link to the original website is also included. Relations that have been manually added state this in their tooltip and their source is given as unknown. To make these tooltips appear, the user only needs to hover the mouse cursor over the respective relation link.

The last available button on the interaction area is labeled "fit Map". This button can be clicked to re-fit the concept map graph on the available display space. It can be used in case the user zoomed or scrolled the map out of focus and all elements should be re-positioned in the users' field of view.

CHAPTER 5

Implementation

This chapter describes the technical details of the implementation, the frameworks and libraries that are used. Section 5.1 gives a rudimentary explanation of the provided backend server. In section 5.2, the details on the developed extension to the Chrome browser are described. This includes the extension itself, as well as the highlighting and the storage used to persist concept maps. Finally, in Section 5.3, the technology behind the rendered concept map graph is presented.

5.1 Server

The server side of this application was developed by Christoph Presch for his Diploma Thesis "Semi-Automatic Creation of Concept Maps" [Pre20]. This program was written in Python, specifically Python 3.7. It can receive the input text for the text analysis via a single Http-POST method that is implemented in a REST-API interface. The natural language processing pipeline uses a framework called spaCy [SPA], which uses neural networks to perform the different tasks of Natural Language Processing. As spaCy does not have built-in coreference resolution, another module had to be included for that purpose. To perform this task, Presch chose NeuralCoref [NEU], a pipeline extension for spaCy [SPA] that can annotate and resolve coreference clusters with the use of neural networks. For further detail on the implementation of the server, we refer to Presch's original thesis [Pre20].

Although the automatic text processing has been left untouched, we have introduced a small pre-processing step to the server. Since the application is now supposed to process the text of web pages, we need a processing step that can convert the input from HMTL chunks to plain text for the NLP pipeline to process. After receiving a String of HTML elements as an input on the Http-POST method, we use Beautiful Soup [BEA], a Python library that traverses HTML document trees and extracts all kinds of information from them. For our server, we use the stripped_strings method, which

Figure 5.1: The Chrome extension manifest.

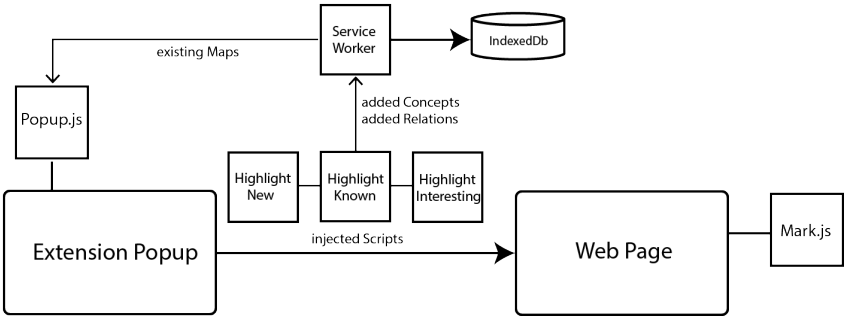


Figure 5.2: Involved scripts and their interaction.

removes any HTML-tags and other accessories from the input and returns only the text content of the given HTML. This stripped input string is then fed to the automatic text processing pipeline, where no further adjustments have been made.

5.2 Chrome Extension

The user interface part of this application is written as an extension for the Google Chrome browser. Extensions are small software programs that allow a user to personalize their browsing experience by adding extra functionality to their browser. We have implemented it using Manifest V3 [MV3], a new generation of Chrome extensions with enhancements in security, privacy, and performance. Every Chrome extension must contain a manifest file (Fig. 5.1). This is the backbone of every browser extension, a sort of settings file, so to speak. It contains information on the extension, like a name, a version, access permissions, and so forth. It is also specified in this manifest which scripts

are included with this extension. This includes declaring a "service worker", which is the core script of the application that gets executed when the extension is launched. Most of our application is written in JavaScript. Different scripts and components communicate via the Chrome runtime API. This API provides methods for sending and receiving messages that can carry any JSON-serializable payload. We use this to pass information between the service worker, which functions as a background page for the application, and the scripts controlling the popup window and the highlighting. Like this, the popup can request information from the database (Section 5.2.2), which is controlled by the service worker, and new entities can be sent to be permanently stored (Fig. 5.2).

In order to manipulate any website the application is applied on, we use the chrome.scripting API. This API allows the extension to inject JavaScript and CSS code into a website at runtime. Unlike the similar content scripts, which are loaded into the target website on the launch of the site, this method hands over more control to the extension's developer, as the extension can make decisions on injected scripts as it is being executed.

5.2.1 Highlights

When a user selects a highlighting method by clicking one of the display mode buttons on the popup, a new JavaScript file gets injected into and executed on the target website (Fig. 5.2). These different highlight scripts receive the keywords (i.e. names of concepts) that need to be highlighted and use them to identify the parts of the text on the website that they should target. The keywords arrive as Strings but are then converted into regular expressions before highlighting them in order to escape potentially harmful characters. Depending on the display mode, not only the concepts received through keywords are highlighted, but their related concepts are also calculated and marked on demand. For annotating all the concepts with their different background colors, a JavaScript package called mark.js [MJS] is used. The mark.js script is included in the manifest file (Fig. 5.1) as a content script. This means it gets automatically included into specified websites whenever they are accessed. By declaring "$matches :< all - urls >$" in the manifest file, it will be included on any website.vis Mark.js always operates on a given context, which can be any single HTML-element (or collection thereof). Within this context, it matches any search term or custom regular expression in the text and wraps the matching parts in a custom tag. By default, the <mark> tag is used. Like any other elements, these mark elements can then be queried by the highlight script.

When marking a keyword or regular expression, a lot of options can be specified. In this case the acrossElements option is used, which allows the program to mark search terms even if they are split by other HTML-tags, like an <a> tag when part of the search term is contained in a link. Within these options it is also possible to give each marked element a class label and to exclude certain parts of the page from being marked. This concerns, for example, the custom tooltip boxes that are added throughout the highlighting process. These typically contain concept names, but the phrases should not be marked in those places.

Once a concept and a related target are marked, the visual connecting lines are calculated by by comparing the bounding boxes of the marked elements. These bounding boxes are received with the help of the Element.getBouningClientRect function, which returns the smallest rectangle that contains the entire marked element. The edges of these bounding boxes serve as anchor points for the lines connecting marked elements. For rendering the lines the "CanvasRenderingContext2D" interface of the Canvas API is used. This interface provides a 2D rendering context for a drawing surface on an HTML <canvas> element. There, one can draw shapes, in this case a path, by programmatically specifying points on the surface and connecting them with a stroke. The paths always stretch all the way from one end of the canvas to the other, as the size of the canvas always matches the distance between the concepts' bounding boxes. Quadratic Bézier curves are drawn to produce a clean arc between source and target concepts. Control points are always positioned at the beginning and the end of the path with one extra control point at the half-way point and offset by a fixed amount. The canvases with the rendered lines are then positioned in between the calculated bounding boxes with a negative z-index. This results in them being drawn behind the other elements on the page, like other text, so as not to occlude any important information.

5.2.2 Storage

When new concepts and relations are created/saved, a representation of them needs to be placed in a permanent storage. A concept map is represented as a JSON object that has three attributes: 1) A name that preferably describes the topic or the focus question of the map, 2) a list of nodes, and 3) a list of edges. Nodes (i.e. concepts) have only two attributes to themselves: 1) A unique id that identifies them and 2) a label that serves as the display name of the concept. Edges (i.e. relations), on the other hand, consist of five attributes. They also have a label that contains the text that is displayed for this relation. They have a source and a target attribute, which contain the unique ids of their respective source and target concepts. Finally, they have two attributes that give information on the relation's origin. The sourceText attribute contains a paragraph of the text from which the relation was extracted and the url attribute gives the website where the relation was found.

These representations of concept maps are stored in the client's web storage. This application uses the IndexedDB API [IDB] to store, read, and update its concept maps. This API allows for storage of significant amounts of structured data. It works similar to an SQL-based database system with the key difference that IndexedDB uses JavaScript objects instead of fixed column tables. Any actions on the database are performed through transactions. This ensures thread safety and that all operations of a transaction succeed. Upon installation of the extension, a single database is created in which all future concept maps will be stored. Within this database, the name attribute of the concept maps serves as the unique key, which means that one user may not have multiple concept maps of the same name.

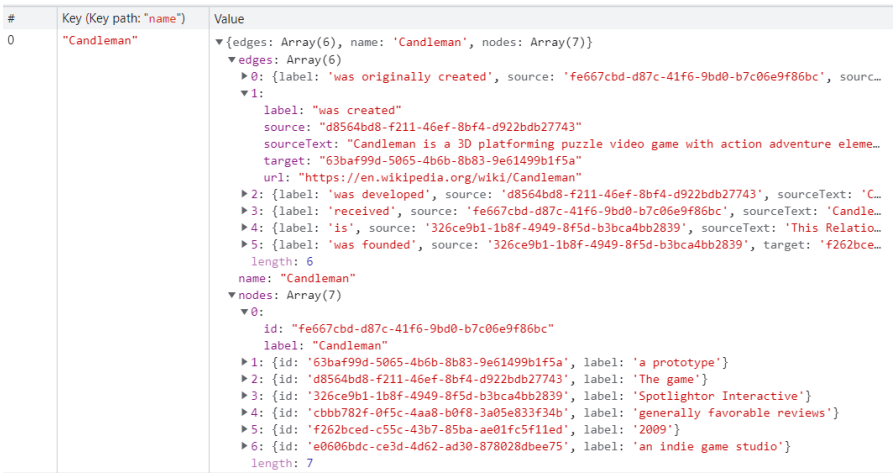


Figure 5.3: A concept map in the database.

In addition to the IndexedDB, Chrome's storage API [STO], another object based storage, is also employed to save any information that needs to be passed between different scripts. This API is specifically optimized for the storage needs of extensions. It provides local storage and synced storage properties. Synced meaning that objects put in this storage are available to the user across devices. In this application, however, only the local storage property is used. Any content script (including scripts injected at runtime) have immediate access to this storage which is of particular use because no function arguments can be passed to a script that is injected at runtime. With this method it is possible to pass, for example, the keywords that need to be highlighted from the extension's main script to the different highlighting scripts. This is achieved by first saving these keywords to the local storage and receiving them again during the execution of the highlighting script. The entries that are saved to this local storage are all of temporary use for this application. Nonetheless they are permanently saved in the browser's storage. In case a user wants to clear this storage space, there is the possibility to clear all of it on the extension's options page. This page is available via "options" in the right-click context menu on the extension icon.

5.3 Graph

The graph page is also written in pure JavaScript. In order to display a concept map as a node-link diagram, the open-source graph theory library Cytoscape.js [FLH+16] is used. This library makes creating and interacting with a graph quite easy. Nodes and edges (concepts and relations) can be integrated into the cytoscape instance in the same format as they are stored in the concept maps from storage. The program then automatically positions them according to a specified layout algorithm. In Cytoscape.js, layouts are considered extensions. These algorithms need to be imported into the application and called upon. Some basic layouts, however, are already integrated in the default cystoscape

package. There are geometric layouts like grids and circle layouts, hierarchical layouts, force-directed layouts, and more. For this application the “cola” layout algorithm [CYC] is applied. This is a force-directed layout that allows for some extra custom constraints to be placed on the physics simulation that places nodes and edges, like node spacing, avoiding overlap, or a specific edge length.

All targets (nodes, edges, background, ..) within the cytoscape instance are equipped with event handlers, which makes it possible to put different interactions on all of them. This is used to provide different interaction possibilities when e.g. nodes or edges get clicked or hovered over. When nodes or edges are manually added or deleted they are automatically placed in/removed from the graph. New nodes are initially positioned at a fixed location in the center of the screen, edges automatically connect their source and target nodes. At the same time, a message with the appropriate request is sent to the extension service worker to execute an update transaction on the database and permanently add/delete the respective item.

CHAPTER 6

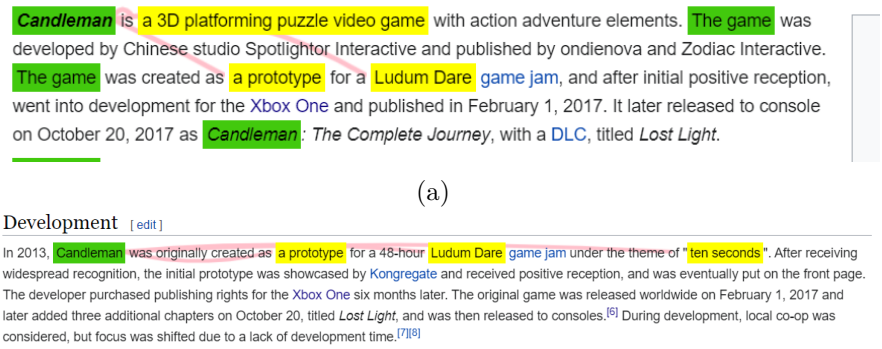
Limitations

Even though the application works as intended in most cases, there are a number of factors that limit the use in a real-case scenario. In this chapter we try to explain a few different situations where some parts of the application cannot work properly or the results might not look exactly as expected. Examples that showcase how the application is functioning correctly are shown later in Chapter 7. Problems that arise include the following:

Only certain content can be analyzed

The application is intended to be applicable on arbitrary web pages. This entails the problem that we have no clue about the structure of these web pages. We need to determine somehow what parts of the pages are relevant for analysis and which are not. The backend server expects a collection of HTML elements that contain the text it is supposed to process. One possibility would have been to simply select all the contents of the <body> tag, but that would in most cases include distracting page elements like menus, side bars, tables, and site notice at the bottom of the page. All these are elements that typically contain text that is not a coherent part of the text content of the page and they would interfere with meaningful concept extraction. This meant we needed a more restrictive selection of elements to submit for analysis. In the end we decided to consider only the contents of paragraph elements <p>. In most cases, these are the elements that contain the bulk of the relevant text. Similar to the text analysis, only these paragraph elements are considered a valid context where highlights can occur. We decided to do this, because only these elements are analyzed in the first place and we do not want any misplaced highlights in some menu or side bar. Unfortunately, this can be too restrictive in some cases. There are web pages that omit the use of <p> elements and put text e.g. directly into container elements ( ). On such pages the application can not be properly used as neither the text can be analyzed, nor concepts highlighted.

Highlights can appear in places where they make little sense



(a)

(b)

Figure 6.1: (a) and (b) both show paragraphs from the same text. But the relation between “Candleman” and “a prototype” should only exist in (b).

Due to the fact that the web page’s text can be fragmented and is pre-processed before entering the automatic text processing software, the exact places of occurrence for concepts and relations are not able to be determined. Presch’s software [Pre20] does provide exact occurrence indices for concepts and relations (i.e. the index of the first and last character of the word or phrase in the text), but because the arrangement and length of the analysed text might not always match the text on the web page exactly these occurrences cannot be used to determine where a concept or relation comes up on the web page. Because of this, all occurrences of a concept are highlighted on the page regardless of whether it is part of a relation in that exact spot. This means that sometimes concepts are highlighted even though they most likely do not serve any direct purpose in the context of the user’s task. In addition, there is the possibility that two concepts that form a relation in some part of the text also appear close to each other in another part but without the explicit relation between them. In this case, the relation would be emphasized in both parts although it does not exist in one of them. An example of this behaviour can be seen in Figure 6.1. The relation “‘was originally created” between “Candleman” and “a prototype” is highlighted in both 6.1a and 6.1b. This relation, however, only really exists in the context of Fig. 6.1b. In the paragraph on Fig. 6.1a “Candleman” and “a prototype” do not actually have a direct relation in the meaning of the sentence. A user can still add this relation to their concept map from both parts of the text. But if they choose to add it from the part where it does not occur, the source text passage shown later in the concept map graph will show a text passage where the relation might not come up at all.

The length of text influences the performance

Like Presch has already discussed in his thesis [Pre20], the performance of the program degrades continuously with increasing length of input text. As no part of the NLP pipeline has been altered, the same issues still exist with this version of the software. The

automatic analysis step takes increasingly more time the more text is presented on the target web page. Presch describes that the system provides good performance for texts with up to 750 words. Depending on the target web page, this performance limit can be exceeded by potentially thousands of words which would increase the computation time from seconds to possibly minutes. (Details follow in Section 7.3 Performance). With more text the quality of the highlights also potentially reduces. The longer the text, the more likely it can be that some concepts might co-appear in multiple paragraphs, and relations might get marked in places where they do not actually exist (as described in the previous paragraph).

Regular Expressions and illegal characters

While highlighting concepts on the web page, the concept's label is converted into a regular expression for the mark.js framework [MJS] to match it with the text on the page. In some occasions the concepts returned by the automatic text processing software can contain characters that cause this regular expression to result in an error. This is mainly the case when the label contains an opening parenthesis but the closing parenthesis is not included in the concept's label. This appears to be one flaw of the automatic text processing software, as this case occurs more often than not when a text contains phrases in brackets. This error leads to a halt in the execution of all further functions and therefore no more concepts beyond this point in the text get highlighted. The application will resume as normal once a new highlight script gets executed by selecting a different concept or switching display modes.

Highlights are strongly tied to the server output

The highlighting and suggesting features of this extension are strongly restricted by what the automatic text processing software puts out. This means that only those concepts that the NLP algorithms detect can be emphasized in the "interesting" display mode. Concepts that were manually added to a concept map by the user will never receive any highlights under this mode, much less be connected to any new suggestions. This is a result of the program comparing the concept map loaded by the user with the concepts and relations extracted by the backend, and only regarding those concepts further that appear in both of those maps. This is one way the program's usefulness could be limited if e.g. a user wanted to revolve their concept map around an acronym or other uncommon word or phrase that does not get classified as a concept entity by the NLP pipeline. This also means that the application can cease to provide interesting suggestions for concept maps where many concepts have been added by hand. Manually added concepts and relations are, however, highlighted in the "known" display mode, as this mode only takes the concept map loaded by the user into account when marking parts of the text.

Multiple links are drawn for one relation

There are also situations where the connecting lines between a concept and its suggested relations appear to produce a visual bug. On some occasions, multiple links are drawn between two concepts where only one relation should exist. Although this looks like a bug, it is the program doing everything it is supposed to do. Multiple lines can happen

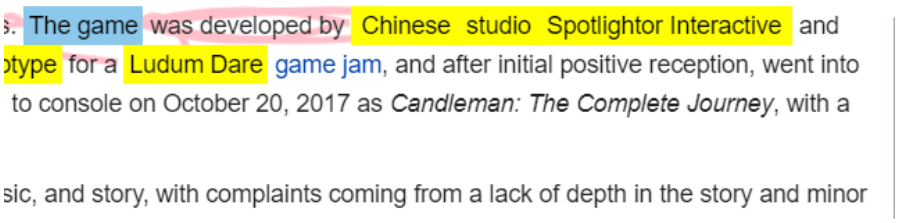


Figure 6.2: Multiple links are drawn for what seems to be one target.

when e.g. a phrase consisting of multiple words is recognized by the automatic text processing as a concept, but one or more of the phrases' words are also concepts on their own. In this case, the program creates each its own mark element, possibly stacking multiple marks. Then, lines are drawn from the source concept to each of the target mark elements resulting in multiple lines reaching to what appears to be only one target. This can be seen in Figure 6.2, where "Chinese", "Spotlightor Interactive" and "Chinese studio Spotlightor Interactive" are all target concepts. A similar result appears when parts of a target phrase are contained in different HTML elements. E.g. when part of the phrase is a hyperlink and therefore enclosed in an <a> tag. Like before, multiple mark elements are created, one for the part of the phrase that is inside the link tag, and one for the part that is not.

CHAPTER 7

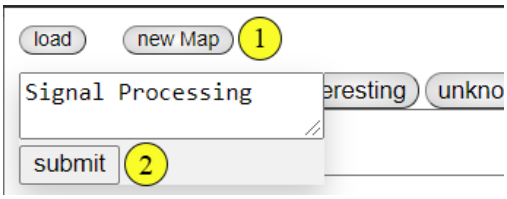
Discussion & Results

In this chapter, we describe an example walkthrough which showcases how a typical use-case could look like for this application. We have also conducted a usability inspection by presenting that walkthrough to three test users and receiving their feedback. Finally, we discuss that feedback.

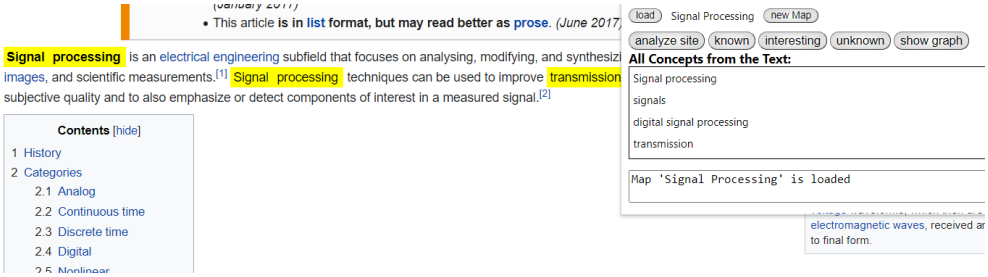
7.1 An Example Walkthrough

As with any new task, we start at the beginning by creating a new concept map. For this, the "new map" button on the opened extension popup can be clicked and a name for the new concept map is entered into the text field. For this example, we want to create a concept map that deals with Signal Processing. Submitting the name automatically loads the empty map as currently active (Fig. 7.1a). To start building the Concept Map, we do a Google search for "Signal Processing", the key phrase of the map, and choose the Wikipedia article on Signal Processing [WT1] as a starting point. Opening the extension popup and clicking the "analyze site" button yields, for the moment, nothing at all. As it is still an empty map, no concepts on the site are yet related to it. Therefore, we click the "unknown" button on the extension popup to reveal all concepts extracted from the text in a list on the popup, as well as highlights in the text for the ten concepts that appear the most throughout the text (Fig. 7.1b). We can immediately see that our key phrase, Signal Processing, is an identified concept. A click on the yellow highlight prompts an alert that "Signal processing has been added to the map". Re-opening the extension popup now marks the phrase Signal Processing in green, signifying that it is contained in the currently loaded concept map. We can now hover the mouse cursor over these green highlights to see what other concepts directly relate to the phrase "Signal Processing". For example, we find that "Signal Processing" is "an electrical engineering subfield", or that it can be found in "the classical numerical analysis techniques". The first one we add to the concept map by clicking the yellow highlight on the target concept. At this time,

no more interesting relations appear automatically, but we see that “Signal Processing” still appears in more places on the website. The text lists different categories of signal processing, which could be useful. As these categories are not directly linked to the phrase “Signal Processing” by the automatic text processor, we go back to highlighting unknown concepts and find “digital signal processing”, and a little further down the list “analog signal processing”. Both these concepts can be added to the map by selecting them from the list and then clicking the blue highlight that marks them in the text.



(a) creating a new map



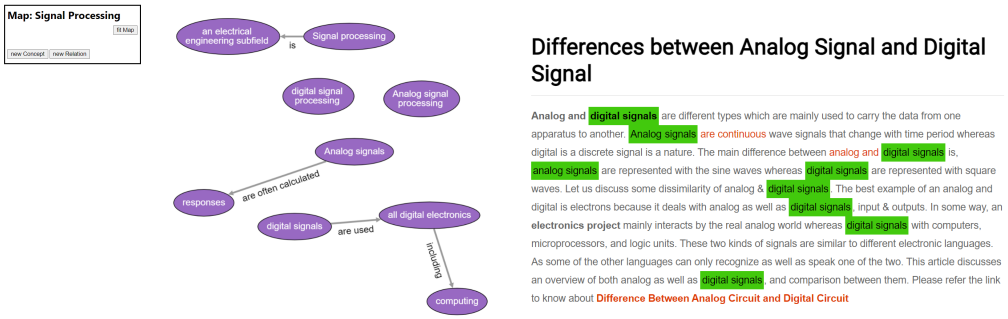
(b) First concepts with the “unknown” display mode.

Figure 7.1

At this point, we are done with Wikipedia. Since we want to extend the map at the two new sub-categories of signal processing we added, we now do another Google search, this time for “analog signal processing”. A few results in, a headline reads “Analog vs. Digital Signals: Uses, Advantages and ...”. Visiting this website [WT2] and analyzing the text shows that, although the phrases analog-, and digital signal processing are present on the site, they do not have any outgoing relations in this context. But looking at the extension popup we see recommendations for “Analog signals” and “digital signals”. We select “Analog signals” to find out that e.g. “Analog signals are often calculated responses to changes in light, sound, temperature, ...”. The relation extracted by the text processing software only links “Analog signals” to “responses”, but we add it regardless and come back to it later. Looking at “digital signals” gives us “Digital signals are used in all digital electronics”, and following that “all digital electronics, including computing”. The second part of the phrase “computing equipment” is omitted by the program, and the second concept, “data transmission devices”, related to “all digital electronics” is not recognised at all.

At this point we have a look at the graph view of this concept map. There we can manually add some of these missing concepts an edit the relations we got so far. We get

there by clicking the “show graph” button on the extension popup. On the graph page we see all the concepts and relations we have added this far (Fig. 7.2a). To clean things up, we first link the disconnected components of the graph by adding new relations. We specify that “Signal processing” includes “Analog signal processing” and “digital signal processing”. “Analog signal processing” uses “Analog signals“ and “digital signal processing” uses “digital signals”. We refresh the page to let the layout algorithm consider all the new relations. Now we can rename some concepts and relations slightly to make them more understandable. For example, we change “are used” to “are used in”. We can also add the concepts we found in the text earlier that were not recognized by the program. We change the concept “computing” to “computing equipment” and add the extra relation “all digital electronics” including “data transmission devices”. On the other end of the graph we remove the word “calcualted” from the relation and add it to the target concept “responses”. We also extend this thought by adding the examples from the text that were not automatically related to “responses”. We add the new relations “calculated responses to changes” and then “changes in light”, “changes in sound”, and “changes in temperature”.



(a) first set of disconnected components (b) a lot of green highlights already show up

Figure 7.2

After this cleaning, we return one more time to the Google search engine. This time, looking for digital signals eventually brings us to another website [WT3]. Analyzing this site immediately yields several green highlights (Fig. 7.2b). We can quickly extract that “Analog signals are continuous wave signals” and further down the page “digital signals include a limited variety“ (of values which lie among 0-to-1). We add both these relations and return once more to the graph view. To finalize this example concept map, we specify that it is “a limited variety of values” and add to it the relation “between 0 and 1”. the final product of this walkthrough can be seen in Figure 7.3.

7.2 Usability Inspection

In order to evaluate the usability and detect possible shortcomings of the whole application, the above example has been presented to three independent experts in the computer

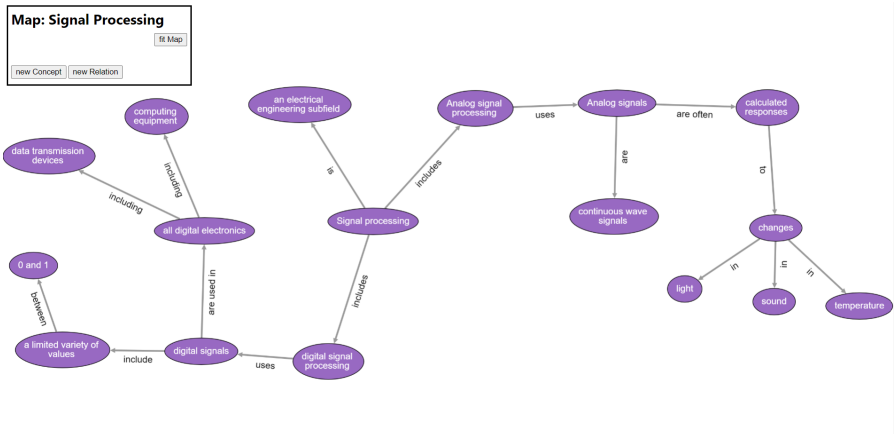


Figure 7.3: The final product of the Walkthrough

science field. They were led through the whole process of creating and extending a concept map on the topic of signal processing from scratch. On the way, they were repeatedly asked to evaluate the usability and clarity of the presented elements of the application.

The findings from these inspections were mostly unanimous. Three main concerns could be identified as follows:

1. Not enough (usable) suggestions,
2. interaction with certain elements is not always clear, and
3. manual interaction is tedious.

But we did not only receive this negative feedback. All test users did come to the conclusion that:

4. The application can be useful for research tasks.

Not enough (usable) suggestions

In the current state of the application, the creation of a satisfying concept map solely off the given suggestions that can be highlighted in the text seems rarely possible. The test users all recognized the fact that concept suggestions many times do not cover the entirety of a phrase that clearly belongs as a unit to the human reader. They also noted that oftentimes some relation between phrases was obvious to them but the application would not suggest it. Due to this lack of suitable suggestions, one user doubted that the application as it is would reduce the effort compared to a fully manual concept map creation. She based this observation on the fact that a lot of concepts and relations had to be manually added and/or edited in this example in order to capture the intended ideas from the source text.

Interaction can be unclear

Two of our test users explicitly noted that not all functions of the various interactive elements on the extension popup would be immediately clear to them without the explanation included in this example walkthrough. One user explained that a button labeled “load” to her does not mean getting another map from the storage. She was wondering what would load if that button was clicked. Similarly, a second user exclaimed that she would rather have used some kind of drop-down list instead of a button for this purpose. The same user also suggested to distinctively separate the buttons for analyzing the site text and switching to the graph view from the buttons for the three display modes. The reason for this being that they serve a completely different purpose from switching between different displays. One more feature that was suggested by one user was some sort of information bubble icon where optional instructions and explanations to all the interactable elements could be found.

Manual interaction is tedious

While all users saw the need for manual adaptations to the concept map, they also pointed out several features that they would like improved/added before the application became usable in a satisfying way to them. The most critique was given to the manual creation of new relations. Although the process of adding a relation seemed self-explanatory to them, all three addressed that having to type out the names of the source and target concepts was inconvenient. Two users explained they would rather have the possibility to select existing concepts from somewhere or add relations between existing concepts through a point and click interaction. One user also asked whether it was possible to quickly move relations from one concept to another. An additional feature that was explicitly asked for by one user was to be able to give different colors to concept bubbles. She explained that, when using tools like mindmaps, she needed to visually categorize concepts. The same user also suggested that the interaction area in the top-left corner of the graph view could be minimized and only extended to its full size on demand in order to maximize the available space for the graph.

Useful for research tasks

All these problems aside, all three test users could definitely see the potential of this application. In an ideal world, where all the program’s suggestions were accurately representing the meaning of the text and no concepts got omitted, they would all have found use-cases to apply this concept map extractor. Even one user, who described herself as “not particularly a mindmap person”, said that she could definitely see the advantages that the automatic highlighting and easy addition of concepts and relations would bring. Another user gave the example of a student having to prepare a presentation. He believed that he would have liked such a tool in that scenario. Two users immediately pointed out the overview aspect of the application as a positive effect. They named that as one of the strengths, with one of them explaining that when researching a topic, it can be very useful to see at one glance whether or not a new text deals with similar concepts to what has been read before.

Overall, it can be said that all test users liked the idea and that it has the potential to be used to assist in research and summarizing efforts. However, they also pointed out many of the existing flaws that prevent the application form being used in a real use-case in its current state. All of them had useful insights suggestions for possible improvements.

7.3 Performance

We have also conducted a basic evaluation of the application's performance in regards to processing time when analyzing texts of different lengths. As stated in the limitations chapter 6, text on the internet can come in various lengths and the usefulness of the application will inevitably decrease with increasing text length. For this evaluation we have chosen to compare the time needed to process the page text of four different Wikipedia articles:

1. Candleman [Wika], length approx. 600 words,
2. The Shining (Novel) [Wikb], length approx. 2450 words,
3. Daniel Radcliffe [Wikc], length approx. 3950 words, and
4. Elephant [Wikd], length approx. 10800 words.

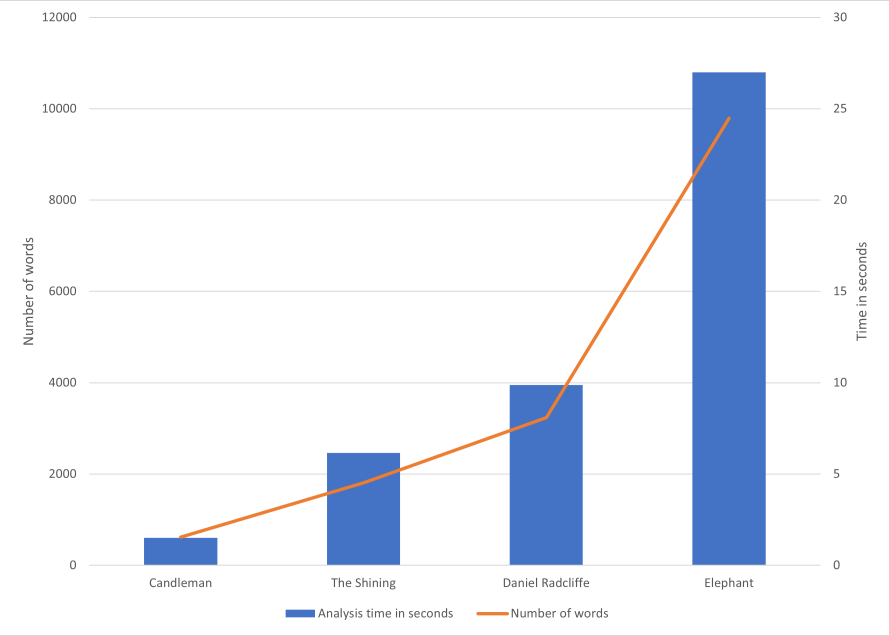


Figure 7.4: Performance benchmarks for different online text.

This comparison yielded that the time to process an online text grows relative to the document's number of words. Analyzing Candleman took the application 1.54 seconds,

for The Shining it took 4.52 seconds, Daniel Radcliffe took 8.08 seconds, and Elephant was done processing after 24.48 seconds. These results are visualized in Figure 7.4.

All the tests were run on a Laptop with an Intel Core i7-7500U CPU and 12GB of RAM.

7.4 Discussion of Results

The feedback received through the usability inspection mostly reaffirms what we have also observed throughout the development and testing process. The user interface works as intended in most cases. Exceptions to this are outlined in the limitations Chapter 6. It is far from perfect, but manages to portray the suggestions received from the concept map mining software to the user and allows them to create a custom concept map from these suggestions.

In the end, it could be shown that it is possible to apply the results of an automatic concept map mining software on arbitrary web pages using a browser extension written in pure JavaScript. It has, however, also shown that relying solely on the suggestions given by this software does not result in a satisfying research process nor a suitable concept map. We have found that there is still a need for manual intervention where concepts and relations can be created and edited without being suggested by the software. Without this, most concept maps created with this tool would fall short in terms of readability and usability overall. This shows that the best way to improve the whole application would be to try and improve the natural language processing aspect of the software. The usability would probably increase drastically with more accurate concept and relation suggestions that capture the precise meaning of the text they come from.

The user interface of course also still has a lot of potential for improvement. As the user evaluations show, not all possible interaction is self-explanatory and some elements have initially confused participants.

CHAPTER 8

Conclusion

In this final chapter there is a brief summary of what has been done for this thesis. We also outline at what points future work could tie on and improve this application.

8.1 Summary

For this thesis, we created an extension to the Google Chrome browser that extracts the active website's text and forwards it to an automatic text processing software. This software applies natural language processing procedures to perform concept map mining. The result of this is a set of concepts and relations that form propositions in the form of subject-verb-object triples. The browser extension receives these suggestions and, by executing different script files, highlights a subset of the suggestions based on different display modes. Users can select highlighted suggestions and add them to a concept map structure. This concept map is visualized on-demand as a node-link diagram.

The development process, the creation of a detailed example, and the ensuing user evaluation revealed the strengths and weaknesses of the application. It showed that the goal of creating an interface that presents the results of the automatic concept map mining directly in the place of text's origin is achievable. But it also showed that more effort needs to be put into both the automatic text processor, as well as the user interface to produce a universally useful application.

8.2 Future Work

Many of the application's shortcomings could be tackled in future projects. This includes work on both the text processing server and the user interface. With the automatic text processing, the employed natural language processing steps need to better handle longer text. At the moment, the processing time the server needs to provide its suggestions

is one major limiting factor for the application. Depending on the topic, websites tend to contain quite a bit more text than the 700 words Presch pointed out in their thesis [Pre20] as a threshold for good computation time. The concept- and relation extraction also need improvements regarding the quality of output. As it is, too few propositions are actually meaningful for the user.

Regarding the user interface, many useful suggestions were made during the user evaluation on how to improve the interaction design. This includes improvements to the extension popup and also the graph view and manipulation. Both these areas could receive a welcome upgrade through a creative design process. An example sketch for an improved extension popup, with user suggestions from our evaluation, can be seen in Figure 8.1.

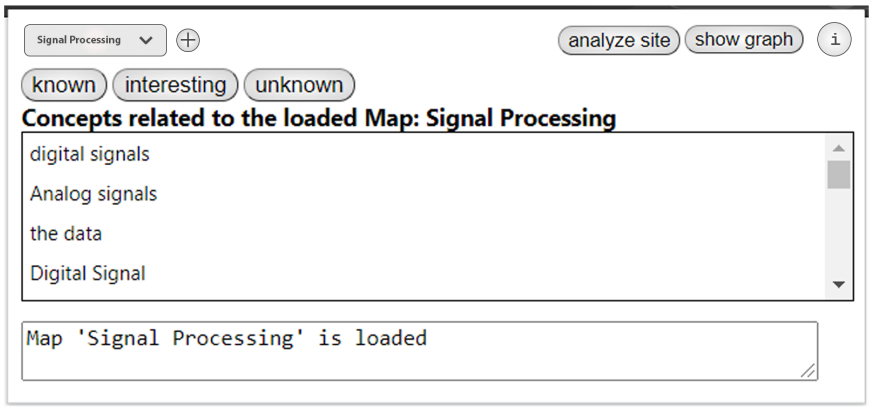


Figure 8.1: Example for an improved popup window.

List of Figures

Bibliography

[AN12] Christopher Andrews and Chris North. Analyst's workspace: An embodied sensemaking environment for large, high-resolution displays. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 123–131. IEEE, 2012.

[Aus63] David P Ausubel. The psychology of meaningful verbal learning. 1963.

[BEA] Beautiful Soup - Python library for quick screen-scraping. `https://github.com/huggingface/neuralcoref`. Accessed: 03.08.2021.

[CCL18] Alberto J Cañas, Roger Carff, and James Lott. ecmap: An embeddable web-based concept map editor. In *Proceedings of the Eighth International Conference on Concept Mapping, Medellín, Colombia*, 2018.

[CDC+01] Mary J Carnot, Bruce Dunn, Alberto J Cañas, P Graham, and Jason Muldoon. Concept maps vs. web pages for information searching and browsing. `https://www.researchgate.net/profile/Alberto_Canas/publication/266489427_Concept_Maps_vs_Web_Pages_for_Information_Searching_and_Browsing/links/543546300cf2bf1f1f28662f.pdf`, 2001. Retrieved from the World Wide Web: last accessed 22.10.2021.

[CF12] April Colosimo and Megan Fitzgibbons. Teaching, designing, and organizing: Concept mapping for librarians. *Partnership: The Canadian Journal of Library and Information Practice and Research*, 7(1), Jun. 2012.

[CHC01] MR Carvalho, Rattikorn Hewett, and Alberto J Cañas. Enhancing web searches from concept map-based knowledge models. In *Proceedings of SCI 2001: Fifth multiconference on systems, cybernetics and informatics*, pages 69–73, 2001.

[CHC+04] Alberto Cañas, Greg Hill, R. Carff, Niranjan Suri, James Lott, Thomas Eskridge, Gloria Gomez, Mario Arroyo, and Rodrigo Carvajal. Cmaptools: A knowledge modeling and sharing environment. *Concept Maps: Theory, Methodology, Technology Proceedings of the First International Conference on Concept Mapping*, 2004.

[CLN+13] David Campos, J Lourencço, Tiago Nunes, Rui Vitorino, Pedro Domingues, Sérgio Matos, and José Luís Oliveira. Egas–collaborative biomedical annotation as a service. In *Proceedings of the Fourth BioCreative Challenge Evaluation Workshop*, volume 1, pages 254–259, 2013.

[CRE] Creately, visual workspace for collaborative work. `https://www.creately.com/`. Accessed: 17.07.2021.

[CS13] Wei-Te Chen and Will Styler. Anafora: a web-based general purpose annotation tool. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2013, page 14. NIH Public Access, 2013.

[CYC] The Cola.js physics simulation layout for Cytoscape.js. `https://github.com/cytoscape/cytoscape.js-cola`. Accessed: 08.08.2021.

[DOM] Mindomo, collaborative mind maps, concept maps, outlines, and gantt charts. `https://www.mindomo.com/`. Accessed: 18.07.2021.

[FLH+16] Max Franz, Christian T Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D Bader. Cytoscape. js: a graph theory library for visualisation and analysis. *Bioinformatics*, 32(2):309–311, 2016.

[Got07] David Gotz. The scratchpad: sensemaking support for the web. In *Proceedings of the 16th international conference on World Wide Web*, pages 1329–1330, 2007.

[GSL+14] Thomas Geymayer, Markus Steinberger, Alexander Lex, Marc Streit, and Dieter Schmalstieg. Show me the invisible: visualizing hidden content. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3705–3714, 2014.

[HBW08] Raphael Hoffmann, Patrick Baudisch, and Daniel S Weld. Evaluating visual cues for window switching on large screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 929–938, 2008.

[HCFC01] Robert R Hoffman, John W Coffey, Kenneth M Ford, and Mary Jo Carnot. Storm-lk: A human-centered knowledge model for weather forecasting. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 1, pages 752–752, 2001.

[HE99] Christopher G Healey and James T Enns. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE transactions on visualization and computer graphics*, 5(2):145–167, 1999.

[HHL19] Hannes Hapke, Cole Howard, and Hobson Lane. *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*. Simon and Schuster, 2019.

[IDB] IndexedDB - a Low-Level API for Client-Side Storage. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API. Accessed: 08.08.2021.

[KCB10] Juliana H Kowata, Davidson Cury, and M Boeres. A review of semi-automatic approaches to build concept maps. In *Proc. 4th International Conference on Concept Mapping*, pages 40–48. Citeseer, 2010.

[LMR04] David B Leake, Ana Maguitman, and Thomas Reichherzer. Mining the web to suggest concepts during concept map construction. *Concept Maps: Theory, Methodology, Technology, Vol. 1*, page 135, 2004.

[LUC] Lucidchart, intelligent diagramming application. https://www.lucidchart.com/. Accessed: 17.07.2021.

[MIN] Mindmup, free online mind mapping. https://www.mindmup.com/. Accessed: 17.07.2021.

[MIR] Miro, the online collaborative whiteboard platform. https://www.miro.com/. Accessed: 18.07.2021.

[MJS] Mark.js - JavaScript keyword highlighter. https://markjs.io/. Accessed: 05.08.2021.

[MSB+14] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[MT14] Narges Mahyar and Melanie Tory. Supporting communication and coordination in collaborative sensemaking. *IEEE transactions on visualization and computer graphics*, 20(12):1633–1642, 2014.

[MV3] Manifest V3 - A step in the direction of security, privacy, and performance. https://developer.chrome.com/docs/extensions/mv3/intro/. Accessed: 04.08.2021.

[NC06] Joseph D Novak and Alberto J Cañas. The theory underlying concept maps and how to construct them. *Florida Institute for Human and Machine Cognition*, 1(1):1–31, 2006.

[NC07] Joseph D Novak and Alberto J Cañas. Theoretical origins of concept maps, how to construct them, and uses in education. *Reflecting education*, 3(1):29–42, 2007.

[NEU] NeuralCoref 4.0: Coreference Resolution in spaCy with Neural Networks. https://www.crummy.com/software/BeautifulSoup/. Accessed: 03.08.2021.

[NGB84] Joseph D Novak, D Bob Gowin, and Gowin D Bob. *Learning how to learn*. cambridge University press, 1984.

[NM91] Joseph D. Novak and Dismas Musonda. A twelve-year longitudinal study of science concept learning. *American Educational Research Journal*, 28(1):117–153, 1991.

[NXB+16] Phong H Nguyen, Kai Xu, Andy Bardill, Betul Salman, Kate Herd, and BL William Wong. Sensemap: Supporting browser-based online sensemaking through analytic provenance. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 91–100. IEEE, 2016.

[PAN] Panama Papers - The Power Players / Visualizations of Offshore Leaks database. `https://www.icij.org/investigations/panama-papers/the-power-players/`. Accessed: 09.07.2021.

[PR94] Stephen Palmer and Irvin Rock. Rethinking perceptual organization: The role of uniform connectedness. *Psychonomic bulletin & review*, 1(1):29–55, 1994.

[Pre20] Christoph Presch. Halbautomatisches erstellen von concept maps. 2020.

[SOK+15] Hendrik Strobelt, Daniela Oelke, Bum Chul Kwon, Tobias Schreck, and Hanspeter Pfister. Guidelines for effective usage of text highlighting techniques. *IEEE transactions on visualization and computer graphics*, 22(1):489–498, 2015.

[SPA] spaCy - Industrial-Strength Natural Language Processing. `https://spacy.io/`. Accessed: 03.08.2021.

[STO] Chrome Storage - an API to store, retrieve, and track changes to user data. `https://developer.chrome.com/docs/extensions/reference/storage/`. Accessed: 08.08.2021.

[SWS+11] Markus Steinberger, Manuela Waldner, Marc Streit, Alexander Lex, and Dieter Schmalstieg. Context-preserving visual links. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2249–2258, 2011.

[VC08] Jorge J Villalon and Rafael A Calvo. Concept map mining: A definition and a framework for its evaluation. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 357–360. IEEE, 2008.

[VPE] Visual paradigm online, editor for a variety of technical and business diagrams. `https://online.visual-paradigm.com/diagrams/`. Accessed: 17.07.2021.

[WGSS21] Manuela Waldner, Thomas Geymayer, Dieter Schmalstieg, and Michael Sedlmair. Linking unstructured evidence to structured observations. *Information Visualization*, 20(1):47–65, 2021.

[Wika] Candleman. From Wikipedia, the Free Encyclopedia. `https://en.wikipedia.org/wiki/Candleman`. Accessed: 21.10.2021.

[Wikb] The Shining (novel). From Wikipedia, the Free Encyclopedia. https://en.wikipedia.org/wiki/The_Shining_(novel). Accessed: 21.10.2021.

[Wikc] Daniel Radcliffe. From Wikipedia, the Free Encyclopedia. https://en.wikipedia.org/wiki/Daniel_Radcliffe. Accessed: 21.10.2021.

[Wikd] Elephant. From Wikipedia, the Free Encyclopedia. https://en.wikipedia.org/wiki/Elephant. Accessed: 21.10.2021.

[WPL+10] Manuela Waldner, Werner Puff, Alexander Lex, Marc Streit, and Dieter Schmalstieg. Visual links across applications. In *Proceedings of Graphics Interface 2010*, pages 129–136. 2010.

[WS11] Manuela Waldner and Dieter Schmalstieg. Collaborative information linking: Bridging knowledge gaps between users by linking across applications. In *2011 IEEE Pacific Visualization Symposium*, pages 115–122. IEEE, 2011.

[WT1] Signal Processing. From Wikipedia, the Free Encyclopedia. https://en.wikipedia.org/wiki/Signal_processing. Accessed: 07.09.2021.

[WT2] Analog Signals vs. Digital Signals. https://www.monolithicpower.com/en/analog-vs-digital-signal. Accessed: 07.09.2021.

[WT3] Difference between Analog Signal and Digital Signal. https://www.elprocus.com/differences-between-analog-signal-and-digital-signal/. Accessed: 07.09.2021.

[ZGI+17] Jian Zhao, Michael Glueck, Petra Isenberg, Fanny Chevalier, and Azam Khan. Supporting handoff in asynchronous collaborative sensemaking using knowledge-transfer graphs. *IEEE transactions on visualization and computer graphics*, 24(1):340–350, 2017.

[ZK10] Caroline Ziemkiewicz and Robert Kosara. Laws of attraction: From perceptual forces to conceptual similarity. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1009–1016, 2010.