View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Technische Universität Dresden: Qucosa

Attribute Exploration on the Web

Robert Jäschke[1] and Sebastian Rudolph[2]

[1] L3S Research Center, Hannover, Germany, jaeschke@l3s.de
[2] AIFB, Karlsruhe Institute of Technology, Germany, rudolph@kit.edu

Abstract We propose an approach for supporting attribute exploration by web information retrieval, in particular by posing appropriate queries to search engines, crowd sourcing systems, and the linked open data cloud. We discuss underlying general assumptions for this to work and the degree to which these can be taken for granted.

Keywords: Formal Concept Analysis, Attribute Exploration, Web Information Retrieval, Linked Open Data

1 Introduction

In Formal Concept Analysis [6], objects are described by their attributes. The idea of *attribute exploration* is to determine a minimal set of implicational dependencies between attributes that hold for all objects of a certain domain. To this end, one starts with a partially defined formal context, i.e., a selection of objects and their attributes. From this "sample", hypothetical implications are computed and presented to an expert who either confirms their universal validity or refutes it by providing an object as counterexample. Normally, obtaining these counterexamples is a laborious task, depending on the domain of the objects. For example, the domain of one of the earliest applications of attribute exploration [18] was lattice theory and confirming a hypothetical implication between properties of lattices meant to find an appropriate proof whereas refuting it meant to provide a specific lattice violating the hypothesis. Attribute exploration has also been used for creating and completing ontologies based on description logics [16,3], or for building access control models [12].

Until now, attribute exploration was mostly driven by an expert who has to check the implications. Typically, each implication is presented to the expert as a question in the form "Is it true that all objects that have the attribute(s) $l_1, l_2, \ldots$ also have the attribute(s) $r_1, r_2, \ldots$?" However, the knowledge we are seeking is often already available on the web, e.g., as facts in Wikipedia, or it can be obtained by leveraging massively collaborative Web 2.0 platforms. While we acknowledge the role of the expert and do not want to replace him or her, we aim to better support the expert in employing the knowledge found in the World Wide Web by automatically posing appropriate queries to web search engines in order to retrieve potential counterexamples. Thereby, we assume that the expert is not omniscient but may benefit from external knowledge, at least

for the answer of some questions. Our approach has the potential to speed up the attribute exploration process and, by providing context for all questions, to help the expert to avoid errors due to existing counterexamples unknown to him or her.

In this paper we want to outline the chances and limits of supporting attribute exploration by on-the-fly retrieval of information from the web in various ways. While this endeavor is itself exploratory and only preliminary, we hope that our considerations will pave the way toward exploration methodologies that make intelligent use of the abundance of available web data. The paper is organized as follows: In Section 2 we review related work. After a brief introduction to attribute exploration in Section 3, we describe three specific approaches to tackle attribute exploration using the web in Section 4. We present a first implementation in Section 5 and conclude the paper in Section 6.

2 Related Work

Koester's FooCA system [11], retrieves results from major web search engines and allows users to analyze and visualize them using FCA. Therefore, FooCA uses the title, description, and the URL of web pages to build a formal context. The system allows users to modify queries until they fit their information need. In contrast to our work, attribute exploration is not considered in FooCA and queries are built by the user instead of the system itself. Furthermore, FooCA is considering the web pages themselves as objects while our approach considers objects *within* web pages or draws conclusions about empty result sets.

Rudolph [16] proposed to create description logic [2] knowledge bases by means of attribute exploration coupled with automated reasoning systems. Baader et al. [3] show how an extension of attribute exploration, that is capable of handling partial information, can be employed for completing such knowledge bases. Since description logic is underlying the web ontology language OWL [10] in which DBPedia [1] is represented, their approach could be used to check the completeness of DBPedia and therefore also Wikipedia.

The idea to automatically query web search engines to check or extend a knowledge base has been applied in the area of ontology learning, where Hearst patterns [9] are used to learn relationships between concepts [5].

3 Formal Concept Analysis and Attribute Exploration

In the following we briefly introduce the important notions in FCA by means of a small example. Imagine a user that is interested in European politics and therefore investigates political, military, and economic alliances in Europe by considering the membership of European countries in the NATO and the EU and their participation in the Euro and the Schengen Agreement.[3] Collecting

[3] http://ec.europa.eu/dgs/home-affairs/what-we-do/policies/borders-and-visas/index_en.htm

the information for all European countries is a tedious task and since the user is anyway only interested in the implications that hold between the four attributes NATO, EU, Euro, and Schengen, he decides to apply attribute exploration to find these implications.

3.1 Formal Contexts and Formal Concepts

Using the notation from [6], we are considering *formal contexts* $\mathbb{K} := (G, M, I)$ where G is a set of objects, M a set of attributes, and I a binary relation between G and M, i.e., $I \subseteq G \times M$. We read $(g, m) \in I$ as "object g has attribute m". For $A \subseteq G$, let $A' := \{m \in M \mid \forall g \in A : (g, m) \in I\}$, and dually, for $B \subseteq M$, let $B' := \{g \in G \mid \forall m \in B : (g, m) \in I\}$. For an object $g \in G$ we often write g' instead of $\{g\}'$.

Table 1 shows the formal context with which our user starts. It contains as objects the three countries Czech Republic, Norway, and Germany and as attributes the four alliances/agreements NATO, EU, Euro, and Schengen. The

Table 1. A formal context about properties of European countries.

	NATO	EU	Euro	Schengen
Czech Republic	×	×		×
Norway	×			×
Germany	×	×	×	×

information about the membership of the countries is taken from their corresponding pages in Wikipedia. We will use that context as a running example throughout this paper.

3.2 Implications Between Attributes

An *implication* between the attributes of a formal context is a pair of subsets L, R of M denoted by $L \to R$, in which L is called the *premise* and R the *conclusion*. For simplicity, we always assume $L \cap R = \emptyset$, i.e., we omit the attributes in the premise from the conclusion. An implication $L \to R$ *holds* in a formal context, if each object having all attributes from L also has all attributes from R. For instance, the implication $\{\text{EU}\} \to \{\text{NATO}\}$ holds in the context in Table 1.

3.3 Attribute Exploration

The goal of attribute exploration is to compute a set of implications between attributes that hold for all objects under consideration. In particular, if it is not feasible to explicitly list all objects of a formal context (e.g., because there are infinitely many of them), attribute exploration supports us in searching and specifying representative objects.

In an interactive process [6], implications between attributes are computed and shown to an expert who checks if they hold. A found implication $L \to R$ can either be accepted or refuted by a *counterexample*. A counterexample c is an object that has all attributes from L but there exists at least one attribute from R that c does not have. Formally, an object c refutes the implication $L \to R$, if $L \subseteq c'$ but $R \not\subseteq c'$ (i.e., there is at least one $m \in R$ with $m \notin c'$). Assuming a universe of objects, we denote the set of possible counterexamples for an implication $L \to R$ (i.e., all those objects refuting the implication $L \to R$) by $\mathcal{C}(L \to R)$. The algorithm in [6] ensures that a non-redundant and complete set of implications [7] is computed.

Starting the attribute exploration with the context in Table 1 yields the implication $\emptyset \to \{\text{NATO}, \text{Schengen}\}$ and thus the question "Is it true that all objects have the attributes NATO and Schengen?" A counterexample would be a country that is not a member of the NATO or does not participate in the Schengen Agreement.

4 Web-Based Attribute Exploration

How can we leverage the knowledge available in the web to infer implications between attributes? Returning to our example, in the simplest case the user could gather a list of countries in Europe (e.g., from Wikipedia[4]) and for each country look up the information on the web, or do the same for each of the attributes. On the one hand, this is a very tedious task that involves searching, visiting, and screening many web pages. On the other hand, it is not necessary to build this complete list in order to obtain implications between the attributes. As we have seen in Section 3.3, attribute exploration helps us to find a small set of objects whose attribute logic (i.e., the attribute implications jointly satisfied by them) is universally valid.

We are now investigating how the knowledge available on the web can be leveraged for attribute exploration. While we are focussing on web search engines (Section 4.2) that allow us to query a larger part of the web than any other technology, we want to show the wide range of sources available on the web by investigating three other possible options (Section 4.3): social question answering, crowdsourcing, and the linked open data cloud. We start with an overview on query strategies, since there are a few commonalities between these approaches on an abstract level.

4.1 Abstract Query Strategies

In the majority of the cases, the information that we can hope to draw from the web is *factual* (or *assertional*), i.e., it provides information about a singular instance (object) and its properties (attributes). Information about universally

[4] http://en.wikipedia.org/wiki/List_of_sovereign_states_and_dependent_territories_in_Europe

valid propositions (so-called *terminological* knowledge) is more rare and harder to retrieve.[5] However, implications are of terminological nature, hence it will be hard to draw conclusive evidence from the web for the validity of an implication. On the other hand, information about counterexamples is factual. Thus, the general strategy is to focus on the task of retrieving counterexamples from the web and, applying a sort of closed-world assumption, assume that an implication is valid if no such counterexamples can be found.

In general, information retrieval is performed via queries (be it queries posed toward a web search engine or a database or a human). Now, when considering an implication $L \to R$, which type of factual query helps us to find counterexamples?

From a logical viewpoint, an *instance query* is a formula $\varphi[x]$ with one free individual variable x, specified in a fixed logical formalism which is called *query language*. For a given domain (or logically speaking: interpretation) and query $\varphi[x]$, the associated *set of answers* $Ans(\varphi[x])$ is the set of domain elements d for which $\varphi[d]$ is true. We will now assume that each attribute $m \in M$ comes with an instance query $\varphi_m[x]$ for which the answer set is m'.

Under these assumptions, a query $q[x]$ for a counterexample of an implication $L \to R$ can be written as

$$q[x] := \bigwedge_{l \in L} \varphi_l[x] \ \wedge \bigvee_{r \in R} \neg\varphi_r[x] \tag{1}$$

Note that this requires the query language to support conjunction, but also both negation and disjunction. In the case that disjunction is not available, the above query can be split into a set of queries $Q = \{q_r[x] \mid r \in R\}$ with

$$q_r[x] := \bigwedge_{l \in L} \varphi_l[x] \ \wedge \neg\varphi_r[x], \tag{2}$$

then the set of counterexamples can be obtained by taking the union over all corresponding answer sets, i.e.,

$$Ans(q[x]) = \bigcup_{q_r \in Q} Ans(q_r[x]). \tag{3}$$

The logical viewpoint presented here helps in setting the stage and formulating a generic counterexample query (set), however, it assumes "perfect querying" which is not realistic in practice, particularly for information retrieval on the web. Web query answers may be unsound (the answer contains instances that do not qualify), incomplete (the answer does not contain instances that would qualify), usually they are both.[6]

[5] With the exception of knowledge bases formulated in expressive ontological languages like OWL.

[6] Note that systems performing retrieval tasks are evaluated via the measures precision and recall. Sound querying would correspond in 100% precision, complete querying in 100% recall, neither of which is normally achieved.

Thus it will normally be necessary to perform some sort of quality assurance (usually via scrutiny by a human) on the answers retrieved for a counterexample query, in order to make sure that they are indeed counterexamples. If the result contains a valid counterexample, we can add it to the context and continue the attribute exploration.

We will investigate this in more detail in the following sections, noting that the overall process is essentially the same in all cases:

a. Take an implication from attribute exploration and transform it into a query or a set of queries.
b. Pose the queries to the system and show the user the result.
c. Let the user decide if the implication holds or not.

The decision whether to employ queries of the form (1) or (2) not only depends on technical restrictions of the query language but also on the degree of human involvement in the querying task. Depending on whether the queries are posed to a large crowd of users or a single user is checking the results for counterexamples and possibly modifying the queries, we must take the capabilities of the involved human into account. While with the form (1), one query per implication is sufficient (and thus only one result set needs to be checked, respectively), the disjunction potentially causes many results to be returned at once and it is not so obvious for each result, which of the attributes from the conclusion it is lacking. This complicates the task of finding counterexamples. Queries of the form (2), on the other hand, are shorter and more simple and therefore their syntax and semantics are easier to understand by the user. Unfortunately, the results of the queries for one implication might overlap if there exist counterexamples that lack several of the attributes.

Now that it is clear that, theoretically, we can formulate web queries that support a user in checking the validity of an implication, the following research questions arise:

- What background knowledge about objects and attributes must we demand from the user? Is it enough to consider objects and attributes as strings or do we need synonyms, regular expressions, or even the corresponding page in Wikipedia? Which query language will we use?
- Is one query sufficient or do we need something like incremental query-refinement?
- Which results should be presented to the user? In which way?

We tackle these questions theoretically in the following two sub-sections and practically in Section 5.

4.2 Web Search Engines

Search engines constitute the most common entry point to the web and allow us to search over a corpus of documents that is by far the largest set of documents we can access. Their query languages typically support at least the logical

conjunction and disjunction of query terms. For efficiency reasons,[7] negation is only supported in queries that contain at least one positive term, to retrieve documents that satisfy ⟨list of positive terms⟩ *but not* ⟨list of negative terms⟩. Therefore, implications $\emptyset \to R$ with an empty premise must be treated with care, since the corresponding query would entirely consist of negative terms. Except for this case, the query language is therefore suitable to represent queries like the ones in Equations (1) and (2). In principle, any web search engine is useable but for simplicity our examples follow the search syntax of the most popular ones, Bing and Google.[8]

We focus on queries of the form (2), composed only of logical conjunction and negation. Their syntax is more simple and more common to users, since it composes (positive and negative) terms using conjunction only, which is the standard composition operation in most web search engines.[9] Furthermore, the queries are shorter. Both aspects allow the users to easier understand and modify the query. As a drawback, for each implication the users must check as many result sets as there are attributes in the conclusion. The prefix `+` in front of a term ensures that it is textually contained in the web page (possibly modulo morphological variants), the prefix `-` ensures that the term is not contained in the web page, a conjunction of terms is achieved by concatenating them by whitespace (␣). The query from Equation (2) can then be written as

$$q_r[x] := \texttt{+}l_1\texttt{␣+}l_2\texttt{␣}\ldots\texttt{␣+}l_{|L|}\texttt{␣-}r \tag{4}$$

As already mentioned, special care must be taken when $L = \emptyset$, i.e., the premise is empty, since queries containing exclusively negated terms are not supported by web search engines. This can be addressed by specifying the domain d of the objects (e.g, "Countries in Europe") and adding it as positive term to the query:

$$q_r[x] := \texttt{+}d\texttt{␣+}l_1\texttt{␣+}l_2\texttt{␣}\ldots\texttt{␣+}l_{|L|}\texttt{␣-}r \tag{5}$$

This is not a strong restriction, since often the objects we are interested in are instances of a common class. Furthermore, many search engines allow us to restrict the web site of the results by adding it to the query with the `site:` prefix. E.g., to restrict the result set to pages from the English Wikipedia, we can add the term `site:en.wikipedia.org` to a query. For our example context from Table 1 both restrictions are actually useful, since we can expect that the objects, i.e., European countries, we are interested in are well described in Wikipedia.

[7] A set of documents that contain the positive terms can be efficiently retrieved using an inverted index, likewise the documents that contain the negative terms can be retrieved. The first set is then filtered by the second.

[8] http://support.google.com/websearch/bin/answer.py?hl=en&answer=136861

[9] Note that this default is more and more weakened: when no or few documents could be found, or a term is very general, web search engines occasionally return documents that do not necessarily contain all positive terms. That is also the reason why we prefer to prefix all positive terms with `+`, which really ensures that they are contained in the document.

In the standard case, the result of a web search query is a set of web documents.[10] Thus, unless the objects of our domain of interest are indeed web documents (as in the setting described by Koester [11]), the retrieved documents will merely serve as an informative resource, hopefully describing objects of the wanted category.

Motivated by the preceding discussion, our approach is based on the following implicit assumptions:

1. Web documents often describe singular objects.
2. For every attribute there is a search term, the presence of which in such a web document can be regarded as an indicator for the described object having the attribute.
3. Likewise, the absence of this search term can be regarded as an indicator for the object *not* having the attribute.

All of these assumptions are arguable and their applicability varies from case to case. E.g., one problem with that approach is that web pages that mention that an object does *not* have a certain attribute are ignored by the corresponding negated term in the query and thus the counterexample can not been found. In Section 5.2, we will see instances of these problems we have to face in reality.

4.3 Other Paradigms

Besides web search engines, the number of systems that could possibly be employed to support attribute exploration is abundant: One could post questions to blogging or micro-blogging platforms and hope for answers, or even ask friends on social networks. In this section we first focus on two approaches that are particularly intended for posing queries to humans: *social question answering* and *crowdsourcing*. Then we take a look at an approach with considerably less human involvement on the one hand but a much more formal knowledge representation on the other hand: structured knowledge bases that are part of the so-called *linked open data cloud*.

Social Question Answering Systems. As one of the most explicit forms of social search, social question answering systems like Yahoo! Answers[11] or StackOverflow[12] allow users to post questions on the web that can be answered by other users. While Yahoo! Answers is very general and also contains questions like *How do hotels keep their towels so white?*, other systems are focused on certain topics, e.g., StackOverflow on programming (with questions like *How can I draw a flow chart using LaTeX?*). The systems provide mechanisms to

[10] A notable exception after the recent advent of http://schema.org/ are cases where the search engine additionally returns data about other entities, as e.g. in http://www.google.com/?q=Rudolf+Wille

[11] http://answers.yahoo.com/

[12] http://www.stackoverflow.com/

rate, comment, and accept answers such that users can more easily find correct answers or discuss alternative solutions.

Leveraging social question answering systems for attribute exploration is straightworward: instead of asking the expert, we could in turn post the question if an implication holds or not to the system and the expert could then conclude from the answers if the implication at hand holds or not. We could even ask the users to answer in a specific format such that we could automatically parse counterexamples and thereby completely automate the process. The rating mechanisms would allow us to judge the quality level of the answers and could support the expert in judging the result. An important drawback of the approach, however, is the high latency of answers (depending on the domain from some minutes to days; some questions are never answered) and the misuse of a social system in an unsocial way. On the other hand, one can retrieve profound answers, if an expert is willing to answer the question.

Crowdsourcing Systems. Something similar can be accomplished (and is technically much easier to implement) using crowdsourcing systems like Amazon Mechanical Turk.[13] They allow programmers to create small "human intelligence tasks" that are solved by a crowd of workers that get a small amount of money for solving these tasks (from a few cents to some dollars). Typical examples for such tasks are optical character recognition, information extraction, or image classification. In our scenario, we could again directly ask the workers if an implication holds and if not, which counterexample they can provide. Alternatively, we could break down each implication into questions of the form introduced in the previous section and ask the workers to answer these. The systems provide an application programming interface such that we can program user interfaces where the workers can directly enter counterexamples. In contrast to social question answering systems, crowdsourcing platforms are explicitly intended for this kind of human-machine interaction and therefore better suited as automatic source for attribute exploration. On the other hand, each answer costs money (though we can be lucky that FCA ensures that a minimal number of questions is asked) and the quality of the results often is not very good which requires to distribute each question to several workers and employ voting, or reputation mechanisms [15].

Linked Open Data Cloud. An increasing amount of knowledge is published online in the linked open data cloud in knowledge bases like YAGO [17] or DBPedia [4] where it is represented in triples of the form *(subject, predicate, object)* that express the fact that the *subject* is in relation *predicate* with the *object*, e.g., *(Germany, is member of, European Union)*. These triple sets – which could be conceived as multi-valued formal contexts – are represented using the Semantic Web standards RDF and OWL [10]. Knowledge bases in these formats can be queried using SPARQL [14] and thus be integrated into the attribute

[13] http://mturk.amazon.com/

exploration process in a similar way as described in Section 4.2 by formulating SPARQL queries instead of web search queries. In the sequel, we focus on DBPedia as one of these exemplary knowledge bases. DBPedia contains millions of triples that are automatically extracted from Wikipedia and thus constitutes a valuable source of information for various domains of interest. For each city in Wikipedia, for example, facts like name, country, geo-location, population, etc. are available.

In order to answer the queries posed during the exploration process, in the most simplest case the objects of the formal context should be entities that are represented by Wikipedia pages, preferably from a particular category, like *Countries in Europe*, or *Internet standards*.[14] For a multi-valued context, the attributes should map to properties of DBPedia, e.g., *total population*. The attribute values could then be categories, pages in Wikipedia, or arbitrary data (strings, numbers, dates, etc.). One would then apply conceptual scaling to derive a one-valued context amenable for attribute exploration. One exception to this is the particular case where the attributes directly map to categories of Wikipedia. This situation can be represented by a one-valued context in which the intent of an object is the set of categories associated to this object. During the attribute exploration process, an implication between attributes can be checked by querying the knowledge base with an appropriate SPARQL query. How such a query is built depends on the chosen attributes and scales. Due to space restrictions, we can not present the complete SPARQL queries that would correspond to Equations (1) or (2), but instead we give an example for the context in Table 1.

We require that all objects belong to the category *European countries*[15] and map the attributes to categories of Wikipedia in the following way: *NATO* $\mapsto$ *Member states of NATO*, and *EU* $\mapsto$ *Member states of the European Union*. The attribute *Euro* needs special care, since there exists no category for all countries that have the Euro as currency. Instead, we can use the category *Currency* and restrict it to the value *Euro*. Unfortunately, there exists no category for countries of the Schengen area and thus we can not map the attribute *Schengen*. This shows two limitations of our approach we will discuss at the end of this section. As an example, we now consider the query $q[x] = \neg\varphi_{\mathrm{NATO}}[x] \vee \neg\varphi_{\mathrm{EU}}[x]$ for the implication $\emptyset \to \{\mathrm{NATO}, \mathrm{EU}\}$ that comes up during the exploration of the context in Table 1. Using the mappings of attributes to categories presented above, we can map the query to the SPARQL query in Figure 1. The disjunction in our original query is represented by a `UNION` of two patterns that match countries of Europe that are not in the NATO or not in the EU, respectively. Since the current SPARQL standard does not support negation, we must employ the rather complicated `OPTIONAL { ?y ... FILTER (?country=?y) . }`

[14] The English Wikipedia contains more than 850,000 hierarchically organized categories (source: extracted category labels in DBPedia, http://downloads.dbpedia.org/3.8/en/category_labels_en.nt.bz2).

[15] This category has been changed to *Countries in Europe* in Wikipedia, but this change is not available in DBPedia, yet.

```
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX dcs: <http://purl.org/dc/terms/>
SELECT DISTINCT ?country WHERE {
  {
    ?country dcs:subject dbc:European_countries .
    OPTIONAL {
      ?y dcs:subject dbc:Member_states_of_NATO .
      FILTER (?country = ?y) .
    }
    FILTER (!BOUND(?y))
  }
UNION
  {
    ?country dcs:subject dbc:European_countries .
    OPTIONAL {
      ?y dcs:subject dbc:Member_states_of_the_European_Union .
      FILTER (?country = ?y) .
    }
    FILTER (!BOUND(?y))
  }
}
ORDER BY (?country)
```

Figure 1. A SPARQL query to DBPedia that retrieves Wikipedia pages of the category *European countries* that do either not belong to the category *Member states of the NATO* or *Member states of the European Union*, respectively.

`FILTER (!BOUND(?y))` construct.[16] Posing the query to the DBPedia SPARQL Explorer[17] indeed returns a list of European countries that are either not in the NATO or not in the EU, e.g., *Albania*, or *Andorra*. These could now be added as counterexamples to the context and the attribute exploration could continue.

One question quickly comes up with this approach: Why should we build the formal context through attribute exploration when we could as easily create it by a single SPARQL query? On the one hand, we want to show in this paper the wide range of possible sources in the web that can support attribute exploration, and linked open data is an obvious candidate. On the other hand, the goal of attribute exploration could be to find errors in the knowledge bases and check them for completeness. This requires human interaction. Compared to querying web search engines, it is considerably easier to exactly specify the requested or unrequested attributes of an object and one has to deal with fewer ambiguities. On the other hand – as we have seen with the attributes *Euro* and *Schengen* of our example context – only a small fraction of the knowledge available in

[16] The upcoming refinement of the SPARQL standard [8] will allow for a more direct way of expressing negation.

[17] http://dbpedia.org/snorql/

Wikipedia (let alone in the web) is accessible using the described method and therefore this approach clearly has its limitations.

5 Implementation

In this section we present a prototype that – since it is freely available on the web – allows everybody to test our approach using web search engines and discover its chances and limitations. We first describe the prototype and then present first insights, limitations, and plans for improvement.

5.1 Prototype

We developed a web-based prototype[18] in Java that implements the querying strategy described in Section 4.2. On its start page, the application allows the user to upload a formal context in a file in the ConExp [19] XML format CEX or to select one of the predefined example contexts. In addition, the user can specify the domain of the objects and restrict the results to a specific site. The prototype is based on the attribute exploration algorithm available in FCAlib.[19] We implemented the `Expert` class of the FCAAPI[20] such that for each implication that the expert shall check, queries are generated and sent to a web search engine (we are using Microsoft Bing,[21] since it provides an API which allows a limited number of free requests per month). The context, the accepted implications, the current implication, the corresponding queries, and the first ten retrieved results for the active query are then shown on a web page to the user who is asked if the result set contains a counterexample (see Figure 2). Each result contains the title and URL of the corresponding web page and a short text snippet from the page that contains the matching query terms. If the user found a counterexample, he or she can add it to the context and the next implication is checked. If no counterexample could be found, the results for the other queries of that implication can be inspected, until either a counterexample can be found or all queries for the implication at hand were inspected. A text input field allows the user to modify the query and retrieve further results from the search engine, if necessary.

5.2 Example

Returning to our context in Table 1, the first implication that can be derived is $\emptyset \to \{\text{NATO}, \text{Schengen}\}$. A counterexample would be a European country that is not a member of the NATO or does not participate in the Schengen Agreement. Since this is an implication with an empty premise and since all objects we are interested in belong to a common class, namely "Countries in Europe" – a

[18] http://greymane.l3s.uni-hannover.de:8888/
[19] http://code.google.com/p/fcalib/
[20] http://code.google.com/p/fcaapi/
[21] http://www.bing.com/

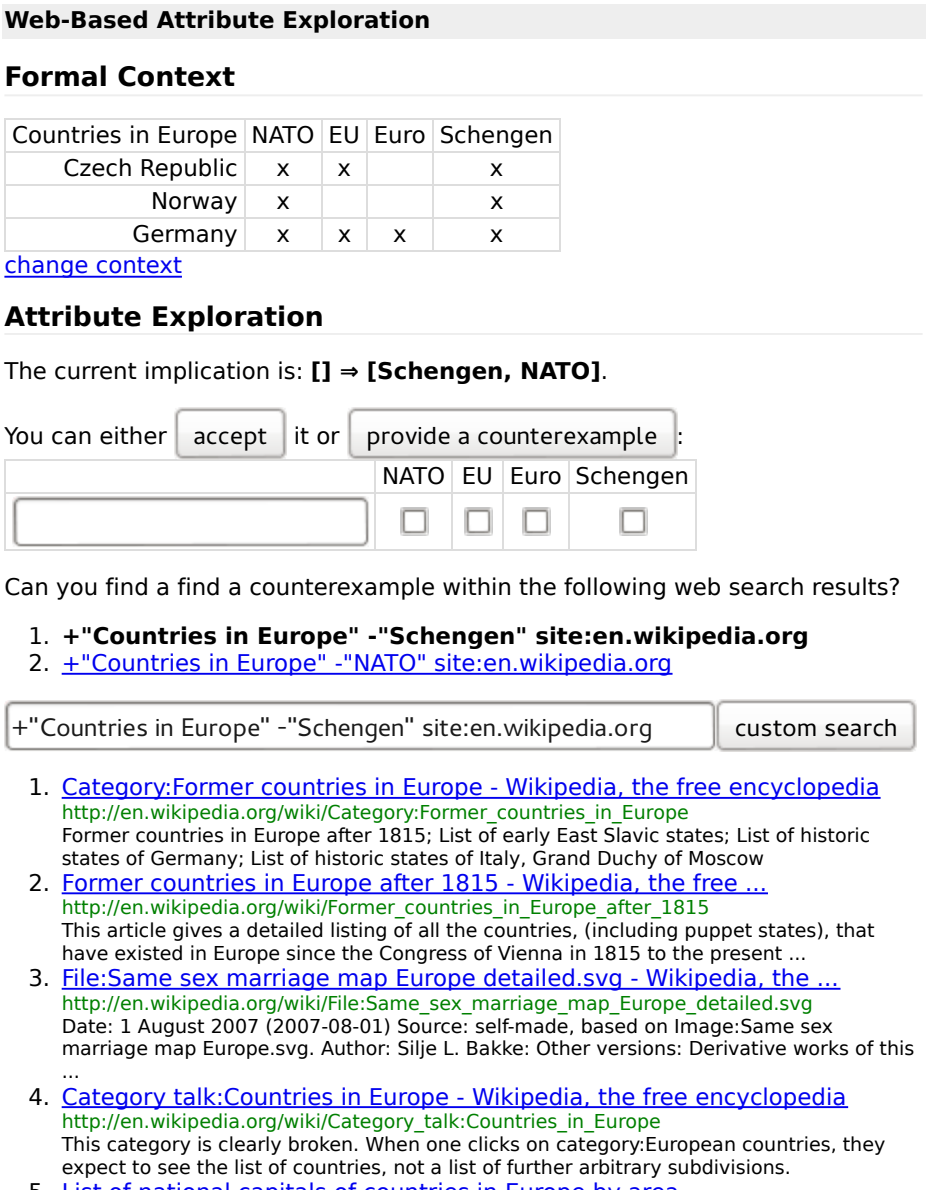

Figure 2. A screenshot showing our prototype implementation.

category of the English Wikipedia,[22] we add the domain restriction `+"Countries in Europe"` and the site restriction `+site:en.wikipedia.org` and obtain the two queries

[22] http://en.wikipedia.org/wiki/Category:Countries_in_Europe

1. `+"Countries in Europe" -Schengen +site:en.wikipedia.org`
2. `+"Countries in Europe" -NATO +site:en.wikipedia.org`

As can be seen in the screenshot in Figure 2, none of the top four results for the first query is a Wikipedia page about a specific country. The same applies to the following six results and also to the top ten results of the second query (which are actually the same as for the first query). However, as we have seen in Section 4.3, there do exist several countries that would constitute a counterexample.

This raises the question *Why are no countries among the top results?* The two likely reasons for the discovered problem can be found in the way web search engines work: they retrieve pages whose *textual* content *matches* the query and return only the top hits according to a *ranking*, which is typically computed by an algorithm like PageRank [13]. The matching against the text of the pages returns many pages that contain the string *Countries in Europe* but do not belong to the corresponding category, which is a property we can not yet enforce with standard web search engines. In addition, for five of the top ten results the string is contained in the page title, which typically increases their ranking score. A PageRank-like ranking further prefers pages that have a high number of incoming links, which is typical for category and listing pages that constitute a large part of the top results. Hence, for our approach it would be very helpful, if we could enforce to receive only pages of a specific Wikipedia category. As a workaround, we can add terms to our query that we would expect to find on a Wikipedia page that is describing an object from the domain at hand. In our example, where the objects are (European) countries, we can assume that every page that describes a country contains a section about *politics*, *history*, *geography*, etc. This assumption is supported by the results we retrieve for the extended query `+"Countries in Europe" site:en.wikipedia.org -Schengen +politics +history +geography`: The top ten results on Bing contain eight countries: Azerbaijan, Spain, Armenia, Greece, Turkey, United Arab Emirates, Ukraine, Vatican City – with the Ukraine being a valid counterexample. Unfortunately, although at least Spain and Greece are part of the Schengen Area, they are returned among the top results. On the other hand, countries like Romania, that are not part of the Schengen Area, are missing. This brings us to the questions *Why are countries missing that do constitute a counterexample?* and *Why are countries returned that do not constitute a counterexample?* One explanation is the limited validity of our assumptions 2 and 3 from Section 4.2: on the one hand, web pages *do not* always contain search terms corresponding to attributes that the objects they describe *do* have (e.g., *Schengen* is not mentioned on the Wikipedia pages of Spain and Greece), and on the other hand, web pages *do* sometimes mention terms corresponding to attributes that the objects *do not* have (e.g., the term *Schengen* is mentioned on the Wikipedia page of Romania, because it is mentioned that the country wants to join the Schengen Area[23]).

[23] http://en.wikipedia.org/wiki/Romania

These examples also show that, even for a human expert, it is often not sufficient to rely on the information about objects that can be found on their web pages. Sometimes it is necessary to investigate further web pages.

6 Conclusion

In this paper we have indicated that there is a potentially wide range of options to employ the web for attribute exploration: querying search engines, asking questions on social question answering or crowdsourcing platforms, or retrieving counterexamples from the linked open data cloud. All of these approaches have their limitations, for some of them we provided examples and explanations. In all cases we have to cope with the open world assumption, i.e., in principle we can not assume from the absence of a fact that the fact is not true.

Nevertheless, the approach presented in this paper can ease the attribute exploration process by automatically posing queries to the web that a human would start with to find counterexamples. The approach can also be employed for learning, such that students can interactively investigate the topic of interest and at the same time learn to search the web, understand the underlying mechanism, and learn to judge the quality of the results.

Future extensions of our approach could mitigate some of the limitations:

- The user could provide more information about the objects and the attributes which could be incorporated into the query (like the additional query terms we have added in Section 5.2).
- A combination of the different sources could improve the efficiency of the process. E.g., one could use structured knowledge bases to automatically find counterexamples and only if none could be found query a web search engine. Based on the results, the user could then decide if the implication holds or not or she could forward the question to a social question answering platform or a crowdsourcing service to see if other people know the answer.
- The retrieved information could be subject of further automated analysis. E.g., web pages might be analyzed by deep semantic analysis tools to detect the presence or absence of the wanted information more reliably.

References

1. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, Berlin/Heidelberg, 2007.
2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications.* Cambridge University Press, New York, NY, USA, 2003.
3. Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. Completing description logic knowledge bases using formal concept analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 230–235, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

4. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.
5. P. Cimiano and S. Staab. Learning by googling. *ACM SIGKDD Explorations Newsletter*, 6(2):24–33, 2004.
6. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.
7. J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95:5–18, 1986.
8. Steve Harris and Andy Seaborne. SPARQL 1.1 query language. W3C proposed recommendation, W3C, November 2012. http://www.w3.org/TR/sparql11-query/.
9. Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, volume 2, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.
10. Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):7–26, 2003.
11. Bjoern Koester. *FooCA: web information retrieval with formal concept analysis*. Beiträge zur begrifflichen Wissensverarbeitung. Verlag Allgemeine Wissenschaft, Mühltal, 2006.
12. Sergei Obiedkov, Derrick G. Kourie, and J.H.P. Eloff. Building access control models with attribute exploration. *Computers and Security*, 28(1–2):2–7, 2009.
13. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.
14. Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C recommendation, W3C, January 2008. http://www.w3.org/TR/rdf-sparql-query/.
15. Alexander J. Quinn and Benjamin B. Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 1403–1412, New York, NY, USA, 2011. ACM.
16. Sebastian Rudolph. Exploring relational structures via $\mathcal{FLE}$. In Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach, editors, *Proceedings of the 12th International Conference on Conceptual Structures (ICCS 2004)*, volume 3127 of *Lecture Notes in Computer Science*, pages 196–212. Springer, 2004.
17. Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.
18. Rudolf Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered sets*, pages 445–470, Dordrecht–Boston, 1982. Reidel.
19. Serhiy A. Yevtushenko. System of data analysis Concept Explorer. In *Proceedings of the 7th national conference on Artificial Intelligence KII-2000*, pages 127–134, Russia, 2000. (In Russian).