Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science for Digital Media

Rapid Prototyping for the Digital Humanities

Master's Thesis

Hans Lienhop
Born July 2, 1994 in Verden (Aller)

Matriculation Number 114926

1. Referee: Prof. Dr. Benno Stein
2. Referee: Dr. Andreas Jakoby

Submission date: May 12, 2022

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, May 12, 2022

. . . . . .
Hans Lienhop

Abstract

Research questions in the digital humanities can often be framed as a semantic search query that has to be answered with respect to a given corpus. Such semantic search queries are usually specific and complex. Specific in the sense that it is not easy to anticipate the search query given the corpus. Complex in the sense that the semantic annotations needed to answer the query cannot be obtained algorithmically without human intervention. To account for this situation, in this thesis, a faceted search system is proposed which allows users to explore arbitrary relations between concepts. Users can introduce custom semantic concepts to the search system and manually edit the semantic annotations in the corpus. Furthermore, an API allows algorithms to access the corpus and to contribute semantic annotations. This way, the user can actively engage in generating the answer to the search query in a semi-automatic fashion. The prototypical implementation of the semantic search engine is evaluated in a user study.

Contents

1 Introduction **1**

2 Background and Related Work **3**
2.1 Related Work 3
2.2 Theory of Faceted Search 7

3 Approach **11**
3.1 Framework 11
3.2 Model 12
3.2.1 Basic Model 12
3.2.2 Extended Model 14
3.3 Faceted Taxonomy Generation 17
3.4 Taxonomy Modifications 20
3.5 Query Refinement 21
3.5.1 Query Language 23
3.6 Result Ranking 26

4 Evaluation **28**
4.1 Prototype 28
4.1.1 Notebook 29
4.1.2 Web Application 32
4.2 User Study 38

5 Conclusion **43**

Bibliography **44**

Chapter 1

Introduction

In the field of the digital humanities, as research at the intersection of the digital technologies and the humanities, many different search systems and tools are used to answer research questions of any kind, especially regarding textual corpora. While some research questions explore basic linguistic or grammatical properties of corpora[6], others are more elaborate and can not be answered trivially. Many of these more elaborate research questions can be formulated as to investigate relations between different concepts. For example, one could ask the question of how many sentences in a corpus that *intersect* occurrences of a term are *similar to* paragraphs intersecting occurrences of another term. Here, the explored relations are spatial overlap and semantic similarity. With this thesis, we offer a framework that allows iterative construction of a research environment tailored to a set of research questions that can be answered using a simple query language. We use the concept of faceted search as the underlying base of our approach.

Search engines, and more specifically faceted search systems, are ubiquitous and indispensable tools for exploring information spaces by offering means of iteratively refining search results using a faceted taxonomy. A faceted taxonomy describes a hierarchically organized set of terms, describing different aspects of underlying data. When working with textual data, usually entire documents are the focus of the exploration and the set of documents is filtered by using facets describing meta data.

In a regular faceted search system, relations between terms in facets are defined to exist if the terms index the same underlying object. Considering a relation not to be an abstract concept but rather an arbitrary but well defined association between two terms, like semantic similarity or spatial overlap, those relations may be too complex to be described by this simple definition.

Thus we want to introduce a model for faceted taxonomies which shifts the focus from exploring a set of underlying objects to exploring relations between terms. This model should be able to properly handle unidirectionality of relations to construct more complex dependencies. Also, different types of relations should be explorable in one faceted search system. This leads to a necessary variability in the used query language to properly formulate complex requests.

The digital humanities have some specific requirements for successful operation of search systems. The quality of the underlying annotations is crucial for receiving adequate results and precision and recall of the system must be as good as possible[10][2]. For this reason, we include means for a user to manually and semi-automatically alter semantic annotations during the use of the system to improve the general quality of the annotations and thus improve the quality of results and allow the user to iteratively take part in constructing the research environment.

In the following chapters, we will first introduce the theory of faceted search which serves as the basis for our model and also present some related works in the fields of faceted search and the digital humanities. Then, our model for exploring relations in a faceted search system is presented, which we consider as our main contribution. Afterwards, a prototypical implementation of the model is demonstrated and evaluated. The evaluation shows that while the framework can be an efficient tool to answer complex research questions, it is not straightforward to provide an easy and clear user interface for it.

Chapter 2

Background and Related Work

2.1 Related Work

Modern search engines adequately address the problem of what was historically called known-item search: the searched object is known and it is certain it exists in the collection one searches[25]. Exploratory search on the other hand describes the seeking of information without a clear target object or even without a well established information need. The term was first introduced in the book "Exploratory Search: Beyond the query-response paradigm" by White and Roth in 2009 [38]. It picks up existing problems from the field of information retrieval addressing vague information needs of users, like search result clustering[8] or subject search[5], which offer topical overviews in a retrieved document set. Exploratory search now shifts the focus from supporting users to find specific document sets to a general attempt to aid a user's learning and investigating of a document collection [20]. Various common problems exist not only in the field of exploratory search, but in most tools for information retrieval. Two major issues are the vocabulary problem, which describes a possible discrepancy between the user's vocabulary and the vocabulary of the indexed objects, and the challenge of receiving an overload of search results which hinders the efficient exploration[32].

Faceted search is one concept that is well established for use in exploratory search. The term *facet* is conventionally used to describe one side of a cut gemstone[23]. Each facet contributes differently to the gem's optical performance. In the context of information science a facet refers to some orthogonal attributes of underlying information objects where each facet should describe the objects from a different perspective.[11]. The term facet was originally introduced by S.R. Ranganathan, a mathematician from India, in the 1930s[24], in the process of creating colon classification which found

use in classification in libraries. He used faceted classification to organize all knowledge available in libraries using the five main facets personality, matter or property, energy, space and time. In general, faceted search is a technique that allows users to narrow down search results by iteratively applying multiple filters from a given set of categories. It enhances conventional search engines by offering navigational opportunities that allow and encourage exploratory search. Many different faceted search systems were developed in the last two decades (See [19] for a recent review of current research). While faceted search is a very broad field with many different approaches and resulting models, we will now focus on important and related search systems.

The Flamenco browser[40], which stands for FLexible information Access using MEtadata in Novel COmbinations, was one of the earliest tools for the exploration of large information spaces using faceted navigation. Flamenco uses a combination of hierarchical faceted metadata and full text search allowing users to iteratively refine their query. Figure 2.1 shows an exemplifying query in the Flamenco interface.

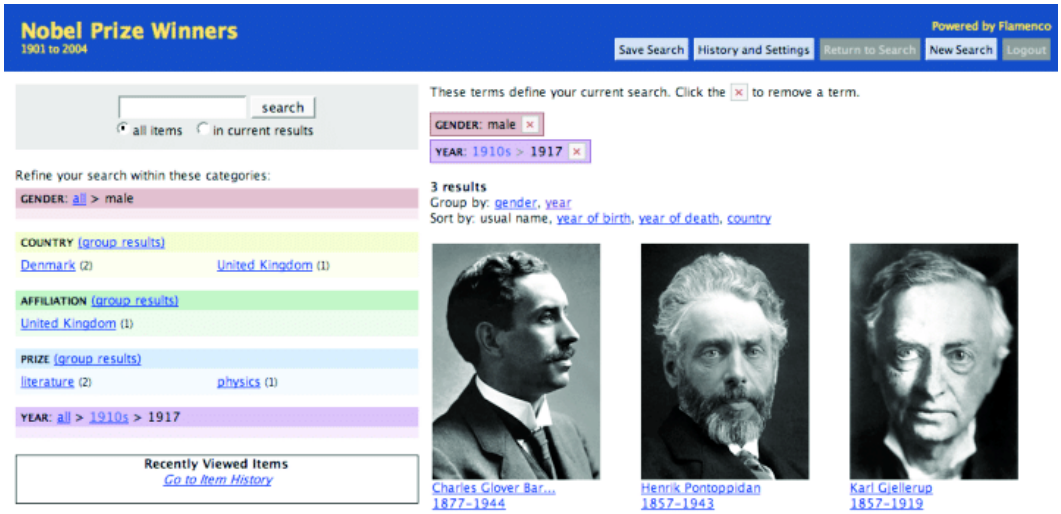


Figure 2.1: Filtering nobel prize winners with multiple facets using the Flamenco browser. Image by [39].

The mSpace explorer[28] is another notable mention. Other than most tools for faceted navigation like Flamenco or also commercially used systems for online retail, mSpace relies on a spatial multicolumn layout. This layout uses a persistent display, thus information are always present in the same window, to assist maintaining awareness of contextual information. Also, the need to

remember what appeared before the current selection is reduced. Figure 2.2 shows the mSpace classical music explorer, which is based on the mSpace model. Since we also use the concept of sequentially aligned facets in the implementation of our prototype, we consider mSpace a related work.

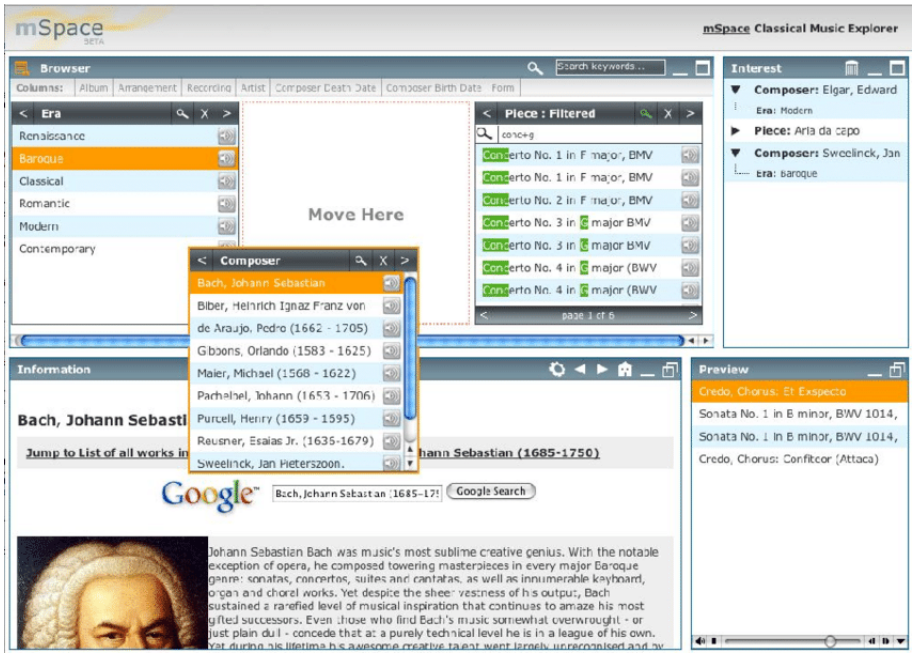


Figure 2.2: The mSpace classical music explorer, using the mSpace model. Image by [30].

The most recent and probably most related work is the publication 'Relation-oriented faceted search method for knowledge bases' by Aso et al. from 2020[1]. The authors introduce a faceted navigation for knowledge bases which includes a novel facet describing relations between entities. Predicates in a knowledge base are clustered as relations and made accessible in the navigation. Thus the focus of their system is the search for interesting relations between entities in a knowledge base, and not finding interesting entities. So, the motivations for that paper and for our work overlap. But we want to provide a most general model that does not rely on the RDF format on which most knowledge base on, but as an extension for a simple model for faceted navigation. Figure 2.3 shows the navigation of 'RelFacet', Aso et al.'s implementation of their system.

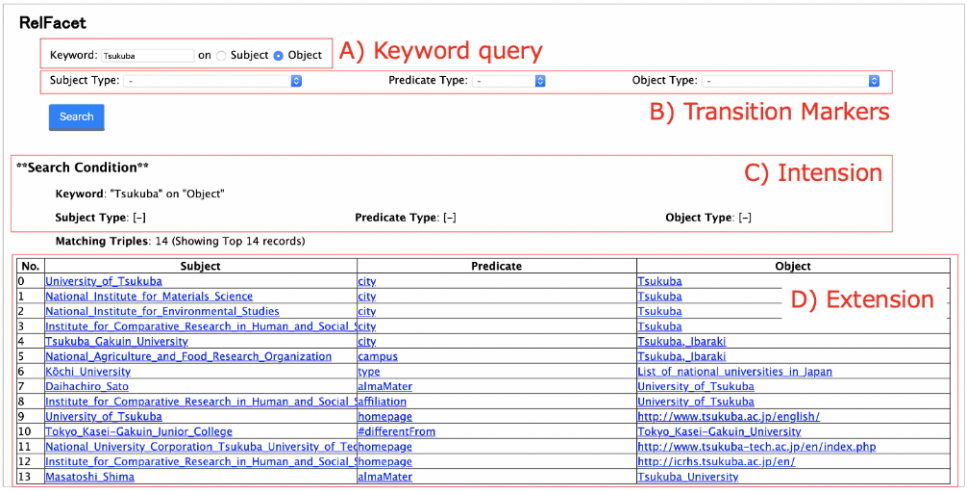


Figure 2.3: The interface of the RelFacet navigation. Image by [1].

Next to faceted navigation, annotations and improving their quality is another pillar of our proposed system. We consider an annotation to be some descriptor of a piece of data. This can be a named entity highlighted in a text, or keywords added to an image. Many free and commercial tools for annotating data exist. Most of these tools are meant for creating annotated data to use in machine learning applications. Some examples are the tools doccano[21], Labelbox[14] or Prodigy[12]. Usually such annotation tools are meant to be used collaboratively to, for example, annotate named entities in texts to use as training data. The data is not meant to be explored, but only serves as input and texts to annotate are displayed one after the other.

The tool tagtog[31], while also providing features to annotate texts as machine learning training data, also offers some explorative capabilities by supplying a concept search with which users can search the annotated corpus using boolean queries for entities that filter documents that match that query.

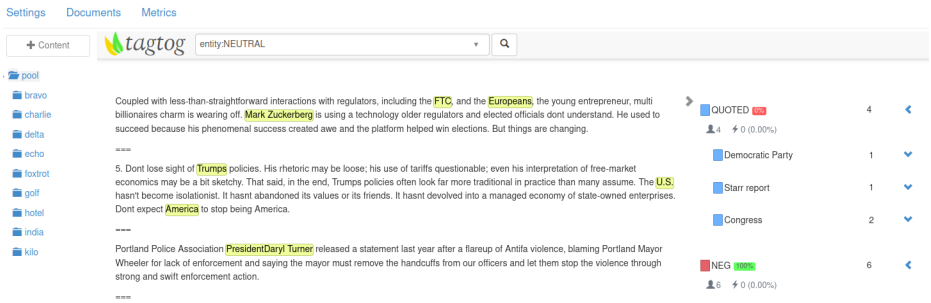


Figure 2.4: The interface of the tagtog navigation with an active query and an opened document

For this reason, we consider tagtog to be the annotation tool that is related the closest to our model and our prototypical implementation. Figure 2.4 shows an example. A document is opened with its annotations highlighted and the search for specific entity type active.

When moving away from annotation tools with the purpose of creating data for machine learning and to annotating with the purpose of corpus linguistics in the domain of the digital humanities, the tool CATMA[15] needs to be mentioned. Other than allowing users to annotate corpora, complex queries using those annotations are possible as well as statistical analysis of corpora. CATMA is a tool for statistical analysis and not for exploration, thus while using some of the same concepts, CATMA and our system differ regarding the type of research question one attempts to answer with it.

2.2 Theory of Faceted Search

To introduce a theoretical foundation, we will first formally describe a model of faceted taxonomies. We follow the notation of Sacco's and Tzitzikas' model for dynamic taxonomies[26] by defining:

Definition 1 *A **terminology** is a finite set of abstract concepts, called **terms***

Definition 2 *A **taxonomy** is a pair $(\mathcal{T}, \leq)$, where $\mathcal{T}$ is a **terminology** and $\leq$ is a reflexive and transitive relation over $\mathcal{T}$, called **subsumption***

With a and b being terms of $\mathcal{T}$ and $a \leq b$, we say that b *subsumes* a or, likewise, a *is subsumed by* b. Equivalently we also say that a is *narrower than* b, or b is *broader than* a and that a is a *child* of b. For example, $\texttt{dog} \leq \texttt{mammal}$. Thus, a taxonomy describes a tree-like hierarchy between a set of abstract terms. Given a set of taxonomies, they can be combined into a faceted taxonomy. We formally define a faceted taxonomy roughly after [33].

Definition 3 *Let $\{F_1, ..., F_k\}$ be a finite set of taxonomies, where $F_i = (\mathcal{T}_i, \leq_i)$ and the terminologies $\mathcal{T}_1, ..., \mathcal{T}_k$ are assumed to be pairwise disjoint. The pair $\mathcal{F} = (\mathcal{T}, \leq)$, where $\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i$ and $\leq = \bigcup_{i=1}^{k} \leq_i$ is called the **faceted taxonomy** generated by $\{F_1, ..., F_k\}$. The taxonomies $F_1, ..., F_k$ are called the **facets** of $\mathcal{F}$.*

Since the terminologies and subsumptions in a faceted taxonomy $\mathcal{F}$ are assumed to be disjoint, it is in fact again a regular taxonomy, the only difference being that the terminology and subsumption relation of $\mathcal{F}$ are partitioned. Choosing the criteria for partitioning depends on the chosen methodology and

the underlying *domain*. A domain describes a finite set of unique objects which are indexed by a taxonomy. Combining a faceted taxonomy with a domain results in a materialized faceted taxonomy, which we will also define roughly after Sacco[26]:

Definition 4 *A materialized faceted taxonomy* $\boldsymbol{F}$ *is a quadruple* $F = (T, O, I, Q)$ *where* $\boldsymbol{T}$ *is a taxonomy with the terminology* $\mathcal{T}$ *and the subsumption* $\leq$. $\boldsymbol{O}$ *is the set of objects indexed by the taxonomy,* $\boldsymbol{I}$ *is a function* $I : \mathcal{T} \rightarrow 2^O$ *called interpretation, and* $\boldsymbol{Q}$ *is the set of queries that can be formulated over* $\mathcal{T}$ *using standard boolean operators.*

A compound term is any subset of the terminology $\mathcal{T}$. The interpretation function I basically maps a compound term to a subset of objects in O. $I(t)$ only describes the shallow extension of the term t. While a shallow extension of t includes only the set of terms directly subsumed by t, the deep extension of t is defined as the set of all shallow extensions in the subtree with the root t. So we define

$$\bar{I}(t) = \bigcup\{I(t')|t' \leq t\}$$

as the interpretation of the deep extension of t. Considering an object $o \in O$, the *description* following an interpretation I is a set of terms describing the object. We denote the description $D_I(o)$:

$$D_I(o) = \{t \in \mathcal{T}|o \in I(t)\}$$

Consider the materialized faceted taxonomy given in Figure 2.5. While $D_I(h_2) = \{Hamburg, < 100\}$, if we consider $\bar{I}$ we have $D_{\bar{I}}(h_2) = \{Hamburg, Germany, Europe, < 100, < 200\}$.

We see that the resulting compound term is either only the set terms whose interpretation include the described object when using the interpretation I, or the set of terms where the interpretation of any term in its deep extension includes the object. We call the latter the complete description of h_2. While we can use interpretations to define descriptions as we did above, it is also possible to define interpretations through descriptions. Using a function $D : Obj \rightarrow 2^{\mathcal{T}}$, mapping objects in O to terms, we can define an interpretation I_D:

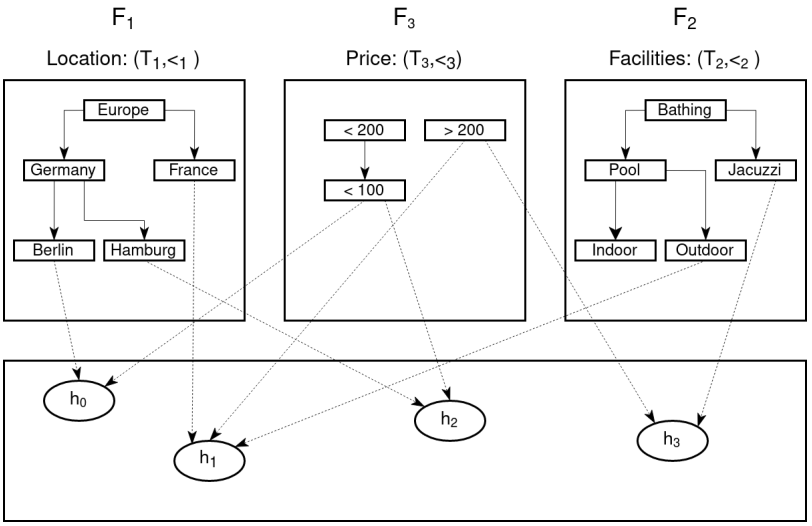


Figure 2.5: An example for a materialized faceted taxonomy $\mathcal{F} = \{F_1, F_2, F_3\}$ comprised of three facets to describe an underlying domain of objects $O = \{h_0, h_1, h_2, h_3\}$ characterizing hotels.

$$I_D(t) = \{o \in O | t \in D(o)\}$$

While both methods are possible, it seems more natural to consider interpretations being defined by descriptions, since often objects are accompanied by a set of tags which serve as descriptions and can be used to define interpretations.

Navigating the information space in a faceted taxonomy is done by the notion of a users $focus$. A focus can be any query or compound term, although we will here first concentrate on simple conjunctions of terms. The contents of a focus f is the resulting set of objects $\bar{I}(f)$. While the initial focus can be for example the empty compound term, resulting in an empty set of objects from the interpretation function, it could also be the top term of a facet. The focus of a user can be refined using the concept of *zoom points*. While there are more types of zoom points, we limit ourselves to defining $zoom - in$ and $zoom - out$. Given a faceted taxonomy and a focus f we define the whole set of zoom points regarding a facet F_i as $AZ_i(f)$:

$$AZ_i(f) = \{t \in \mathcal{T}_i \mid \bar{I}(f) \cap \bar{I}(t) \neq \emptyset\}$$

Thus, zoom points are all the terms in a terminology whose selection results in a non-empty set. Usually, a zoom point t_x is accompanied by the count of objects that selecting the point would yield. This count is simply the cardinality of the set $\bar{I}(f) \cap \bar{I}(t_x)$. A focus may have contributions from different

facets. Considering f to be simple compound term, we denote $f_i = f \cap \mathcal{T}_i$ as the contribution of some facet F_i to the focus f, where $\mathcal{T}_i$ is the terminology that facet. We define candidate zoom-in points of a facet F_i to be the direct children of f_i, or, the immediately narrower terms. Given a focus f and the facet F_i, we denote candidate zoom-in points by $CZ_i(f)$. Since we want to avoid empty sets, the set of zoom-in points of a facet F_i is denoted as follows:

$$Z_i(f) = \{t_x \in CZ_i(f) \mid \bar{I}(f) \cap \bar{I}t_x \neq \emptyset\}$$

When a zoom-in point t is selected, the focus is updated as $f = f \cup \{t\}$, and all terms broader than t removed from f. In the case of zoom-out, a term t can be deselected from the focus. The term t can then replaced by its immediately broader term or simply removed. Since t was part of the focus and thus did not produce an empty set, it is trivial that its broader term would also not produce an empty set.

Chapter 3

Approach

In the following sections, we will present the theoretical foundations for our model. First, we will introduce a basic model as a modification for Sacco's model for dynamic taxonomies. Then, we extend that model for a specific use case to be applied in the field of digital humanities.

3.1 Framework

In this section, we formalize the framework our for approach to fulfill the previously illustrated requirements. Wei et al. proposed a general framework for faceted search systems[37] which we will apply in a modified form. It is illustrated in Figure 3.1. The original framework consists of three main modules. Namely, the generation of a faceted taxonomy, query refinement and

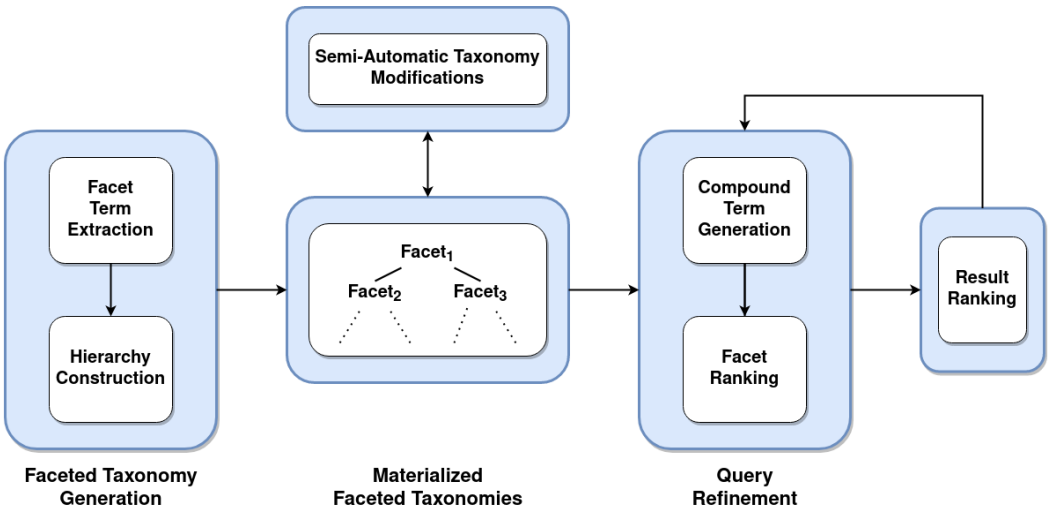


Figure 3.1: The general framework for the faceted search system.

result ranking. These cover the key technologies facet term extraction, hierarchy construction, compound term generation, and facet ranking. We add the module of taxonomy modification to the framework as an additional component. In the following sections we will describe those components and how we utilize them in a tool for rapid prototyping.

3.2 Model

3.2.1 Basic Model

We use Sacco's model for dynamic taxonomies, introduced in Section 2.2, as a basis for our approach. Thus, we work with a materialized faceted taxonomy consisting of a faceted taxonomy F with a taxonomy T having a terminology $\mathcal{T}$ and a subsumption $\leq$, a set of objects O, an interpretation function I, and a set of queries Q. Q is the set of queries that can be formulated over $\mathcal{T}$ using standard boolean operators.

Following Sacco et al., a connection between the taxonomy and the underlying set of objects is achieved by the description function D and the interpretation function I. Objects are described by some terms in the taxonomy, and terms in the taxonomy can be interpreted to reach some objects. For our model, we now want to shift the focus from exploring objects to exploring the relationships between terms. Sacco introduced the the *base extensional inference rule* to describe relations between terms and defined it as follows:

Definition 5 *Two terms A and B are related iff there is at least one object o in the extension which is classified at the same time under A or under one of A's descendants and under B or under one of B's descendants.*

Thus, A and B are related if there is some object o with its description $D_{\bar{I}}(o)$ including both the terms A and B or some terms from their deep extensions. This definition of relations between terms has two major drawbacks. First, it can only represent bidirectional relations between terms. Secondly, it does not allow multiple different relations. Because our focus lies on the exploration of those relations, a more nuanced representation is necessary to also model unidirectional relations of different kinds between terms. Thus, we break with the base extensional inference rule and introduce our own concept of relationality between terms. For this, we consider the collection of objects O in a materialized faceted taxonomy to consist of sets describing relations. A relation is not inherently defined and can be any association between two

terms. We introduce a new function to the model of dynamic taxonomies, the reference function R. Each object references exactly one node and contains all relations that the referenced term has with other nodes. The function $R(o)$ returns the node assigned to the object o. The set O can contain relations of many different varieties. We denote the existence of a relation of the kind r between two terms A and B as follows:

$$A \bowtie_r B$$

Note here that $A \bowtie_r B$ does not imply $B \bowtie_r A$. Relations are directed, and just because A somehow relates to B does not necessarily mean that B also relates to A in the same way. $A \bowtie_r A$ may hold true depending on the relation, so a leaf node may relate to itself. Thus, the description of any object o with respect to a relation r is denoted as follows:

$$D_{I,r}(o) = \{t \in \mathcal{T} \mid R(o) \bowtie_r t\}$$

$D_{I,r}(o)$ only includes the specific nodes to which $R(o)$ has a relation to. We consider relations between terms to only exist between leaf nodes, so terms that subsume no other terms. So, the terms in the description $D_{I,r}(o)$ of o are exclusively leaf nodes. To extend the descriptions to include terms from the entire set $\mathcal{T}$, using $D_{\bar{I}}$ is necessary to include each terms deep extension. We denote the set of leaf nodes of the deep extension of any t as $L(t)$:

$$D_{\bar{I},r}(o) = \{t \in \mathcal{T} \mid L(t) \cap D_{I,r}(o) \neq \emptyset\}$$

Thus, the description of any object comprises all those terms that the referenced term has a relation to. In the other direction, we can define the interpretation function for any term t with respect to the description D and a relation r as follows:

$$I_{D,r}(t) = \{o \in O \mid t \in D_{I,r}(o)\}$$

$$\bar{I}_{D,r}(t) = \{o \in O \mid t \in D_{\bar{I},r}(o)\}$$

Consider as an example the materialized faceted taxonomy presented in Figure 2.5. To represent that taxonomy with our model, the previous set of objects, which was composed of hotels described by the taxonomy, is now included as a regular facet. The described representation is shown in Figure 3.2. Note that we removed the facet `Facilities` simply to reduce clutter. Each leaf node is referenced by an object that describes relations between the referenced term and other leaf nodes in the taxonomy. In this example, only one type of relations exist between objects. The objects referencing hotels

comprise those terms which were part of the description of the hotel objects in Figure 2.5. But all other leaf nodes are referenced by an object as well which is described by all terms to which the referenced term relates, which in this case is only hotels. Consider the object $\texttt{R}_0$ which references the term `Berlin`. Since in Figure 2.5 the object $\texttt{H}_0$ was described by `Berlin`, $\texttt{R}_0$ includes $\texttt{H}_0$ in its description. Also $\texttt{R}_4$, the object referencing $\texttt{H}_0$, includes `Berlin` in its description. So in this case, $\texttt{Berlin} \bowtie \texttt{H}_0$ and $\texttt{H}_0 \bowtie \texttt{Berlin}$, but directionality is given due to each leaf node being referenced by an individual object.

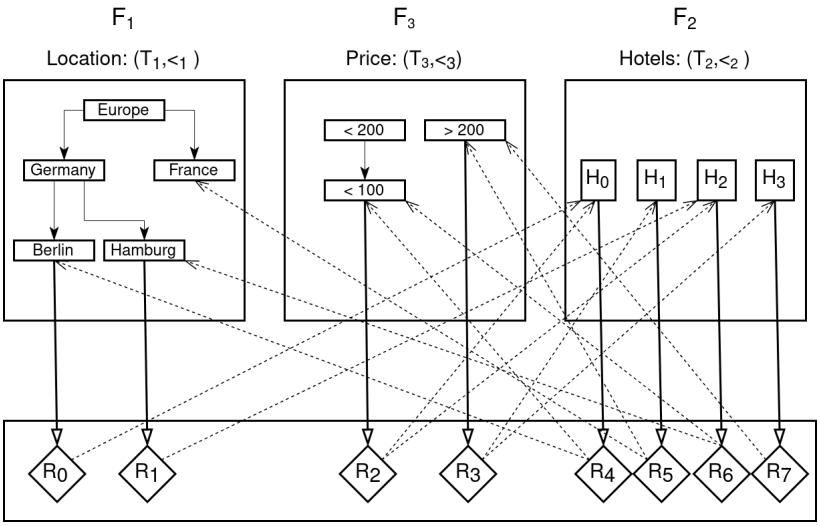


Figure 3.2: Transforming Figure 2.5 to conform to the modified model. The facet `Facilities` was removed for reasons of simplicity.

One can consider this model as applying some methodology from knowledge graphs to classical faceted navigation. All $is-a$ relations are organized in a faceted taxonomy, while all elements in the leaf nodes are connected by some relation, thus constructing an illustration of a knowledge graph. Our contribution here consists mainly in using relation objects as the domain of a materialized faceted taxonomy. This allows to apply all operations and concepts from faceted navigation to explore relations between terms.

3.2.2 Extended Model

To fully utilize the possibilities of the basic model, we extend it further by defining the concept of *spans*.

Definition 6 *A **span** is a non-empty set ϕ of data points in an n-dimensional space.*

So in the most general way, a span can be seen as any set of data points. For our extended model of a faceted taxonomy, all but the narrowest terms of $\mathcal{T}$ are simply described by labels, or abstract concepts. The narrowest terms, or leaf nodes, are spans which belong to the set $\mathcal{S} \subset \mathcal{T}$. A term can subsume either labels or spans, but not both. Since all spans are leaf nodes, they have no children. For our extended model, we only consider the existence of relations between leaf nodes in the taxonomy. All other terms are considered to be abstract terms to simply divide the terms in a facet into meaningful units. Thus, in our extended model, objects consequently only comprise relations between spans. A relation can exist between any two spans whereas the condition for a relation to exist is not inherently defined but relies on the requirements and the underlying data.

As an example, let us consider the exploration of a textual corpus. When working with textual data, a natural way to represent that data could be by using *character spans*, describing segments in a textual data source. We define character spans as follows:

Definition 7 *A **character span** is a span $\phi = \{S, E, R\}$, where S is the start character, E is the end character, and R is the underlying data source.*

There can exist many relations between character spans. In our example, we consider *intersections* as relations. An intersection between two character spans exists if they are fully or partially overlapping. Note here that the following definition holds only for character spans on linear data sources like traditional text or audio. Multidimensional data like images require a more complex definition.

Definition 8 *An **intersection** exists between two character spans A and B iff $A_R = B_R$ and $B_E \geq A_S$ and $B_S \leq A_E$, or vice versa.*

We denote the existence of an intersection between two spans A and B as follows:

$$A \bowtie_i B$$

Using the definition of intersections and given a set of character spans, we now can construct a set of objects O consisting of relation objects which in turn hold a reference to a span and its intersections. A schematic overview of a faceted taxonomy using character spans is pictured in Figure 3.3. Notice that the representation currently misses a domain O. The hierarchically organized elements denoted with letters represent regular terms, while the highlighted

areas on the bar serve as spans. The term A describes the root node. Note here that despite using a root node and essentially merging different taxonomies, we still consider the model to be a faceted taxonomy, partitioned into the imminent descendants of the root node. The spans are actually part of the taxonomy and subsumed by their respective parent. The visualization is chosen to emphasize the concept of intersections between character spans. The intersections, and thus relations, between spans become clearly visible and exist wherever the highlighted areas overlap each other.

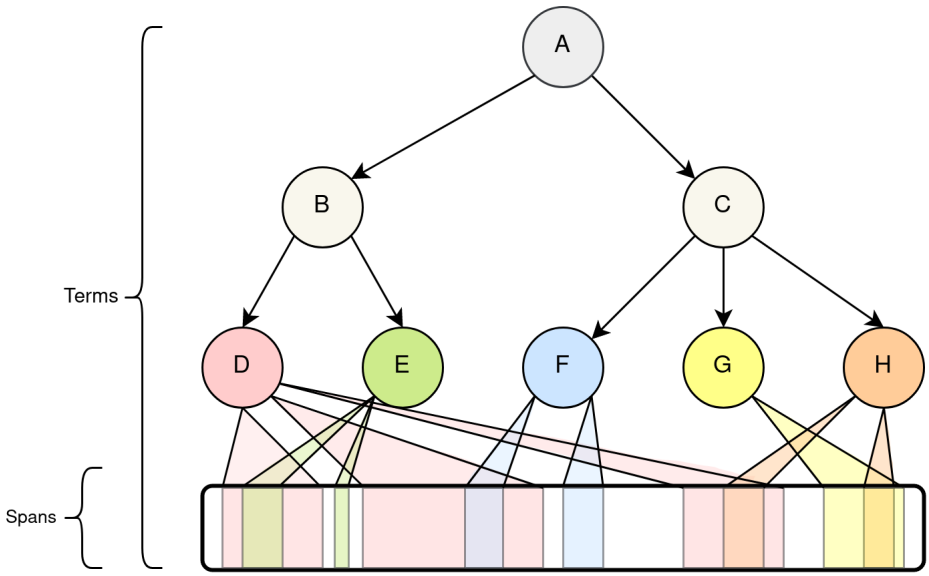


Figure 3.3: A faceted taxonomy with character spans as leaf nodes. The spans are visualized as ranges on a linear data source to highlight intersections.

On the basis of this visualization, we can now construct the full materialized faceted taxonomy with intersections as relations which are represented in the underlying set of objects O. Figure 3.4 displays this materialized faceted taxonomy. Notice that the set of spans $\mathcal{S}$ is now integrated in the taxonomy just as the regular terms. Each span is now referenced by an object in O. All intersections between ranges visible in 3.3 are considered as relations and added to the objects referencing the respective spans. Note here that the relation in this case is commutative as well. The relation could be easily transformed to being one-directional by removing *or vice versa* from Definition 8. Now, a span only intersects another span if it is encompassed by another span, the encompassing span does not intersect the other. Considering for example, Figure 3.4, now $\mathtt{E}_0 \bowtie_i \mathtt{D}_0$, but not the other way around.

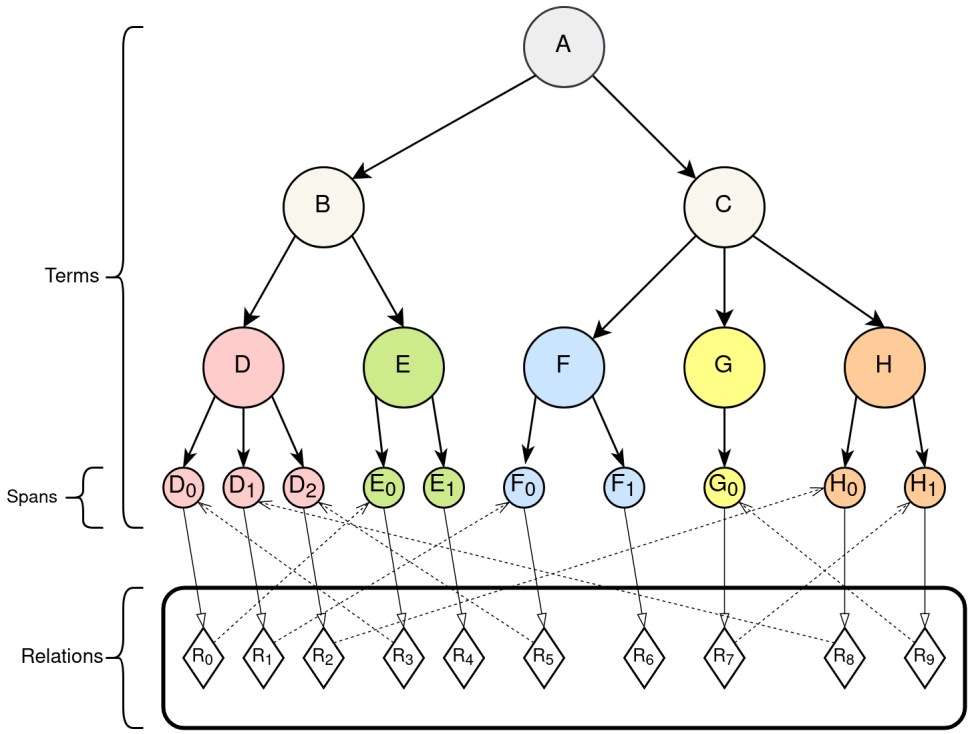


Figure 3.4: The materialized faceted taxonomy resulting from Figure 3.3. The set of relations consists of all intersections between spans. Some relation objects are empty, if the referenced span has no intersections.

The presented model offers the possibility of a domain which moves away from actual objects described by the taxonomy being the area of interest, but relations between terms. Considering available faceted search systems for the exploration of textual corpora[16], conventionally, the explored set of objects is static and usually consists of the set of the full documents. In the following sections, we will continue to focus on the application of the model on textual data sources.

3.3 Faceted Taxonomy Generation

The generation of a faceted taxonomy traditionally consists of two tasks. The extraction of viable facet terms from the underlying data, and the construction of a hierarchy. Our model requires additional steps, namely the extraction of relevant spans and the identification of their relations. Depending on the research goals and the underlying data, these steps may differ a lot. In this section, we want to give an example for the faceted taxonomy generation for collections of texts. Thus we consider spans to be character spans and

the relation between spans to be commutative intersections. Textual corpora are often separated into structured and unstructured data. By structured we mean data that can be precisely assigned to a field. In the case of a textual document collection, this includes metadata like author names and publishing date. On the other hand, while being an imprecise term, we define unstructured data as data that has not been clearly identified and is, as such, not part of any field. In the case of a textual document, this is the actual body of the text. Conventional faceted search interfaces allow faceted navigation using the structured data to filter documents with an additional tool to perform a full text search in the unstructured data [32]. Using the concept of spans as leaf nodes in a taxonomy, we have the opportunity to apply structure to the unstructured data.

At first, all relevant meta data describing objects from the corpus need to be identified. Meta data consists of key-value pairs, with keys being i.e. "Author" and "Year". All keys are added as a facet to a joint faceted taxonomy, with all unique values of a key being its shallow extension. For example, a subsumption in the form of `1994` $\leq$ `PublishingYear` would be possible. While extracting those meta information, the entire length of the underlying data source is added as a span as a child of each value. So the previous example of `1994` $\leq$ `PublishingYear` would receive another subsumption, narrower than `1994` containing a span S with S_R being the underlying text, S_S being 0, the start character, and S_E being the last character of S_R, thus S spans the entire text. So, each meta value V subsumes a set of spans $\mathcal{S}_V$, each of those spans referencing the content of a data source described by that meta value.

After extracting meta data, the actual content of the texts is analyzed and added to the faceted taxonomy. As already mentioned, conventional faceted exploration tools perform operations to filter a static set of underlying data sources. To fully utilize all aspects of our model, text segmentation is performed on every data source. Using different granularities, texts are divided into meaningful units. These units can be for example entire documents, pages, paragraphs or sentences. Note that the focus of this work is not the automatic detection of meaningful units in texts. While the area of detecting sentences or paragraphs in texts is well researched but not without flaws[9][13], we assume a perfect division into the desired elements for the theoretical model. Consider now the currently investigated unit to be sentences. A facet `sentences` is added as a child to the root node of the taxonomy. While iterating through the entire document collection, each detected sentence is added as a term to the taxonomy, subsumed by the

facet `sentences`. Now, the span describing the location of the sentence in the data source is identified. This span is added as child of the newly added term. Although the spans could be added directly as children of the facet `sentences`, since they are spans describing sentences, adding the additional layer of terms containing the individual sentences helps with repetitions. For a reoccurring sentence, no new term would be added, but the already existing term would gain the new span as child. This process can be applied for any meaningful unit that covers entire documents, or for non sequential units, like domain specific descriptions. These could be for example `research reports` or `abstracts`, depending on whether a viable method for extracting those is available.

A similar approach is taken in expanding the faceted taxonomy by areas of interest for the researcher. Depending on the research question, relevant facets need to be identified. If, for example, the corpus and the research question are of medical nature, one additional facet could be `Diseases`, whose subsumptions are different illnesses related to the research question. The method of creating such hierarchies is not fixed, but they could be constructed manually by the domain expert, or, for example, by using the means of a semantic network like BabelNet[22]. Once this hierarchy is constructed, relevant spans for each disease must be identified in the underlying corpus to create leaf nodes in the taxonomy. Identification could be performed for example by simple full text searches, regular expressions or machine learning models. We call spans that do not describe any meaningful textual segments, like sentences or paragraphs, *semantic annotations*.

In the next step, the indexable set of objects needs to be constructed. As already mentioned, each span is referenced by an object describing the relations of that span with other spans. So for each span, an object o is added to the set O which references the respective span. In the case of our example, these relations are intersections. Thus, following Definition 8, we identify all pairs of spans a and b where $a \bowtie_i b$, thus all pairs of spans between which there exists a relation from a to b. Again, in our example the relations are commutative. Now we identify the object o where $R(o) = a$ and add b to its description. The constructed set O now serves as the domain of a materialized faceted taxonomy.

3.4 Taxonomy Modifications

After constructing a faceted taxonomy, the entire underlying data is navigable by means of faceted navigation. In the case of the example of a textual corpus, this is achieved by extracting all desired types of text segments and annotations. Thus, when reaching the layer of leaf nodes, one or many character spans will be displayed. Spans are abstract concepts, and each type of spans must have specific rules for displaying them. Since a character span is made up of a start character, an end character and the reference to a data source, the text segment referenced by the span will be shown. For example, only all occurrences of a specific sentence are displayed. Naturally, depending on the data and the chosen type of span, this representation differs.

Another important aspect of our approach is to offer a user means to actively engage in the modification of the taxonomy. Depending on the research interest, the quality of the annotations is crucial. In our case, this is equivalent to improving precision and recall of the faceted search system. While precision describes how many of the retrieved items are relevant, recall describes the ratio of all relevant items that is retrieved. Thus, when improving precision and recall, one wants to reduce the number of false positives and false negatives respectively. Depending on the chosen methods during the initial generation of the faceted taxonomy, the quality of the annotations may be not sufficient and a query would return too many false negatives or false positives, if annotations are either wrongly detected or missed. Improving the quality of queries is the incentive for including dynamic taxonomy modifications in our faceted search system.

There are multiple modifications that can be performed in the taxonomy. Terms and spans can be both added and removed. When adding a regular term, it is simply inserted in the hierarchy. Note here that a term can only either subsume other regular terms or spans, so the locations for inserting terms is limited. When removing a term, not only the term itself is removed from the taxonomy but also the entire deep extension of it. Should spans be included in the deep extension, the set of objects O needs to be recomputed since some objects may reference removed spans or relations between spans may not exist anymore. The locations for adding spans are also limited, since they can only be inserted as leaf nodes. In both cases of either adding or removing spans the set of relations O needs to be recomputed.

As an example, we continue with a textual corpus as underlying data. Consider a character span S with the reference S_R ranging over the interval

between S_S and S_E. Since the span is just a reference, it is necessary to create an ordinary string out of it to display properly. We recognize this string to have a start character 0 and a length of $S_E - S_S$. Consider now a substring of this string to be a false negative annotation, so it should be annotated but is not. Now the false negative substring can be marked and added as a span S^* to its designated parent term. Trivially, $S_R = S_R^*$ since the newly added span S^* shares the underlying data source with the original span S. Consider i_s to be the start character and i_e to be the end characters of the actual substring in the string. Now $S_s^* = i_s + S_S$ and $S_E^* = i_e + S_S$. The span is now part of the faceted taxonomy and it is possible to reference it in its data source and access it by means of faceted navigation. Now, all intersections between S^* and the entire set $\mathcal{S}$ are determined and the set of objects O updated. This technique can be used to improve the quality of the annotation on while exploring the taxonomies and increase the precision of future queries. False negative results can simple be removed from the taxonomy on encountering to improve the recall.

3.5 Query Refinement

As described in Section 3.2, in our model, the set of objects O in a materialized faceted taxonomy consists of objects referencing a span and being described by relations of that span to other spans. The result of the interpretation function of any term t in a materialized faceted taxonomy is the subset of objects in O that share at least one span with any span in the deep extension of t. One part of the faceted taxonomy is the set of queries Q that can be formulated over $\mathcal{T}$ using standard boolean algebra. Thus, it contains all queries described by the grammar $q ::= t \mid q \wedge q' \mid q \vee q' \mid \neg q \mid (q) \mid \epsilon$, where t is a term in $\mathcal{T}$ and ϵ the empty query. Since our model uses the same notions of interpretation and description as Sacco's original model, the interpretation I can easily be extended to queries. So, queries can be interpreted just like regular terms and produce a set of objects as follows:

$$I(q \wedge q') = I(q) \cap I(q')$$

$$I(q \vee q') = I(q) \cup I(q')$$

$$I(\neg q) = O - I(q)$$

Note that the $-$ in the last equation denotes set difference. Thus, the answer of a query q is the set of objects that is produced by $\bar{I}(q)$. The way in which the resulting set is used is dependant on the kind of objects and often

trivially deducible when using conventional objects, like articles or products. In Sacco's model, the set of objects is not inherently defined but only their connection to the taxonomy by the description and interpretation. Other attributes of the object are irrelevant for the navigation, only for displaying them and further interaction. In our model, the objects are fully defined. Other than their description, the only attribute they have is the reference back to their respective span. Now the resulting set of a query is a collection of objects that have no attributes that can trivially be used for displaying them or interact with them further.

When considering a set of objects O, we can use the references $R(o)$ for each object $o \in O$ to produce a usable result. Consider a query q which acts as a focus in the materialized faceted taxonomy, r as some relation that is explored and $I_r(q)$ as the set of objects in which the interpretation of the query results. Thus we denote the set of terms $T_{I,r}(q)$ that are referenced by $I_r(q)$ as follows:

$$T_{I,r}(q) = \{R(o) \mid o \in I_r(q)\}$$

Thus the result of a query can be interpreted as a set of spans. It is worth to note that when considering a materialized faceted taxonomy with relations between spans, the set of terms $T_{I,r}(q)$ can be used to construct a new taxonomy that can be utilized to access the results of the query. In other words, the result of a query are simply all terms in the taxonomy to which the query relates. Formally, we define the terms $\bar{T}_{I,r}(q)$ that are included in the new taxonomy as follows:

$$\bar{T}_{I,r}(q) = \{t \in \mathcal{T} \mid L(t) \cap T_{I,r}(q) \neq \emptyset\}$$

Where $L(t)$ are all the leaf nodes in the deep extension of t, thus only spans, and $T_{I,r}(q)$ the set of terms that are referenced by the query q with respect to the relation r.

As an example, we consider a simple query applied to the schematic visualization in Figure 3.4. The application of the query is visualized in Figure 3.5. Highlighted in green are the query term and those spans whose relations are considered in the query. Highlighted in pink are the relations that result from the interpretation of the query and the terms that are included in the set $\bar{T}_{I,i}(q)$ with respect to the intersection relation which we denote by i. Consider a query q consisting solely of the term $\mathtt{D}$. Now, $\bar{I}_i(q)$ utilizes all spans in the deep extension of $\mathtt{D}$, so $\mathtt{D_0}$, $\mathtt{D_1}$ and $\mathtt{D_2}$. Ergo, $\bar{I}(q) = \{\mathtt{R_0}, \mathtt{R_1}, \mathtt{R_2}, \mathtt{R_3}, \mathtt{R_5}, \mathtt{R_8}\}$ and $T_{\bar{I},i}(q) = \{\mathtt{D_0}, \mathtt{D_1}, \mathtt{D_2}, \mathtt{E_0}, \mathtt{F_0}, \mathtt{H_0}\}$. To construct a new taxonomy, we consider the set $\bar{T}_{I,i}(q)$ with respect to the query q consisting of the term $\mathtt{D}$. With

$\bar{T}_{I,i}(q) = \{\mathtt{D_0}, \mathtt{D_1}, \mathtt{D_2}, \mathtt{E_0}, \mathtt{F_0}, \mathtt{H_0}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{H}, \mathtt{B}, \mathtt{C}, \mathtt{A}\}$ and their respective relations in the taxonomy $T = (\mathcal{T}, \leq)$, a reduced taxonomy can be constructed that only includes terms in whose deep extension are spans to which the spans in the result of the query q relate to.

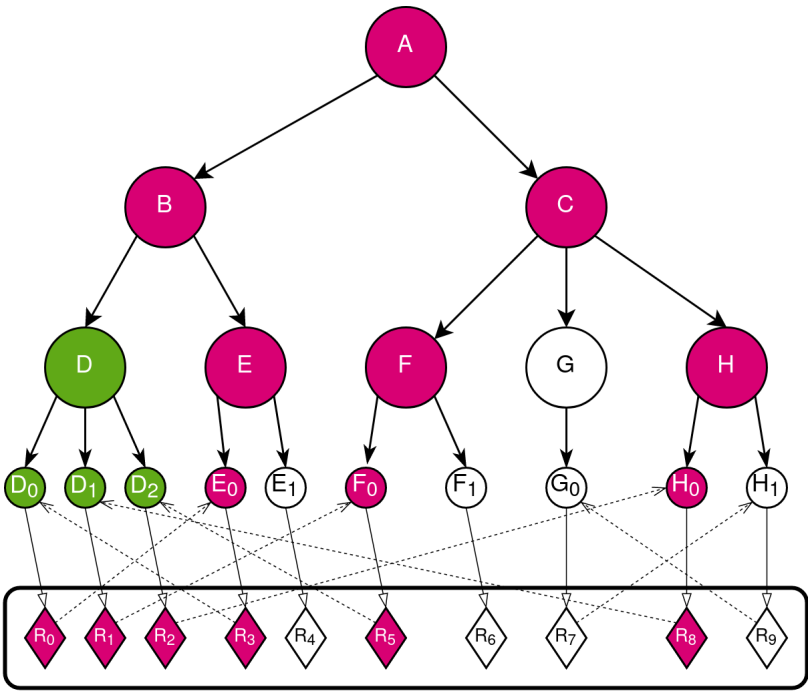


Figure 3.5: The query consisting only of the term D is applied. Highlighted in pink is the resulting taxonomy which is created in addition to the selection in green.

The resulting taxonomy can now be used to find the answer to a research question posed as a complex query.

3.5.1 Query Language

When formulating a research question, one often has not a full taxonomy in mind as a result but a specific term and how other terms relate to it. For this reason, each query has to include a so called *target*. A target describes the term that is investigated by a query. While it is possible to apply any boolean query to the materialized faceted taxonomy, desired results are often only achieved by using a specific pattern. Consider a taxonomy with the terms A, B, C, D and E each with any number of spans, relations of type r between them and a research question that wants to find all instances of A to which instances of B and C are related to. Now it is an important notion that even though there may not be any relation between B and C, the query $\mathtt{B} \wedge \mathtt{C}$ returns

all those spans $s \in \mathcal{S}$ where $s \bowtie_r \texttt{a}$ and $s \bowtie_r \texttt{b}$ for any $\texttt{a} \in \texttt{A}$ and $\texttt{b} \in \texttt{B}$. In the newly created taxonomy from the filtered spans a focus is set to examine the subtree with the target as a root, so the target becomes the focus of the new taxonomy. In other words, the target is added as an additional clause to the query, so the full query is now $\texttt{A} \wedge (\texttt{B} \wedge \texttt{C})$. Note that using standard boolean operators, we have to assume a definitive relation between terms without the ability to specialize it further.

While logical operators are a great tool to mathematically display these queries, they are tedious for non-expert users to interact with and it is virtually impossible to handle multiple different relations in a single model using only standard boolean operators. Thus for any user to be able to formulate such complex queries, we introduce a small query language. Common query languages like SQL[7] or SPARQL[35] or expression languages like XPath[36] offer many possibilities to formulate complex queries to access data organized in specific structures. Due to the variability of the possible relations in combination with the small amount of necessary operations, a new and small query language is a viable option which adopts some established concepts while introducing its own necessary complexity.

First of all, let us replace logical operators with their natural language equivalents `AND`, `OR` and `NOT` respectively, modifying our example query as follows:

```
A AND (B AND C)
```

In the previous examples we always only used a single relation between spans in a model. Now it is possible that multiple different relations are included in the same model, so each span referenced by multiple objects, in which case one would have to identify the one relation between terms one is interested in examining. Now the keyword for the desired relation is dependant on the requirements and not inherently defined. Considering a model with character spans and intersections as relations, the query can look as follows:

```
A INTERSECTING (B AND C)
```

Note that between `B` and `C` the keyword `AND` remains, but the relation keyword describes how the query executes. It now returns all spans of `A` with which any spans of `B` and `C` *intersect*. This keyword could be replaced by any relation included in the model. The keyword describing the relation can be be prefixed by a `NOT`, to include all instances of `A` which do not intersect with any instance of `B` and any instance of `C`. Other operators do not make sense at this

point. We call any construction of a relation in combination with a regular boolean query, like `INTERSECTING (B AND C)` a sub-clause. Any number sub-clauses can now be appended to the query to further filter the amount of target spans. While the first sub-clause can only have the keyword `NOT` prepended, all others can also be prefixed by `AND` or `OR` to specify the desired operation to filter the target spans. Consider an additional relation in the model called `SIMILAR TO`. It constructs a relation between all spans which are, according to some definition, semantically similar. We can now extend the previous query by introducing a new sub-clause as following:

```
A INTERSECTING (B AND C) OR SIMILAR TO (D NOT E)
```

This now rather complex query would return all those spans in `A` that intersect both any span `B` and any span in `C`, are similar to a span in `D` but not to any span in `E`, or both. Since the result of such a query is again just a set of spans, one can use a complex query as part of a sub-clause in another query. We can construct an example for a nested query as follows:

```
A SIMILAR TO (B) AND INTERSECTING (C INTERSECTING (D OR E))
```

This complex query would now return all those spans in `A` which are similar to any span in `B` and intersect any span in `C` which intersect any span in `D` or `E`.

How any query is applied now defined, but not how the returned result should be handled. An example could be a simple get-command which is comparable with SQL's `SELECT` keyword. But since the operation describes from a user's perspective less of a select but more of a search operation, we call the standard get-command `FIND`. Thus, to transform all previously defined queries into get-operations, they simply need the keyword `FIND` as a prefix. Many other operations are possible. For example, a `COMPARE...WITH` operation could take two queries as input and return some statistics that show differences between the results of the queries, like the amount of the results or their ranking.

Using these tools, a great number of complex queries can be constructed, or, possibly, research questions formulated as such queries.

3.6 Result Ranking

After applying a query and receiving some result, those results must be ranked in some way depending on their relevance to allow users to find those results that are most relevant to their current query and more generally their research question.

After applying a query that has a term as target which does not directly subsume leaf nodes, it returns a taxonomy. In this case, one the most trivial way to rank those facets is by using the notion of the object count. The remaining terms in the new taxonomy can be considered simply as zoom points t_x and as such are accompanied by the count of objects that selecting the point would yield. This count describes the imminent children, so, the amount of directly subsumed regular terms or directly subsumed spans. Thus it would be the cardinality of the set of terms in the shallow extension of t_x in the newly created taxonomy. It would also be accompanied by an additional count referring to the total count of direct children in the original, unfiltered taxonomy. This way, both a relative and an absolute ranking of the terms can be achieved.

Result ranking is the process of ranking the individual spans that are returned by a query. Depending on the investigated relation, different approaches are possible. An obvious method that can be applied irrespective of the type of relation is to investigate the amount of spans that each span in the target facet is related to and use that for scoring. Such a score for an object $o \in O$ with respect to a query $q \in Q$ and an inspected relation r can be computed as follows:

$$Sc_{q,r}(o) = |\{s \in L(q) | s \in D_{I,r}(o)\}|$$

Thus, the score of a relation object o is the cardinality of the set including all those spans that describe the leaf nodes of the query, thus, all those spans that are considered for relations between terms, which also are part of the the description of the inspected relation object. So, this score would rank objects, and in turn spans, higher, depending on the amount of terms which are related to it. This score can of course be computed for any term in the newly constructed taxonomy in a query, counting the relations that any spans in the deep extension of the term have with other terms. Thus, for some term $t \in \bar{T}_{I,r}(q)$, the terms of the resulting taxonomy after a applying a query, the score can be computed as follows:

$$Sc_{q,r}(t) = \sum_{o \in \bar{I}_{D,r}(t)} Sc_{q,r}(o)$$

While simply counting the relations between objects is a method of scoring that is universally usable for any kind of relation, depending on the type of relation, the measuring of a score can be adjusted to fit the requirements of that relation. For example, intersections between character spans can be regarded as boolean relations. Either an intersection exists between two spans, or it does not. The previously defined score may already suffice for ranking. Another measure for relation already mentioned was the similarity between spans. The similarity could be also regarded as a boolean relation when considering a threshold above which a span is considered similar to an other. But similarity can also be considered to be a numeric value depicting not if two spans are similar, but *how* similar they are. Of course, still only relations above a certain threshold must be considered. The score $Sc_{q,r}(o)$ of an object o could now not only count the relations but may consist for example of the sum of similarities, the average similarity or the highest singular similarity. Thus, each relation may offer its own possibilities for ranking the results of a query.

Chapter 4

Evaluation

In the following sections, we will introduce a prototype implementing the presented model in a basic form. Then, we will evaluate the prototype and present the results of a small user study concerning the usability and user experience.

4.1 Prototype

For the implementation of a prototypical application, we want to include the main building blocks of a faceted search system described in Section 3.1. These building blocks are the generation of a faceted taxonomy, with the extraction of facet terms and the construction of a hierarchy, applying modifications to the materialized faceted taxonomy, and provide the possibility to conduct complex queries and rank the received results.

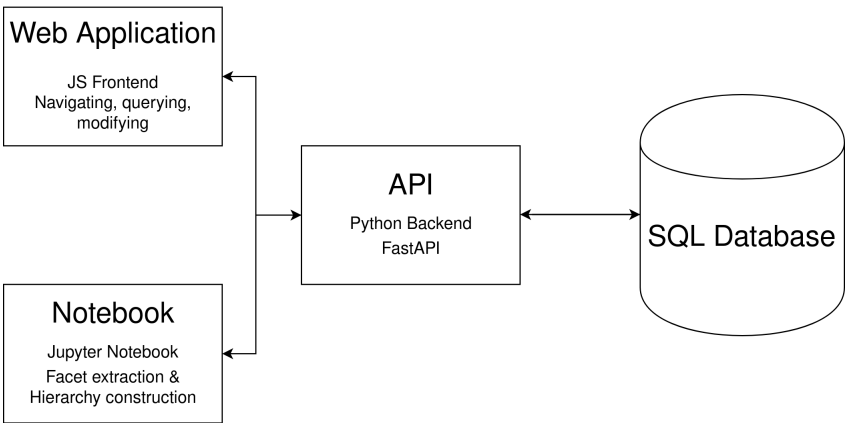


Figure 4.1: A simple data flow diagram describing the exchange of data between the components of the prototype.

We split the prototype into multiple components. First, a database provides long term storage for terms, spans and relations. Next, an API as backend provides means to interact with the stored data. Both a web application as frontend and a jupyter notebook can interact with said API. While the notebook provides easily accessible tools for developers or more experienced users allowing the importing of different data sources and the means to perform facet term extraction and hierarchy construction on a larger scale, the web application provides an interface for users to interact with the materialized faceted taxonomy. This interaction includes navigation, constructing complex queries, and applying basic modifications to the taxonomy. Figure 4.1 provides a schematic overview of the data flow.

We implemented the prototype to work exclusively with textual data. The focus of this thesis is still the application of the model to the field of digital humanities, thus even though the model offers more possibilities, we consider the work with textual data the main focus. Accordingly, the prototype uses character spans as the only kind of span which serve as leaf nodes, and the intersections between them as the sole relation.

4.1.1 Notebook

For the facet extraction and hierarchy construction, the notebook is used to access the API. Basically, the procedure presented in Section 3.3 is applied. The text corpus that should be explored is first split into meaningful segments. For most cases, we used entire documents, paragraphs and sentences as granularities. First, we create a term for each used granularity and append them to the root node. For each unique document, paragraph or sentence, a term is created and subsumed by its respective designated parent. Thus, for example, a sentence is subsumed by the sentence term. Now, for each segment, a span is constructed that contains its start character, the end character, and a reference to the document of its origin. This span becomes the child of the term that was created for the same segment, or to the term that describes an identical segment, should they not be unique. During this segmentation, we also construct spans for the meta data for each document. Each of these spans also have a range that covers the entire document between their start character and end character. For each unique meta key a child of root is created and for each meta value a child representing the respective key. After iterating all documents in the underlying corpus and constructing their spans, all intersections between all spans are calculated. Using these intersections, the objects of the faceted taxonomy are generated. Each span is referenced by an object,

that stores which other spans intersect the referenced span. An example for a layout of such a faceted taxonomy is displayed in Figure 4.2. Note that in the illustration the set of spans and the set of relation objects is only presented schematically, so the displayed relations and references between the two sets is just for illustration purposes. Notice also that terms that are meta values subsume multiple spans, while each document and paragraph only subsume one span. When an author has multiple works or multiple works were released in the same year, these meta values would subsume multiple spans. Documents, in some cases, can be considered unique, in which case each only subsumes one span. Due to the fact that sentences may very well be repeated in different works, a sentence term can subsume many spans. This taxonomy serves as a base for the system and is usable in its state. Although, for the answering of research questions that extend meta data, domain specific annotations are necessary.

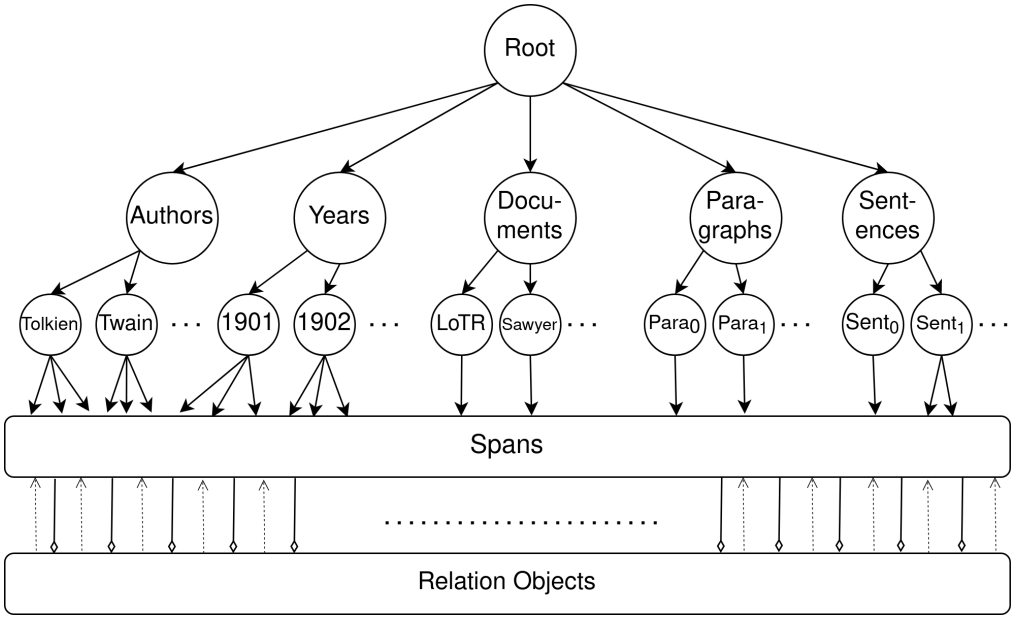


Figure 4.2: An example for a taxonomy constructed by the facet extraction and hierarchy creation of our model.

For our prototype, we decided to only implement the discovery of possible annotations by using regular expressions (regex). Using regex, it is usually easier to achieve a high precision with a low recall, thus it is easy to ensure that the returned results are relevant, but many relevant results may not be returned. We deemed this preferable for a prototype to at least work with mostly relevant data, even though many false negatives may exist. Any elaborate machine learning algorithm may be employed here, but none is

implemented thus far since that would be beyond the scope of this work. Regular expressions provide a sufficient amount of results to use as domain specific annotations.

Also, the construction of the additional taxonomy elements is conducted manually. While it is possible to use some knowledge base to generate terms and subsumptions, the resulting taxonomies were bloated and often not usable. To not be dependant on the quality of some knowledge base, and since specific research questions may not need too large a taxonomy, we decided to only create terms and subsumptions manually. First, one would add a term as a child of the root. Now it is already possible that this one term is sufficient, or maybe a deeper hierarchy has to be constructed. For each added term, the wikipedia page ID for the closest found match is identified. It can be later used to display some basic information and provides thumbnails in the taxonomy. The page ID for each term can be manually modified. When the desired depth of terms is reached, one can build a regular expression to describe occurrences of a current leaf node. All occurrences of the regular expression in any text are used to construct spans which are added as children of the current term. Then, all intersections of the newly added spans with any already existing spans are

Algorithm 4.1: findIntersections

```
1  findIntersections(S, S_A, R)
   Input  : Set of all spans S, set S_A ⊂ S of spans that were newly
            added as annotations, the set of all relations R
   Output: A modified set of relations R
2  begin
3      for each a in S_A do
4          r_a := element r in R where R(a) == a
5          for each b in S do
6              r_b := element r in R whereR(b) == b
7              if a_R == b_R then
8                  if ((b_E ≥ a_S & b_S ≤ a_E) | (a_E ≥ b_S & a_S ≤ b_E)) then
9                      r_a.addRelation(b)
10                     r_b.addRelation(a)
11         end
12     end
13 end
```

Figure 4.3: A simple depiction of the algorithm to find intersections between some newly added spans and the entire set of spans.

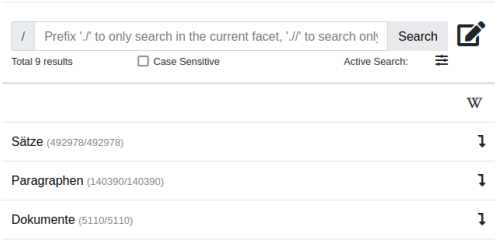
calculated and relation objects constructed. The newly added terms and spans are now fully integrated components of the materialized faceted taxonomy. Figure 4.3 shows a simple algorithm used to calculate all the intersections between some newly added spans and the entire set of spans. Note here that due to the bidirectionality of intersections, relations are added to both spans in one iteration.

This concludes the description of the generation of a faceted taxonomy. Since the notebook communicates directly with the API and thus with the underlying database, requests may be posed at any time and any extension regarding facet term extraction, hierarchy construction and annotation discovery can be implemented here.

4.1.2 Web Application

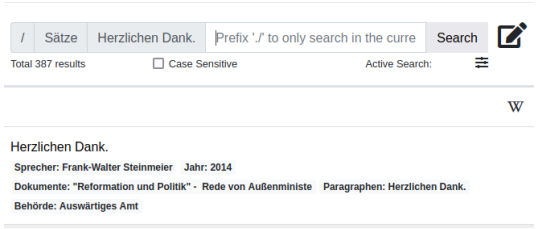
After generating the basic taxonomy, the web application is used to navigate, modify and conduct queries. Several issues arise regarding how to implement opportunities for interaction that fulfill all requirements of the underlying model, especially for how to construct complex queries.

The faceted taxonomy itself is presented in a temporal layout[27]. So, when navigating the taxonomy, the previously displayed terms are fully replaced by new ones. A user starts navigation with the focus on the root node, thus all immediate descendants of the root node are visible. Terms are displayed as a list, accompanied by their total count of children and the count of children remaining with some query applied to it. If no query is applied, both values are the same. Each non-leaf node also has an interaction element that allows to change the focus to that term, thus zooming into that term. Upon reaching leaf nodes, or spans, one naturally can not zoom in further. Since we use character spans in the prototype, they can easily be displayed by simply showing the text they reference. A history of the steps taken deeper into the hierarchy is shown which can be used to navigate back to previously used foci. As an navigational aid and for simply putting spans into context, all intersections each span has are displayed. These can be used to navigate to the respective intersected term. An example navigation is shown in Figure 4.4. If a wikipedia page ID for a focused term is defined, a short excerpt of the article can be displayed. A single taxonomy itself can be filtered already by using a search bar which accepts regular expressions. Three different modes for searching are available. In the default mode, all terms and spans that match the search query remain in the taxonomy.



(a) The first elements of the initial state of a taxonomy.

(b) The state of the taxonomy after changing the focus to the "Sätze" facet.



(c) The first element in the list after zooming into some sentence, displaying spans. Intersections of a span with other segments are displayed.

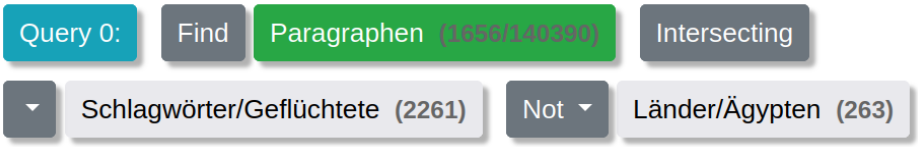
Figure 4.4: A sample navigation from the initial state of the taxonomy to the spans. The history of the navigation is displayed next to a search bar to return to a previous focus.

That means, if a regular term matches the query, all its children remain in the taxonomy and even if a term does not match the query, if one of its children does, it remains. For more search modes, we used prefixes inspired by XPath[36]. If a search is prefixed by ./, the query is only applied to currently visible terms and does not affect children at all. If prefixed by .//, the query is only applied to leaf nodes and ignores regular terms. While this simple search already allows to filter the taxonomy, it misses the ability to formulate complex queries using the query language defined in the previous chapter.

To let the user formulate complex queries combining multiple terms, we decided on using a setup that utilizes the same taxonomy multiple times, once for each query term. So, each query term, or set of query terms, is selected by navigating the temporal taxonomy. At the start of a session, a user is presented with the basic layout for a query. This includes a changeable name for the query, the `FIND` keyword, the root facet as query element, and the `INTERSECTING` keyword. Since we only implemented the intersection relation and the find operation, the keywords are static and shown for the

sake of displaying a complete query. The query element, that is currently the root element, is the target facet, thus the taxonomy that will be used for displaying the results. So the actual term that one wants to examine is selected by navigating the taxonomy. The current focus of the taxonomy is displayed in the query element. Now, any number of query terms can be added to the query. Each query term consists of two things; an operator and a focus in a taxonomy. A user can choose between the operators `AND`, `NOT` and `OR`, while the first additional query term has only the options `NOT` and no operator at all. In a displayed taxonomy, terms can be selected to be added to the focus of the current query term. When multiple of the available zoom-in points are selected, they are chained with `OR` operators. One aspect of faceted navigation is the condition that each query produces a non-empty set of results, since available features are updated after adding a query term[34]. So optimally, the current query in our system would be executed after adding a new term and the taxonomy for selecting the new query term should display only those element which selection would lead to a non-empty result. Due to expensive calculations for executing a query, we decided to allow empty sets as results. Only when selecting the target element the current query is executed. While not optimal, frequent executing of the query and the wait as a consequence thereof impaired the workflow heavily.

(a) The initial state of each query. The root facet of the taxonomy is the focus and selected.



(b) The query after multiple elements were added to it to filter the results.

(c) A new query is constructed using the results of the previous query to build a nested query.

Figure 4.5: Queries of different complexities for filtering results.

A problem with this method is the construction of nested queries. To solve this, a user can create any number of simple queries and use the results of a query as a query term of another one. Since each query returns a set of relations, and as a result a set of spans, they can easily be reused. In the taxonomy to choose the focus for any query term, all other queries are simply available for selection and can then, after selection, be treated as any query term and are provided with an operator. See Figure 4.5 for examples of queries of various complexities. This way of constructing nested queries brings additional benefits. Queries are constructed iteratively, and one can always easily return to previous results. Also, the same query can be a used as a query term in any number of other queries, which allows for simple comparison between multiple questions regarding similar topics.

Algorithm 4.2: executeQuery

```
1  executeQuery(t, Q)
   Input  : A target facet t and a list of query terms Q. Each q ∈ Q is a
            pair of an operator q_o and the set of spans q_s that is
            described by the query terms or the query.
   Output: A set of spans tSpans
2  begin
3      tSpans = L(T)
4      for each q in Q do
5          qSpans := set()
6          for each span in q_s do
7              if span.intersects(L(t)) then
8                  qSpans.add(span)
9              if q_o == OR then
10                 tSpans = tSpans + qSpans
11             else if q_o == NOT then
12                 tSpans = tSpans − qSpans
13             else
14                 tSpans = intersection(tSpans, qSpans)
15         end
16     end
17     return tSpans
18 end
```

Figure 4.6: A simple depiction of the algorithm to execute a query. For simplicity, we use the spans that are in the deep extension of each query term to include both regular terms and entire queries in one step.

A simple depiction of the algorithm to execute a query is shown in Figure 4.6. First, all spans of the target facet are determined. For each query parameter, the spans described by the query terms or the full query are detected. It is then determined, which of the query spans intersect the full set of target spans. The resulting query spans are then used to filter or extend a set of target spans, depending on the used operator. Note that the arithmetic operations ins lines 10 and 14 describe set operations.

For each query term, a set of options can be modified by the user. The set of options can be seen in Figure 4.7. First, the sorting mode can be set. Sorting can be seen as the ranking of the results, with the displayed elements in the taxonomy sorted according to the selected ranking. All sorting modes can be applied in ascending or descending order. The first three sorting modes are trivial. Elements can be sorted according to their absolute count, relative count, and alphabetical. The absolute count describes the amount of children any term has, and the relative count of a term in this instance measures

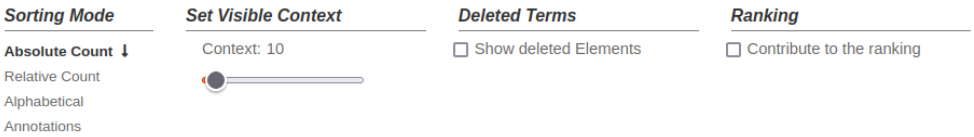
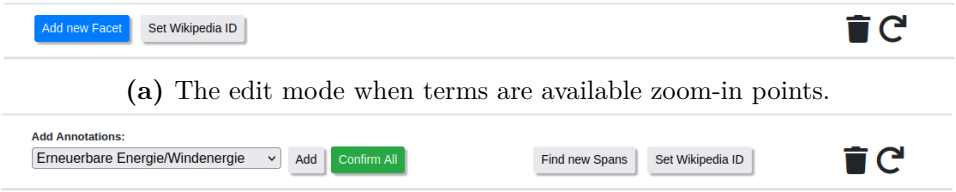


Figure 4.7: The set of options available at all times.

the amount of children of a term remaining after a query divided by its total amount of children. The fourth sorting mode, annotations, sorts terms according to their score $Sc_{q,i}(t)$ for each term t, with i being the intersection relation. Thus, the rank of term t is described by the amount of spans in the deep extension of the applied query q that intersect spans in the deep extension of the the term t. Another option for user to apply is the visible context. Often times, character spans depend heavily on the surrounding context. When viewing annotation spans for example, the user would only see the exact text the annotation references. Often, this is only a single word. Thus the user is given the option to display any number of characters before and after each span, to see each span in its surrounding context. During the modification of the taxonomy, the user is given the opportunity to delete certain terms and spans. These are not fully deleted, but rather hidden and removed from consideration. A user can select the option to show deleted terms, after which those hidden elements are visible again and fully viable for further use. As a last option, users can select if the current query element should contribute to the ranking. Ranking in this case means exclusively

the ranking using the number of annotations. It is possible that a user is interested in the contribution of a specific query element to the resulting target taxonomy, in which case the contribution of non-relevant elements can be simply hidden.

We call navigating through the taxonomy and the formulating of queries *exploring*. The explore mode is the default setting of the system with the goal to find some information through exploratory search. To achieve the possibility for a user to perform taxonomy modifications, we introduce an additional *edit* mode. Two different edit modes are available to a user, depending on which term is currently focused in the taxonomy, since the operations for regular terms and for spans differ. The two modes can be seen in Figure 4.8. If the visible zoom points in a taxonomy are terms with their own children, the available operations for a user are to add a term as a child of the currently focused term, or to remove any number of visible terms t_x from the taxonomy.



(a) The edit mode when terms are available zoom-in points.

(b) The edit mode when spans are displayed.

Figure 4.8: The two different states of the edit mode.

When a term is added as a child, the wikipedia page id of the closest match with respect to the name of the term is determined and a thumbnail and description text created. Should the id be wrong, it can be manually set by the user. If deleted terms are set to be visible in the options, they can be selected and restored to the taxonomy to be again regularly usable. If the visible zoom-in points t_x are spans, other operations are available in the edit mode. Since spans are leaf nodes by definition, no new facet term can be created. But now the user has the opportunity to create new annotations in two ways. As the first way, the user can perform an operation to search all underlying documents for a regular expression. For all matches, spans are created and added as children of the currently focused term. This is the same operation that is usable in the notebook and the algorithm depicted in 4.3 is applied as well. The second way is a purely manual operation in which single spans can be added or removed. Since character spans are used, the user sees the actual referenced text for each span. Next to the list of intersections underneath the spans, all existing annotations are highlighted in the text

itself. See Figure 4.9 for an example. An existing annotation can simply be selected and then removed, should it be a false positive. Deleted annotations can also later be restored. If the user detects a false negative, thus a missing annotation, the respective range of characters needs to be highlighted by the user. In the available operations for the edit mode, the user can select of all available terms whose spans describe annotations. So, terms describing regular units of text or metadata, like sentences or years, are not available here. While an argument could be made to include at least some of those terms to counteract potential faulty segmentation, we decided to only keep annotations to reduce clutter since the visualization of available terms is achieved in a simple drop-down menu. After highlighting a text segment and selecting the desired term, the highlighted text can be added as an annotation candidate. Any number of annotation candidates can be added by the user in any of the currently visible spans, as children of any of the available annotation terms. All candidates can now be confirmed to be added as regular spans to the taxonomy, as children of their respective term. Then, all intersections of the newly added spans are determined using the algorithm in 4.6.

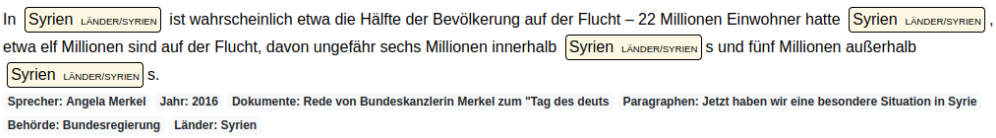


Figure 4.9: Annotations highlighted in a span. Annotations can be selected to delete them or to change to focus to its respective term.

With these presented features we implemented all relevant aspects of our theoretical approach, namely the faceted taxonomy generation, the modification of the taxonomy, the query refinement using the custom query language, and the ranking of the results according to the amount of relations using a notebook and a web application, each specialized for a group of tasks.

4.2 User Study

To evaluate our implemented prototype, we conducted a small user study with 10 participants. All participants are either from the fields of humanities or computer science. With the study we wanted to discover how potential users approach problems, if they are able to solve sample tasks covering the entire range of the functionalities provided by the prototype, how much time the users require to solve these tasks, and how they rate the interface and

interaction using different metrics.

As an underlying dataset, we used a corpus consisting of transcripts of ≈5000 German political speeches[3][4]. Due to that fact, all participants chosen for the study were fluent in German. We chose this dataset for the reason that all participants show at least some interest in politics, and interest in the provided data is desirable for evaluation[17]. These speeches stem from different speakers of different branches of the German government, so they cover the meta data branch, speaker, and year. For the generation of the faceted taxonomy, all speeches were split into segments using different granularities, namely the entire speeches, paragraphs and sentences. In addition to the segmented units of text, we prepare a small set of annotations for the user to interact with. We created facets which describe spans from the fields of renewable energies, mentions of several countries, and describing different terms related to refugees. We presented the users with the same five following tasks, of which four were of exploratory nature and one required modification of the taxonomy:

1. How many speeches has speaker A between the years Y_1 and Y_2?
2. How many paragraphs contain annotations from facet F_1 or F_2, and from facet F_3?
3. Which speaker wrote most of the paragraphs from the previous question?
4. Which sentence, used exclusively by author A, is the most common?
5. Manually annotate two spans for facet F_4. Candidate spans are in direct vicinity of spans of F_5

We provided the participants with information about what what the system is able to do and what elements for interaction exist, but no information about how to precisely use the provided interface to complete the posed tasks.

All participants in the study were able to complete all five tasks. After the completion of each task, we recorded the required time a user needed and asked the participant to estimate how difficult it was to solve the task with the provided interface[40], with 1 being very easy and 5 being very difficult. The table in Figure 4.10 shows the average time in seconds a participant needed to complete the tasks and the average estimated difficulty. Figure 4.11 displays a boxplot showing the ranges of the required durations participants spent solving each task. We first note that while the participants required on average 167 seconds to complete the first task, the fourth exploratory task required only

137 seconds on average, despite its heavy increase in complexity. This leads to the consideration that a learning process takes place for exploratory tasks and the understanding of the interface increases with each use. For reference, an experienced user required around 30 seconds for each explorative task and 60 seconds for task 5.

Task	avg. Time	avg difficulty
1	167	3.3
2	124	1.7
3	133	2.3
4	137	2.5
5	235	4.3

Figure 4.10: A table presenting the average time each user required for each task and the average estimated difficulty.

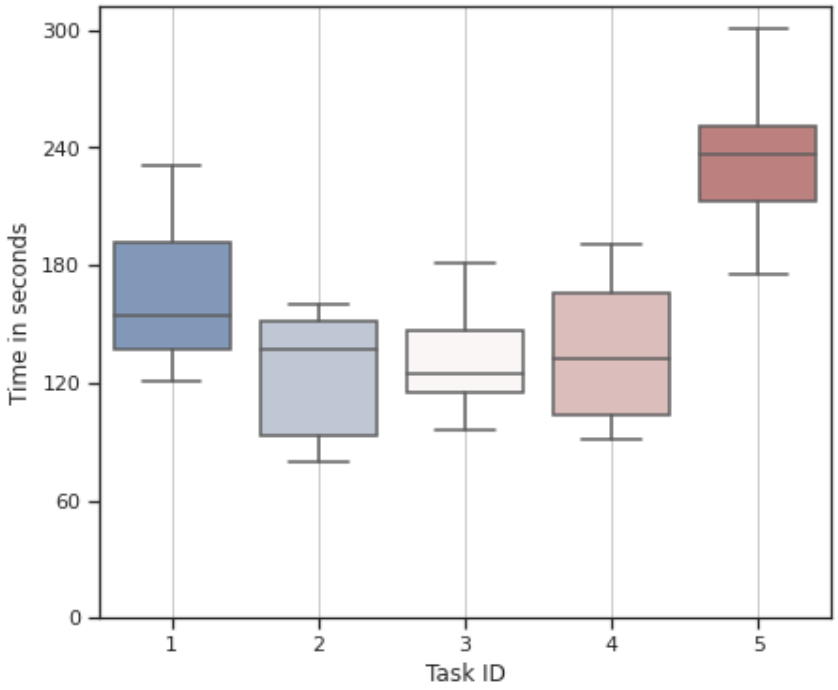


Figure 4.11: Annotations highlighted in a span. Annotations can be selected to delete them or to change to focus to its respective term.

This assumption is supported by the estimated difficulty. While the participants rated the first task on average with a difficulty of 3.2, which is

rather high, the second task has only an average estimated difficulty of 1.7. So, despite still requiring a relatively large amount of time of $\approx$ 2 minutes for the second task, the participants deemed it easy to solve. The other exploratory tasks were of higher complexity than the first two, which explains the slightly longer required time and higher estimated difficulty. Task 5, requiring the detection of missing annotations and their manual addition, proved to be the most difficult task, which is shown by requiring the largest amount of time, 235 seconds on average, and being rated by far the most difficult task to complete with an estimated difficulty of 4.3 on average. In Figure 4.11, one can also see a large spread between the fastest and slowest participant. Solving exploratory tasks can be assumed to not prepare users for the task of performing taxonomy modifications. It is unfortunate that we did not prepare a second task for taxonomy modifications because the possibility exists that a similar learning effect like for exploratory tasks would be introduced. All in all we can make the assumption that a learning effect can be observed after performing multiple exploratory tasks, while modifying the taxonomy in a complex way proved to be a very difficult task to complete successfully without any introduction.

As another measure, we used the User Experience Questionnaire(UEQ)[18] to evaluate the interaction with the interface itself. We limited ourselves to the shorter version of the questionnaire UEQ-S[29] which participants in the study filled out after completing all tasks. The short questionnaire consists of four pragmatic and four hedonic dimensions of quality which a participant should evaluate with values from 1 to 7, with 1 being the most negative and 7 the most positive estimation. A boxplot showing the results of the questionnaire and the ranges of all 8 dimensions is shown in Figure 4.12. One first notices that the system on average feels complicated and confusing for the users. While there is a large spread with some participants rating the system rather easy than complicated, even after completing all tasks one participant rated the interface both very complicated and very confusing. A frequently voiced point of critique and source of confusion was that the construction of a query starts with selecting a target facet to which a user has to return after adding query terms to see the actual results. On the other hand, the system is deemed to be rather efficient. Thus, the proposed system can be assumed to be a complicated tool which is efficient in solving the problems it was designed for. Users also rate the system to be rather inventive and leading edge, which leads to the assumption that the tool provides interaction techniques which are new to users.

The user study gave valuable insight to evaluate how well the prototypical

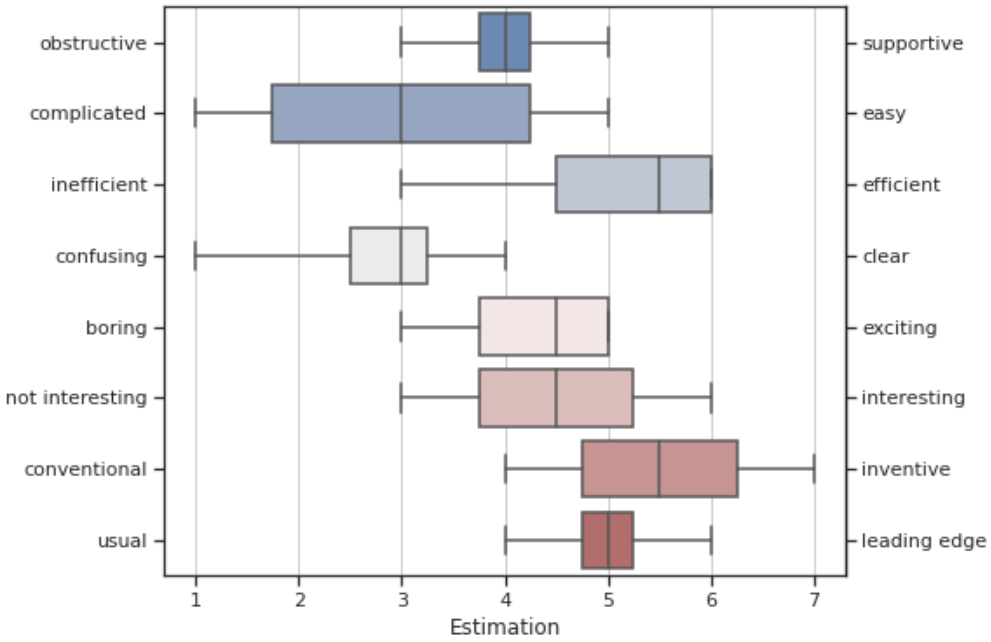


Figure 4.12: Annotations highlighted in a span. Annotations can be selected to delete them or to change to focus to its respective term.

implementation of the system can be used to answer simple research questions using the proposed model. Since users were given no instructions for how to use the given interface and all participants managed to solve all tasks, we consider the study and the prototype to be successful. But due to the fact that the time required to answer the research questions was still rather high and the user experience questionnaire showed that the system was deemed fairly confusing and complicated, the interface itself has room for improvement to make interaction more intuitive.

Chapter 5

Conclusion

In this thesis, we introduced a model for a faceted search system that allows the exploration of different relations between concepts. We specified the model further to enable its application to large text corpora to answer research questions in the digital humanities. We proposed a query language to construct arbitrarily complex requests to represent such research questions. Thus, a framework for iteratively constructing a research environment to answer tailored research questions from the field of the digital humanities was introduced.

We presented a prototypical implementation of the model for the exploration of texts. Other than allowing a user to navigate a text corpora and construct nested queries using one relation between concepts, users have the ability to manually and semi-automatically add or remove semantic annotations to construct the research environment themselves and also improve the quality of the search results. We evaluated the prototype in a small user study using different metrics, namely the required time to complete a task, the estimated difficulty of solving the task, and an assessment of the interface using the user experience questionnaire.

The conducted user study showed that the prototype has a steep learning curve and several flaws, but is still an effective tool for solving tasks using our proposed model. We believe that our contribution can be effectively used for future work. First, the interface of our prototype can be improved to increase the intuitivity and general user satisfaction. In addition, our implementation only covered a small part of the proposed model. Future works could implement multiple different relations between concepts and extend the query language to be able to pose more complex research questions.

Bibliography

[1] Taro Aso, Toshiyuki Amagasa, and Hiroyuki Kitagawa. Relation-oriented faceted search method for knowledge bases. In *Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications & Services*, page 192–199, New York, NY, USA, 2020. Association for Computing Machinery.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. Modern information retrieval, 1999.

[3] Adrien Barbaresi. A corpus of German political speeches from the 21st century. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 792–797, Paris, France, 2018. European Language Resources Association (ELRA).

[4] Adrien Barbaresi. German political speeches corpus. June 2019. Available from `https://doi.org/10.5281/zenodo.3611246`, version v4.2019.

[5] Marcia J. Bates. Subject access in online catalogs: A design model. *Journal of the American Society for Information Science*, 37(6):357–376, 1986.

[6] Douglas Biber, Susan Conrad, and Randi Reppen. Corpus linguistics - investigating language structure and use. In *Cambridge approaches to linguistics*, 1998.

[7] Donald D. Chamberlin and Raymond F. Boyce. *SEQUEL: A Structured English Query Language*. SIGFIDET '74. Association for Computing Machinery, New York, NY, USA, 1974.

[8] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. pages 318–329, 1992.

[9] Yuqi Du, Jin Zheng, Wang Yang, Huang Chen, and Li Ping. Text segmentation model based on graph convolution network. *Journal of Computer Applications*, 2022.

[10] Anette Frank, Thomas Bögel, Oliver Hellwig, and Nils Reiter. Semantic annotation for the digital humanities – using markov logic networks for annotation consistency control. 2011.

[11] Martin Fricke. Faceted classification: Orthogonal facets and graphs of foci? *Knowledge Organization*, 38:491–502, 01 2011.

[12] ExplosionAI GmbH. Prodigy. `https://prodi.gy/`, accessed 04.2022.

[13] Marti A. Hearst. Multi-paragraph segmentation of expository text. *arXiv*, 1994.

[14] Inc Heartex. Labelstudio. `https://labelstud.io/`, accessed 04.2022.

[15] Jan Horstmann. *Undogmatic Literary Annotation with CATMA*, pages 157–176. 10 2020.

[16] Christin Katharina Kreutz, Peter Boesten, Alex Witry, and Ralf Schenkel. Facetsearch: A faceted information search and exploration prototype. In *LWDA*, 2018.

[17] Bill Kules and Robert Capra. Creating exploratory tasks for a faceted search interface. 01 2008.

[18] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. *USAB 2008*, 5298:63–76, 11 2008.

[19] Mohammed Najah Mahdi, Abdul Rahim Ahmad, Hayder Natiq, Mohammed Ahmed Subhi, and Qais Saif Qassim. Comprehensive review and future research directions on dynamic faceted search. *Applied Sciences*, 11(17), 2021.

[20] Gary Marchionini. From finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.

[21] Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. doccano: Text annotation tool for human, 2018. Software available from https://github.com/doccano/doccano.

[22] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[23] The Editors of Encyclopaedia. Encyclopedia Britannica, "facet". `https://www.britannica.com/art/facet`. Accessed 19 April 2022.

[24] S.R. Ranganathan. *Colon Classification*. Madras Library Association, 1933.

[25] J. Reitz. *Dictionary for Library and Information Science*. Libraries Unlimited, 2004.

[26] Giovanni Maria Sacco and Yannis Tzitzikas. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*. 2009.

[27] m.c Schraefel, Daniel Smith, Alistair Russel, Alisdair Owens, Craig Harris, and Max Wilson. The mspace classical music explorer: Improving access to classical music for real people. 01 2005.

[28] M.C. Schraefel, Max L. Wilson, Alistair Russell, and Daniel Alexander Smith. mspace: improving information access to multimedia domains with multimodal exploratory search. *Communications of the ACM*, 49(4):47–49, April 2006.

[29] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(6):103, 2017.

[30] Paul Smart, Robert Clowes, and Richard Heersmink. Minds online: The interface between web science, cognitive science and the philosophy of mind. *Foundations and Trends® in Web Science*, 6, 09 2017.

[31] tagtog Sp. z o.o. Tagtog. `https://www.tagtog.net/`, accessed 04.2022.

[32] Daniel Tunkelang. *Faceted Search*, volume 1. 01 2009. *Synthesis Lectures on Information Concepts, Retrieval, and Services*.

[33] Yannis Tzitzikas, Anastasia Analyti, Nicolas Spyratos, and Panos Constantopoulos. An algebraic approach for specifying compound terms in faceted taxonomies. 07 2003.

[34] Pawan Vora. Chapter 5 - navigation. In *Web Application Design Patterns*, Interactive Technologies, pages 111–142. Morgan Kaufmann, Boston, 2009.

[35] W3C. SPARQL 1.1, W3C Recommendation 21 March 2013. `https://www.w3.org/TR/sparql11-overview/`, accessed 04.2022.

[36] W3C. XML Path Language (XPath) 3.1, W3C Recommendation 21 March 2017. `https://www.w3.org/TR/2017/REC-xpath-31-20170321/`, accessed 04.2022.

[37] Bifan Wei, Jun Liu, Qinghua Zheng, Wei Zhang, Xiaoyu Fu, and Boqin Feng. A survey of faceted search. *Journal of Web Engineering*, 12(1-2):041–064, 2013.

[38] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*, volume 1. 2009.

[39] Max Wilson and m.c Schraefel. The importance of conveying inter-facet relationships for making sense of unfamiliar domains. 01 2009.

[40] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. page 401, 2003.