UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Profile-based selection of answer candidates for LogAnswer

Diplomarbeit

zur Erlangung des Grades eines Diplom-Informatikers
im Studiengang Informatik

vorgelegt von

Timo Eifler

Erstgutachter:	Prof. Dr. Ulrich Furbach (Institut für Informatik, AG Künstliche Intelligenz)
Zweitgutachter:	Dipl. Inf. Björn Pelzer (Institut für Informatik, AG Künstliche Intelligenz)

Koblenz, im Januar 2012

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☒	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☒	☐

.....
(Ort, Datum) (Unterschrift)

Deutsche Zusammenfassung

In dieser Ausarbeitung beschreibe ich die Ergebnisse meiner Untersuchungen zur Erweiterung des LogAnswer-Systems mit nutzerspezifischen Profilinformationen. LogAnswer ist ein natürlichsprachliches open-domain Frage-Antwort-System. Das heißt: es beantwortet Fragen zu beliebigen Themen und liefert dabei konkrete (möglichst knappe und korrekte) Antworten zurück. Das System wird im Rahmen eines Gemeinschaftsprojekts der Arbeitsgruppe für künstliche Intelligenz von Professor Ulrich Furbach an der Universität Koblenz-Landau und der Arbeitsgruppe Intelligent Information and Communication Systems (IICS) von Professor Hermann Helbig an der Fernuniversität Hagen entwickelt. Die Motivation meiner Arbeit war die Idee, dass der Prozess der Antwortfindung optimiert werden kann, wenn das Themengebiet, auf das die Frage abzielt, im Vorhinein bestimmt werden kann. Dazu versuchte ich im Rahmen meiner Arbeit die Interessensgebiete von Nutzern basierend auf Profilinformationen zu bestimmen. Das Semantic Desktop System NEPOMUK wurde verwendet um diese Profilinformationen zu erhalten. NEPOMUK wird verwendet um alle Daten, Dokumente und Informationen, die ein Nutzer auf seinem Rechner hat zu strukturieren. Dazu nutzt das System ein sogenanntes Personal Information Model (PIMO) in Form einer Ontologie. Diese Ontologie enthält unter anderem eine Klasse "Topic", welche die wichtigste Grundlage für das Erstellen der in meiner Arbeit verwendeten Nutzerprofile bildete. Konkret wurde die RDF-Anfragesprache SPARQL verwendet, um eine Liste aller für den Nutzer relevanten Themen aus der Ontologie zu filtern.

Die zentrale Idee meiner Arbeit war es nun diese Profilinformationen zur Optimierung des Ranking von Antwortkandidaten einzusetzen. In LogAnswer werden zu jeder gestellten Frage bis zu 200 potentiell relevante Textstellen aus der deutschen Wikipedia extrahiert. Diese Textstellen werden auf Basis von Eigenschaften (wie z.B. lexikalische Übereinstimmungen zwischen Frage und Textstelle) geordnet, da innerhalb des zur Verfügung stehenden Zeitlimits nicht alle Kandidaten bearbeitet werden können. Mein Ansatz verfolgte das Ziel, diesen Algorithmus durch Nutzerprofile so zu erweitern, dass Antwortkandidaten, welche für den Benutzer relevante Informationen enthalten, höher in der Rangfolge eingeordnet werden.

Zur Umsetzung dieser Idee musste eine Methode gefunden werden, um zu bestimmen ob ein Antwortkandidat mit dem Profil übereinstimmt. Da sich die in einer Textstelle enthaltenen Informationen in den meisten Fällen auf das übergeordnete Thema des Artikels beziehen, ohne den Namen des Artikels explizit zu erwähnen, wurde in meiner Implementierung der Artikelname betrachtet, um zu ermitteln, zu welchem Themengebiet die Textstelle Informationen liefert. Als zusätzliches Hilfsmittel wurde außerdem die DBpedia-Ontologie eingesetzt, welche die Informationen der

Wikipedia strukturiert im RDF Format enthält. Mit Hilfe dieser Ontologie war es möglich, jeden Artikel in Kategorien einzuordnen, die dann mit den im Profil enthaltenen Stichworten verglichen wurden.

Zur Untersuchung der Auswirkungen des Ansatzes auf das Ranking-Verfahren wurden mehrere Testläufe mit je 200 Testfragen durchgeführt. Die erste Testmenge bestand aus zufällig ausgewählten Fragen, die mit meinem eigenen Nutzerprofil getestet wurden. Dieser Testlauf lieferte kaum nutzbare Ergebnisse, da nur bei 29 der getesteten Fragen überhaupt ein Antwortkandidat mit dem Profil in Verbindung gebracht werden konnte. Außerdem konnte eine potentielle Verbesserung der Ergebnisse nur bei einer dieser 29 Fragen festgestellt werden, was zu der Schlussfolgerung führte, dass der Einsatz von Profildaten nicht für Anwendungsfälle geeignet ist, in denen die Fragen keine Korrelation mit dem genutzten Profil aufweisen. Da die Grundannahme meiner Arbeit war, dass Nutzer in erster Linie Fragen zu den Interessensgebieten stellen, welche sich aus ihrem Profil ableiten lassen, sollten die weiteren Testläufe genau diesen Fall beleuchten. Dazu wurden 200 Testfragen aus dem Bereich Sport ausgewählt und mit einem Profil getestet, welches Stichworte zu unterschiedlichen Sportarten enthielt. Die Tests mit den Sportfragen waren wesentlich aussagekräftiger. Auch hier deuteten die Ergebnisse darauf hin, dass der Ansatz kein großes Potential zur Verbesserung des Rankings hat. Eine genauere Betrachtung einiger ausgewählter Beispiele zeigte allerdings, dass die Integration von Profildaten für bestimmte Anwendungsfälle, wie z.B. offene Fragen für die es mehr als eine korrekte Antwort gibt, durchaus zu einer Verbesserung der Ergebnisse führen kann. Außerdem wurde festgestellt, dass viele der schlechten Ergebnisse auf Inkosistenzen in der DBpedia-Ontologie und grundsätzliche Probleme im Umgang mit Wissensbasen in natürlicher Sprache beruhen. Die Schlussvolgerung meiner Arbeit ist, dass der in dieser Arbeit vorgestellte Ansatz zur Integration von Profilinformationen für den aktuellen Anwendungsfall von LogAnswer nicht geeignet ist, da vor allem Faktenwissen aus sehr unterschiedlichen Domänen abgefragt wird und offene Fragen nur einen geringen Anteil ausmachen.

Contents

1 Introduction

In this paper I present the results of my work on enhancing the question answering system LogAnswer with profile-based information. The impulse for my work had been to integrate the ontology of the NEPOMUK semantic desktop system into LogAnswer. After some elaboration the task evolved to using the NEPOMUK system as a source for user related information (i.e. a user profile) which is then used to personalize the question answering process. Specifically the user profile is used to enhance the ranking of text passages which the LogAnswer system selects from the wikipedia as potential answer candidates. The goal here is to identify text passages containing information which is relevant to the user.

In chapter 2 I explain some of the basic instruments I used in my work and also give a short introduction to LogAnswer and the NEPOMUK semantic desktop system. In chapter 3 I give a detailed description of my approach. I start by describing why it is suggestive to use profile information in natural language question answering systems and show how I tried to integrate profile information into LogAnswer in particular. I explain how a profile looks like and how it can be obtained from the NEPOMUK system and also give a notion for alternative approaches. In chapter 3.3 I explain how the profile was used to match answer candidates relevant to the user and I describe how the DBpedia ontology was utilized as a core component of my approach. Chapter 3.4 contains details about the implementation I used for the validation of my approach which is then detailed in chapter 4. This chapter also contains various examples showcasing the problems which came up during my work. Finally I give a conclusion of my work and a short outlook for future work in chapter 5.

1.1 Motivation

Personalization is an ever-growing topic not only in the field of computer science. In a world where applications become more and more complex users have high expectations when they try out a new technology. New users expect services to work and deliver results according to their anticipation. Therefore many developers try to adjust the implementation of their services to satisfy their customers. For example in [RLJL97] the authors describe a machine learning method to integrate the driver's familiar routes into the route planning algorithm of the car-IT-system. So when a new driver uses the system, it will learn his preferred routes and try to act accordingly. Another example can be found in [WBW+02] where the authors describe how mobile devices can be personalized by matching the user's profile with the semantically enriched description of services. In this example user profiles are used to automatically select, compose and execute services which are relevant for the user.

In the field of question answering the applications for personalization might not be as obvious. The main idea which motivated my work is that it is easier to answer a question if you know which knowledge domain is targeted or what the questioner had in mind when posing the question. While in this paper only artificial systems (or rather only LogAnswer in particular) are regarded, this statement could also be made for humans. If a person is asked a question to which he does not know the answer immediately, it is easier to answer it, if he knows in which direction he has to think. The knowledge about the domain might give him a notion of what the questioner wants to know. If the person does not know the answer at all, he can at least make a guess which is not totally out of place. A good example are open questions for which no one correct answer exists. While there are often multiple correct answers for questions of this type only some of the answers might be relevant for the questioner. In this case knowing the fields of interest of the questioner (i.e. the targeted domain) can help to identify relevant answers and hence improve the answering process.

2 Fundamentals

2.1 Knowledge Representation

For my work different types of knowledge representation are important. Since LogAnswer is a natural language question answering system most of the data I work with is represented in natural language only. All the input data for the LogAnswer system is represented in natural language: The question is entered manually by the user and the background knowledge is extracted from the German wikipedia. This type of data is mostly unstructured and therefore not machine readable. Special mechanisms like natural language parsing have to be used to access information contained in this data. The natural language parser used in the LogAnswer system generates a semantic representation of the data in the Multinet formalism (for a detailed description see [Hel06]). With this step the data is transformed into a more structured representation which also offers some form of inference. As the reasoning part of the system is logic based, the representation of the question as well as the background knowledge has to be transformed again into first-order logic. Well-defined inference rules can be used to reason over this type of data. I only mentioned these two types of knowledge representation for the sake of completeness but since they are not directly relevant for my work I will not go into detail about them.

2.1.1 Semantic Web

Another type of knowledge representation which is very important for my work is linked data. It is most often used in the context of the Semantic Web. The World Wide Web Consortium (W3C) describes the Semantic Web as follows:

> In addition to the classic "Web of documents" W3C is helping to build a technology stack to support a "Web of data," the sort of data you find in databases. The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term "Semantic Web" refers to W3C's vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS. [1]

So the Semantic Web tries to make the data on the web as easy to access as data in databases. Another advantage of linked data (as the name suggests) is that different datasets can be linked to each other. So ideally starting from

[1] http://www.w3.org/standards/semanticweb/

a given RDF graph, it is possible to explore more knowledge incrementally by following the links to other graphs.

The common data format is RDF. Here is a short description of the RDF standard, also given by the W3C:

> RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.
>
> RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a "triple"). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications.
>
> This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes.[2]

Each entitiy which is described in an RDF graph is called a resource and is identified by a URI. Usually the URI is a weblink which points to the RDF document where the description of the entity can be found. Information is stored in so called RDF triples: `SUBJECT PREDICATE OBJECT`. The `PREDICATE` describes the relation in which two resources (the `SUBJECT` and the `OBJECT`) are connected. For example:

```
die_glocke has_author schiller
```

With query languages such as SPARQL this knowledge can be accessed directly by programs. SPARQL is reminiscent of the database query language SQL. A simple SPARQL query could look like this:

```
SELECT ?s WHERE {?s hasAuthor schiller}
```

The "?" indicates that s is a variable. So this query selects all subjects that are related to the object "schiller" by the relation "hasAuthor" (i.e. it selects each document whose author is Schiller). The fact that more and more knowledge is represented in this way means that information which was only available in natural language is now machine readable which also opens possibilites for new innovative applications. A perfect example for this is the DBpedia ontology [CB09] which contains the information of the wikipedia formatted in RDF. This ontology is also used in my work (see

[2] http://www.w3.org/RDF/

chapter 3.3.1). Another use case for linked data is showcased in the Semantic Desktop project NEPOMUK which I used in my work to extract profile information (see chapter 2.4). Figure 1 shows an extract of the PIMO ontology (which is used in the NEPOMUK system) as an example for an RDF graph. It shows the description of the class "Topic". Each tag line (marked with "< >") represents one RDF triple. The tags contain a `PREDICATE` (e.g. "rdf:type") and an `OBJECT` (e.g. "http://www.w3.org/2000/01/rdf-schema#Class"). The `SUBJECT` of each triple is the resource which the description refers to (in this case the class "Topic").

```
<rdf:Description rdf:about="http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#Topic">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:label>Topic</rdfs:label>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
<rdfs:subClassOf rdf:resource=
"http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#ClassOrThingOrPropertyOrAssociation"/>
<rdfs:subClassOf rdf:resource="http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#Topic"/>
<rdfs:subClassOf rdf:resource="http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#ClassOrThing"/>
<rdfs:subClassOf rdf:resource="http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#Thing"/>
<rdfs:subClassOf rdf:resource="http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#Thing"/>
<rdfs:comment>
A topic is the subject of a discussion or document.
Topics are distinguished from Things in their taxonomic nature,
examples are scientific areas such as "Information Science", "Biology",
or categories used in content syndication such as "Sports", "Politics".
They are specific to the user's domain.
</rdfs:comment>
<nrl:directSubClassOf rdf:resource="http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#Thing"/>
<nrl:instanceCount>9</nrl:instanceCount>
</rdf:Description>
```

Figure 1: Extract from the PIMO ontology showing the description of the "Topic" class

2.2 User Profiles

A user profile is a collection of information about a person. There are various areas of application for the usage of user profiles and each of them has different requirements. Therefore a user might have a number of profiles which are very distinct from each other because they are all constructed for different purposes. While I only use one specific type of profile in my work the results might be used to extend my approach for use cases with different profile-types which is why I describe different types of profiles here. The most obvious distinction between different profiles is the type of information contained in them. For example a user profile covering the personal taste in music compared to one that contains information about the user's style of driving. These two profiles might even be used in the same system (e.g. a car-IT-system) but they are not interchangeable at all.

Another distinctive feature is the structure of the profile. While the music taste could be described by a simple list of music genres, the description of a driving style may need different attributes and predefined scales and values. With these it is possible to directly access specific data in the pro-

file and directly base decisions on it. Hence a profile which contains more structured information also offers more possibilities for processing the information.

Different types of profiles also differ in the way they are created and the resources they are based on. The most obvious way to create a profile is to offer a form which the user manually fills in. This approach involves some difficulties though. To ensure that all data which is needed for the application is contained in the profile the corresponding fields in the form can be made mandatory. Still things like spelling mistakes, nonsensical or wrong input (intentional or unintentional) and sometimes even input in a different language might cause problems. First and foremost the user has to be willing to create a profile. These factors have to be considered when implementing an application that uses manually created profile data and sometimes it is not even possible to circumvent all of them. Another method to create user profiles which is often used, is to collect information about the user by observing his actions and analyzing his data. A perfect example for this are profiles created by online shops like for example Amazon. By collecting and evaluating data about the user's buying behaviour (e.g. which products he is interested in and which products he buys in combination), detailed profiles about a customer can be created automatically. These kind of profiles however highly depend on the data evaluation and therefore might not always be accurate. Though if the data is accurate and there is no need for evaluation then the resulting profile is also guaranteed to be accurate. A profile describing the user's style of driving for example might contain attributes like "average speed", "maximum speed", "average fuel consumption" etc. This kind of data is collected directly by sensors and does not have to be evaluated. Thus the corresponding application can use the profile as a reliable source.

The data which is used in my work is extracted from an ontology describing the personal information model of the user. As described above this offers the possibility to directly access the data with a query language like SPARQL. This way parts of the ontology can be extracted and composed into a profile which then contains only the information needed for the application (e.g. LogAnswer). A detailed description of how the PIMO ontology is used to create a user profile can be found in chapter 3.2

2.3 LogAnswer

Loganswer [FGHP08] is an open-domain, natural language question answering system. This means that the system is not restricted to a specific domain like for example an expert system which could only answer questions corresponding to that one domain. Using profile information in such expert systems would not make much sense since if every question targets the same domain there is no need to determine it in advance. In contrast

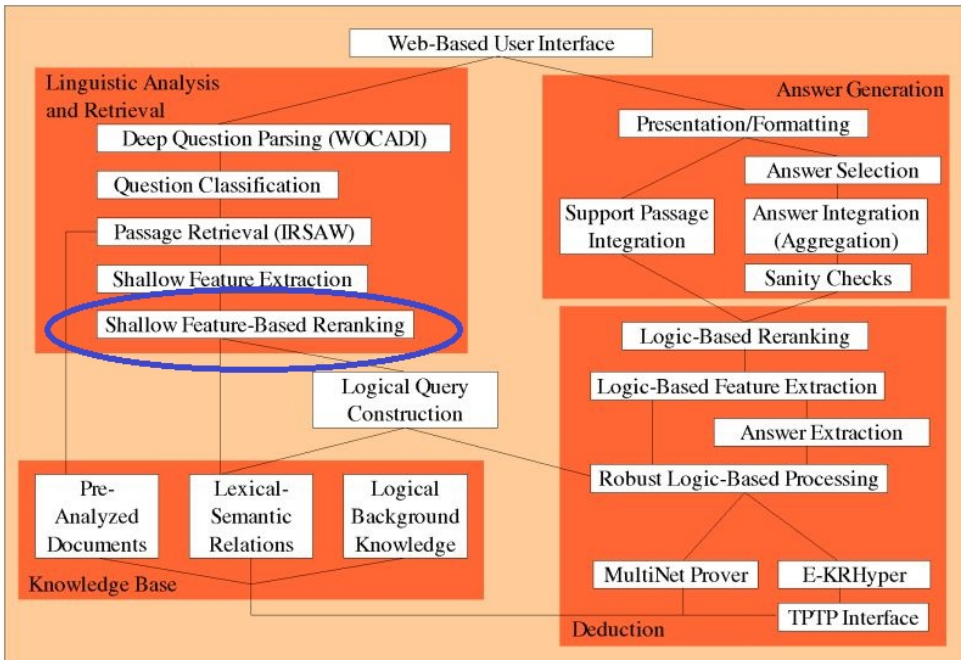

Figure 2: Architecture of the LogAnswer system with the relevant part for my work marked (adapted from [FGHP10])

to regular search engines a natural language question answering system processes questions as a whole instead of unconnected keywords. Another main difference is that the system returns a specific answer which should ideally be concise and correct instead of a list of references. LogAnswer uses the German wikipedia as its main background knowledge to retrieve these answers. The system is developed within a joint project of the working group artificial intelligence (AGKI) of Prof. Dr. Ulrich Furbach at the University of Koblenz and the working group Intelligent Information and Communication Systems (IICS) of Prof. Dr. Hermann Helbig at the University of Hagen. In this paper I will only give a short overview over the main components of the system and will refer to more specific documents for readers who are interested in details of the system's architecture.

The easiest way to access the system is provided through the web interface at `www.loganswer.de`. When a question has been entered into the web search box it will be parsed and a representation in the MultiNet formalism [Hel06] will be created. After that the system will select up to 200 pre-analyzed passages which might contain the answer to the question. These passages are then ranked using a machine learning approach based on different features [Glö08, Glö09, GP08]. This is also the part of the system where my work is applied. Therefore more details about the

ranking process and how my work affects it will be presented in the following chapters. In the next step a logical representation of the question is generated and LogAnswer tries to prove it with each passage and the background knowledge using the theorem prover E-KRHyper [PW07]. The passages are processed in the order of their ranking since time constraints prevent the system from reasoning over all 200 passages. If a proof is found for one of them an answer is extracted accordingly. LogAnswer will also extract some logic-based features which are then used for a reranking of the extracted answer candidates. After this reranking and some sanity checks [Glö07] the five answers with the best ranking are presented together with the text passages from which they were retrieved. Figure 2 gives an overview of the system's architecture. The relevant part for my work (the ranking of the answer candidates) is marked.

2.4 NEPOMUK Semantic Desktop

The idea of the Semantic Desktop [Sau07] is to implement the concepts of the Semantic Web into a desktop system to organize and integrate all of the user's personal data and applications. The Social Semantic Desktop developed in the NEPOMUK project [GHM+07] goes even one step further by connecting and integrating multiple desktop systems and sharing information between those. The main advantage of this system in particular is that it offers very detailed documentation for developers on the project's website (`http://nepomuk.semanticdesktop.org/`). This made working with the system very comfortable and it was also one reason why this system in particular was chosen for my approach. The common knowledge representation format in the system which is also adapted from the Semantic Web is RDF (Resource Description Format)[3]. As described above RDF offers structured information that can easily be queried for. The NEPOMUK Semantic Desktop stores all the information about the users data in an ontology called PIMO (Personal Information Model). [SED07] describes PIMO as follows:

> It is a formal representation of the structures and concepts an individual knowledge worker needs, according to her or his personal mental model. It is an application-independent and domain-independent representation.

Figure 6 in the appendix shows a screenshot of the NEPOMUK application with the PIMO perspective opened. This perspective offers an interface which makes exploring and editing the contents of the ontology very comfortable. Every element in the PIMO ontology is part of the superclass "Thing", as that is the root of the ontology. There are some predefined subclasses like "Person", "Document", "Organization", "Location" or "Topic"

[3] `http://www.w3.org/RDF/`

but each user may also define his own classes. The structure and the complexity of the ontology therefore depend on how much effort the user puts into it. Regarding my work the crucial part of PIMO is that it can be used to extract a profile of the user. The predifined class "Topic" contains keywords which are used to sort and categorize the user's data. As each document in the user's system can be related to a topic the elements of this class potentially offer very detailed information about the user's fields of interest. Hence it is very easy to query the ontology for all topics that the user is interested in. The RDF repository which contains the ontology can be accessed through an integrated server which makes extracting the information even easier. Chapter 3.2 contains detailed information on how the user profiles can be obtained from the NEPOMUK system.

3 Approach

There are two main ideas behind my work: The first idea is that people will most often ask questions about domains in which they are interested. So identifying a user's fields of interest could be used as a heuristic to guess what domains are targeted by the questions he asks. This in order makes answering the questions easier. The second idea is that knowing the fields of interest of a user can be very helpfull when answering open questions which may have multiple correct answers. Here the user's profile could be used to filter away the answers which are not relevant for this particular user.

If knowledge about the targeted domain of a question is helpful for artificial systems depends among other things on the background knowledge which is used to find an answer. Usually the background knowledge is structured in some way and may even be sorted by topics to a certain degree. So if the topic of a question is known, it is also evident which part of the knowledge base has to be regarded (and which parts may be omitted). In the case of LogAnswer the main background knowledge is the German Wikipedia[4]. Here the structure is given by the partitioning into different articles. Each article contains information about a different topic which can be identified by its title. Using the DBpedia ontology it is even possible to group topics by categories which makes it possible to filter all articles from a given domain. Depending on the scope of the domain this step has the potential to cut down the search space for answers considerably. The integration of the DBpedia ontology is detailed in chapter 3.3.1. On the other hand however if the background knowledge is not structured at all, it becomes more difficult to apply this method. For example in the 2011 CLEF competition[5] the competing systems were only allowed to use unstructured texts (i.e. speeches) as sources for answering the given questions. Besides the fact that the LogAnswer system participated in this competition, this use case will not be covered in this paper and I will instead focus on the common application of the system.

To be able to filter the knowledge base as described above, the domain of the question has to be known in the first place. There are scenarios where this additional information is available or at least methods (or heuristics) exist to determine it. A perfect example for this is the IBM *Jeopardy* Challenge [DFW10]. Here the categories of the questions offer additional information that may be used to derive the domain which is targeted. In the current setting for LogAnswer however, such additional information is not available. So the idea of my approach is to use information about the interests of the questioner to derive the targeted domain. Assuming that

[4] http://www.wikipedia.de
[5] http://clef2011.org/

people will often ask questions on specific domains (in which they are especially interested), the question answering process could be optimized by identifying these. Of course this means that there is no definite knowledge about the domain since users will also ask questions which differ from their customary interests. In these cases it is possible that the consideration of the user's profile causes a worsening of the results. Detailed information about this effect is provided in chapter 4.

3.1 Integrating Profile Information into LogAnswer

The first step in my work was to identify the best way to integrate the profile information into the LogAnswer System. The initial idea was, to use the ontologies from the NEPOMUK Semantic Desktop as additional background knowledge for the reasoning part. As the PIMO ontology is used to structure the user's data, it can be used to retrieve additional knowledge. Since there are already attempts to use ontologies such as Opencyc [6] to extend the background knowledge, this approach seemed promising. The problem with this approach is that the additional knowledge provided by the PIMO ontology is already known to the user, since he created it himself while maintaining the ontology. So it is very unlikely that the user would ask a question whose answer can be derived using his personal information model. Therefore this idea was dismissed.

Using the ontology to extract a user profile which could then be used to enhance the ranking of the retrieved answer passages proved to be a much better approach. There are two ways to combine the profile information with the ranking process. One way is to integrate the information directly. The ranking of the candidate passages is done using machine learning algorithms. As described in [Glö09] shallow features like for example "lexical overlap between question and candidate passage" are used to construct a decision tree. So a correlation between the questioner's profile and a given answer passage could simply be used as one additional feature for the machine learning. The constructed decision tree would then automatically rank those passages higher. The only disadvantage here is that the results depend on the training program of the machine learning algorithm and therefore it becomes difficult to evaluate the results.

The other way to use the profile information in combination with the ranking of passages is to use it as a filter. In this approach every candidate passage which does not correlate with the user's profile is entirely skipped. Of course the outcome depends strongly on the use case. The perfect scenario would be one where for all questions asked the correlation with the used profile is ensured. Again a situation similar to the IBM *Jeopardy* Challenge is a good example where this condition is met. In [DFW10]

[6] http://www.cyc.com/opencyc

the authors suggest that the usage of Watson as an expert system to support professionals is the major area of application. In business scenarios it is very likely that each user will only be interested in information about their fields of work. If the special fields of each user are described in his profile the question answering system can use this profile as a hard filter. However in an open domain application where users will ask questions from very mixed categories which often differ from their common domains of interest, this approach is not very effective. As the user's profile is usually narrowed down to the user's main fields of interest, too many otherwise correct answers would be ignored because they do not match. On the other hand if the profile was extended to mitigate this effect, it would also become less effective, as for each question more irrelevant candidate passages would pass the filter.

During my work both mentioned approaches were regarded. In chapter 4 where I present detailed test results the pros and cons as well as different use cases for both approaches are described.

3.2 Obtaining the Profile Information

To obtain the profile information needed for my approach from the NEPOMUK system you first have to decide which types of information from the ontology should be included. As can be seen in Figure 6 the PIMO ontology defines different subclasses of "Thing" (the superclass of all resources in the ontology) like "Location", "Group of Persons" or "Topic" which the user can use to classify his data. While it is also possible to define new classes and subclasses I decided to confine myself to the predefined classes to ensure that my method is usable with the standard configuration of the system. The most relevant subclass is "Topic" since it reflects the idea of representing knowledge domains in the profile the best. Therefore I included the elements of this subclass in all the profiles I used during my work. The other subclasses I tried to use in some of the profiles are "Location", "Event" and "Organization". Typical examples for the class "Location" are cities and and countries which are potentially good additions to a user's profile if he is interested in information about these locations and uses the question answering system to gather this information. But having locations in the user profile also has some special disadvantages which will be discussed in chapter 4. The problem with events is the fact that a particular event has to be known to the background knowledge-base used for answering the questions, which should only be the case for very big or annual events. If this is not the case, then knowing that the user is interested in this event is not useful. Quite the contrary finding matches with the terms in the name of the event could lead to worse results. While the same arguments apply for the inclusion of names of organizations into the profile, these often have unique names which are unlikely to misguide the answer-

ing process. PIMO also contains classes like "Document", "Task" or "Person" which are not suited at all to be used in my approach. Theses classes are important for the actual use case of NEPOMUK to classify the corresponding entities. However the instances of these classes can not really be viewed as knowledge domains and their names will rarely be helpful in determining the fields of interest of the user. One could argue that the names of documents might be useful, but the topics of the interesting documents should already be represented in the "Topic" class.

After deciding which classes of the PIMO ontology would be used to create the profile the instances of these classes had to be extracted from the system to create the actual profile. Since the data is stored in an RDF repository the instances can be accessed with a simple SPARQL query. The result of this query can then be used as a list of keywords which forms the user's profile. For example to extract all topics which the user is interested in (i.e. all instances of the "Topic" subclass) the following query can be used:

```
SELECT ?s WHERE {?s rdf:type pimo:Topic}
```

Extending the profile (for example by names of organizations which are relevant for the user) can be achieved with a query like this one:

```
SELECT ?s
WHERE {{?s rdf:type pimo:Topic }
UNION {?s rdf:type pimo:Organization}}
```

The NEPOMUK system is the only source of user specific data which is regarded in my work. However there are many other potential sources to extract comparable data. For example social network platforms like Facebook store extensive user profiles which may be used in future implementations of natural language answering systems. For LogAnswer in particular it also seems reasonable to implement user accounts where each user can create its personal profile manually by choosing topics which he is interested in. The implementation of my approach is very flexible in this regard since the interface for the profile information expects just a list of keywords representing the fields of interest of the user. So as long as the profile still consists of a list of keywords the results of my work would be applicable.

3.3 Finding Matches

To be able to integrate the user's profile into LogAnswer a method to find matches between the candidate passages and the profile had to be found. The Wikipedia which is the source of the candidate passages is divided into different articles. This is reflected by the way texts in the Wikipedia are written. Since it is obvious that each statement in an article is made

with regards to its topic, the name of the topic itself (e.g. the title of the article) is rarely used in the text. So to find out to which topic the information in a given text passage is related it is not very helpful to look at the text itself. Another reason for this is that in the LogAnswer system in its current state, it is not possible to combine information over the boundaries of one sentence without further ado. Therefore it is not possible to relate information about an object to it, if it is not directly referenced in the same sentence. Example: "Schon kurz nach Fertigstellung stand das Schloss bei Abwesenheit des Königs Besuchern zur Besichtigung offen"[7] This sentence provides information about the castle "Schloss Stolzenfels" and is extracted from the corresponding Wikipedia article. But without that knowledge it is not possible to relate the text passage to a specific castle. For my work I tried to exploit this structure of the Wikipedia. While not every piece of information contained in an article is directly related to the articles title most of the statements refer to it. So the task of matching a candidate passage to a user's profile is transformed into matching the title of the article to which the passage belongs to the profile. Because if the user is interested in the main topic of an article it is likely that he is interested in the pieces of information contained in that article too.

3.3.1 Utilizing the DBpedia Ontology

A user's profile only consists of a relative short list of keywords which offers a more or less detailed description of the domains the user is interested in. However each knowledge domain might have many (sometimes very specific) subdomains which can not be related to the user's profile in a trivial way. For example a profile might contain the keyword "sports" which indicates that its owner is interested in sports in general. Without additional efforts however it is not possible to match an article about "tennis" to that profile even if it clearly should match (assuming the profile does not also contain the keyword "tennis"). To infer if an article belongs to a given domain an additional knowledge base is needed. Conveniently for my use case such a knowledge base already exists in the form of the DBpedia ontology [CB09]. It contains the information of the Wikipedia in a structured format (RDF) and also offers additional information in the form of semantic taggings. Since LogAnswer uses the German Wikipedia as its background knowledge I also use the German version of the DBpedia [8]. For the examples regarding the DBpedia ontology I will use a more convenient notation: Since each resource in the DBpedia starts with the same prefix: `http://de.dbpedia.org/resource/` I will use the short replacement "`Resource:`" from here on whenever I refer to a DBpedia resource. There

[7] Short after its completion the castle was already open for visitors while the king was absent

[8] http://de.dbpedia.org/

are two relations in that ontology in particular that are very usefull. The first (and most important one) ist the subject relation:

```
http://purl.org/dc/terms/subject
```

which I will refer to as "`Subject`" from here on. In the RDF document which contains the description of this relation it is commented with "The topic of the resource", which describes it pretty well. In the DBpedia ontology it relates resources (in this case articles) to broader knowledge domains (i.e. categories). The article about the castle "Schloss Stolzenfels" for example is represented in the DBpedia by the resource:
`Resource:Schloss_Stolzenfels`. The categories related to this resource by the `Subject` relation can be retrieved with the following SPARQL query:

```
SELECT ?category
WHERE {Resource:Schloss_Stolzenfels>
   Subject ?category}
```

The result of this query is this list of resources:

```
Resource:Kategorie:Erbaut_im_13._Jahrhundert
Resource:Kategorie:Architektur_(Preußen)
Resource:Kategorie:Erbaut_in_den_1840er_Jahren
Resource:Kategorie:Kulturdenkmal_in_Koblenz
Resource:Kategorie:
   Kulturlandschaft_Oberes_Mittelrheintal
Resource:Kategorie:Geschütztes_Kulturgut_in_Koblenz
Resource:Kategorie:
   Neugotisches_Bauwerk_in_Rheinland-Pfalz
Resource:Kategorie:Schloss_am_Mittelrhein
Resource:Kategorie:Schloss_in_Koblenz
Resource:Kategorie:Museum_in_Koblenz
```

All of these resources have the prefix "Kategorie:" (i.e. category) which is why I refer to them as "categories" rather than "topics". Each of the categories describes a different feature of the resource it is related to. For example the categories here state amongst other things that the castle was built in the 13th century, that it is situated in the Middle Rhine region and that it is a cultural monument in Coblence. This additional information could also be used to infer knowledge about a given resource but I will not go into detail about this. In the context of this work the categories will be used to decide if the corresponding article is potentially relevant for the questioner by matching the categories with the user's profile. The advantage of looking at the categories is that they act as a generalization since they classify the article in a larger scope. This makes it possible to relate

even very specific articles to a broader knowledge domain if the article is categorized accordingly.

The second relation that I found usefull for my work is:

`http://dbpedia.org/ontology/wikiPageRedirects`

which I will refer to as `Redirect` from here on. Although I already explained how I try to find relevant articles by matching the profile to the categories of that article I still have to look at the title of the article itself. Since if you look at the example of the castle "Schloss Stolzenfels" again one can see that the term "Stolzenfels" does not actually appear anywhere in the categories. If there happens to be a user who is interested in "Stolzenfels" (whether it is the castle in particular or the Coblence district which is also called "Stolzenfels") and therefore has this keyword in its profile, this article about the castle should be a match to that profile. For this reason I also look for direct matches between the title of an article and the profile. On this occasion I also search for matches with terms which redirect to the given article using the `Redirect` relation. In most cases this relation is used to handle synonyms but sometimes it is also used to redirect a search for a term without its own article to the article where its description can be found (for example in the German Wikipedia a search for "Speicherbedarf"[9] will be redirected to the Wikipedia article "Arbeitsspeicher"[10]). Hence if for some reason a synonym is used as a keyword in a user's profile instead of the constituted title of an article, it can be matched using the list of redirects given by the `Redirect` relation.

I also considered other relations which are used in the DBpedia. For example the relation

`http://dbpedia.org/ontology/wikiPageWikiLink`

relates an article to all other Wikipedia articles that are referenced by a hyperlink within the article. Using this information did not prove to be useful since there are too much hyperlinks in an average article and the domains covered by the linked articles often differ too much. Therefore finding a match with a linked article gave no real indication if the root article should also be regarded as a match. After all the `Subject` and `Redirect` relations offered the best results in my test runs which is why I confined myself with these.

3.3.2 Matching Strings

For the actual comparison between two strings I used a matching function which is already integrated in the LogAnswer system. The WOCADI

[9] memory consumption
[10] main memory

parser [Har02] is used to perform a linguistic analysis of two input strings and the matching function returns a value between 0 and 1 as a measure of equality. Each input string is divided into tokens (i.e. words) of different categories which are assigned different quantifiers ("category-weights"). The categories (with corresponding quantifiers) I used are:

- proper names (1.0)
- numbers (0.9)
- nouns (0.9)
- base word of a compound (0.8)
  (e.g. "Verfahren" in "Eilverfahren")
- adjectives (0.8)
- modificator of a compound (0.7)
  (e.g. "Eil(e) in "Eilverfahren")
- adverbs (0.7)
- default (0.4)
  (all uncategorized word types)

Simply described the weights are used to decide which word classes are more important when determining if two phrases should match. Obviously matching names or nouns give a much stronger indication for equality than matching prepositions which is reflected by the scores. For example "Die Bibel" should not match with "Die Glocke" but "Fußball" should match with "Fußballspieler". Besides this "category-weights" the matcher can also be modified by different "criteria-scores" which define quantifiers for different matching criterias. The criterias (with corresponding quantifiers) I used are:

- case-sensitve comparisson (1.0)
- non case-senstive comparison (0.9)
- lemma comparison (0.9)
- concept comparison (0.8)
  (meaning of the words using a lexicon)
- comparison of close synonyms (0.7)
- comparison of all synonyms (0.6)
  (all synonyms know to LogAnswer)

- comparison including nominalisations
  and adjective-attribute relations (0.5)
  (e.g. “lieben” vs. “liebe”, “hoch” vs. “höhe”)

- default (0)

The use of the criterias is pretty self explanatory. Since this default configuration as well as the the configuration of the “category-weights” offered good results I did not change any of the scores. I used 0.5 as a threshold, which means that I regarded two strings as a match if the matcher returned a result greater than 0.5. For my test runs this threshold offered good results since I wanted to have a relative low acceptance barrier. However for future uses the parameters might have to be adjusted to the area of application and the respective requirements. For example the threshold could be set to a higher level to accept only more exact matches to the profile.

3.4 Implementation

In this chapter I am going to describe the system I used for the validation of my approach. Figure 3 gives an overview of the different components and how they were used. The source code can be found at `http://userpages.uni-koblenz.de/~eifler/ProfileTest/src/`

One requirement for my work was that my approach could be tested without having to change the code in the actual LogAnswer system. Therefore a test environment was created which made it possible to access the LogAnswer server to ask questions and then work on the results localy. Besides the use of the matching function which is needed for the actual test runs I did not implement a direct connection to the system. The reason was that if my results showed that the inclusion of profile information had potential, my approach would have to be reimplimented directly into the system anyway. Since the test environment was optimized for a manual interaction via console commands, I used simple shell scripts to get the data I needed. I used a Java application to analyze the output of the LogAnswer test environment and to interact with the DBpedia, the NEPOMUK system and the MySQL database which I used to store the results of the test runs (Figure 5 in the appendix shows an ERM diagram of this database). I chose Java since this enabled me to use the Jena framework [11] which makes working with RDF data very comfortable. For example Jena offers predefined classes for accessing SPARQL endpoints and processing the results. I used these classes to send the queries to the SPARQL endpoint of the DBpedia: `http://de.dbpedia.org/sparql` and to extract the profile information from the PIMO ontology via a local SPARQL endpoint.

[11] http://jena.sourceforge.net/

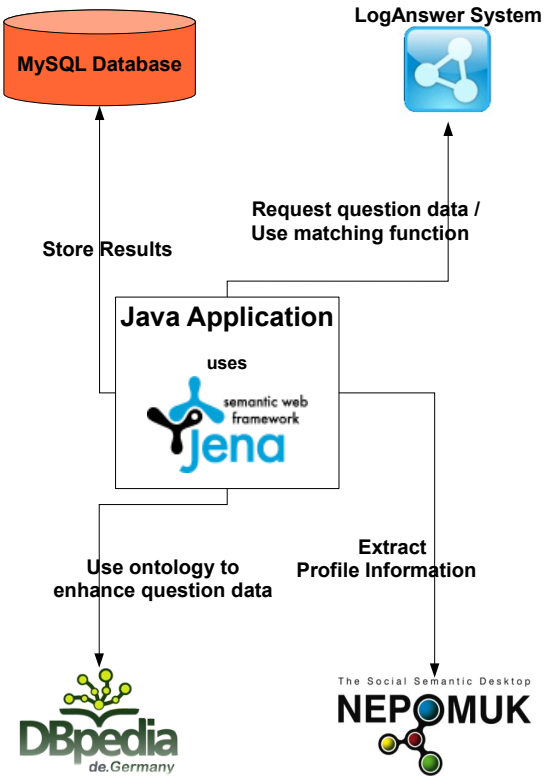

Figure 3: System components used for the test runs

For each test run the question data had to be provided in the local LogAnswer environment. Thus the corresponding questions were sent to the server to retrieve the ranked answer candidates in advance. The PIMO ontology on which the individual profile information is based also had to be created before the actual test run. However since the NEPOMUK system is only an exemplary source for the user related information, the actual implementation of the profile is just a list of keywords (i.e. the topics) to ensure flexibility. Because the interfaces of the methods are implemented accordingly the user profile could also be extracted from a different source or created manually. After these requirements are met the actual test run was started. The question data which consisted of 200 questions, their unique identifiers and 20 ranked answer candidates each was loaded.

For each answer candidate the name of the article from which it was taken was extracted. Since the article names directly correspond to resources in the DBpedia ontology it is very easy to query for the related subjects and redirects (as described in chapter 3.3.1). The results were processed and used to enhance the candidate data. Now for each candidate a list of associated keywords was created which consisted of the articlename, the corresponding categories and the redirects. To compare this list to the user profile the matching function of the LogAnswer system was called once for each pair of keywords (so for n associated keywords and m keywords in the user profile the function had to be called n*m times). The candidate was regarded as a match, if at least one call of the matching function returned a value greater than the threshold of 0.5 which meant that any one keyword from the profile matched with any one keyword associated to the answer candidate (title of the article, redirect or category). The matches were stored together with question and profile data in a MySQL database. The execution of one test run is also detailed in a sequence diagram in Figure 4.

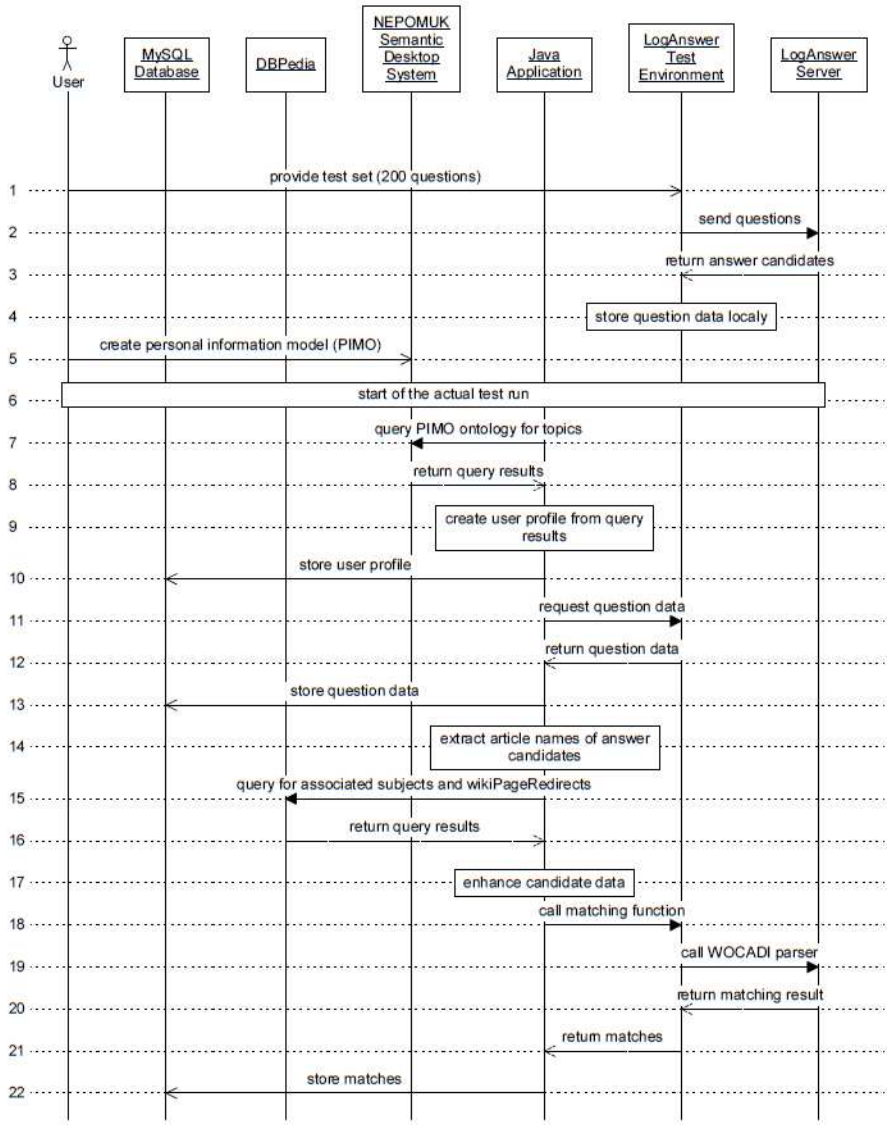

Figure 4: Sequence diagramm showing the execution steps of one test run

4 Validation

During my research I tested the effect of the inclusion of profile data into LogAnswer with numerous small test runs using different profiles and question sets. These tests were mostly used to determine the optimal configuration for my approach. The type of profile which should be used (i.e. the relevant classes of the PIMO ontology) had to be chosen. The best way to match the answer candidates with the profile data had to be identified. In the same step the best way to utilize the DBpedia ontology had to be figured out. Finally the configuration of the matching function had to be evaluated. Though to test the effect of the final configuration a detailed validation was necessary.

4.1 Evalutation Criteria

To decide if the use of profile information could lead to an improvement of the candidate ranking for a particular question, I looked at the rank of the matched candidates and compared it to the rank of the candiates which contained a correct answer. Only questions where at least one answer candidate matched the profile were regarded in detail since otherwise the ranking is not affected at all. If at least one candidate matched and the matched candidates did not contain correct answers a potential for improvement could be ruled out. In this case I looked at the highest ranked candidate containing a correct answer and checked if its rank was higher than the rank of any of the matched candidates. If this was the case it could lead to a worsening of the results since the wrong answer candidate could potentially get ranked higher than the correct answer candidate because it matched the profile. If all correct answer candidates had a lower ranking than the matched (but incorrect) candidates or if no correct candidates existed the use of the profile would have no effect. An improvement was only possible if at least one of the matched candidates contained a correct answer. I regarded a question as potentially improvable if no other correct answer candidate with a higher ranking existed and the matched (and correct) answer candidate was not already ranked first. If in this case other matched candidates not containing a correct answer existed, the potential negative effect was ignored since the goal to rank candidates with correct answers as high as possible was achieved nevertheless. Another critical aspect I looked at was the fact that the top 5 answers are the ones actually presented to the user in the end. While the top 5 answer candidates are not of necessity the same as the top 5 answers there is certainly a correlation. Therefore it was relevant if a change of the ranking causes a correct answer to get into the top 5 or if it caused it to be ranked lower. Some examples where this was relevant are detailed in chapter 4.3.

As each answer candidate had to be examined manually to evaluate if

it offers a correct answer to the corresponding question it was not feasible to regard all 200 candidates for each question. Therefore only the top 20 answer candidates for each question were regarded. As the ranking function already offers relative good results it was unlikely that a significant amount of correct answers were lost through this limitation (this assumption is backed by the test results).

4.2 Test Sets

For the thorough evaluation of my approach two different sets of questions were used. The questions for both test sets were chosen from a list of 8000 questions which were entered into LogAnswer over time. The first set consists of 200 randomly chosen questions which is why I will refer to it as "random set" from here on. This set was chosen to evaluate the effect of my approach on questions which do not necessarily have any correlation with the used profile. This also reflects the idea that a user will not only ask questions targeting domains which are covered by his profile. It was also suggested that for some questions the usage of a profile might improve the results in general. For example open questions where no single correct answer exists are well suited for the use of profile information and might therefore be improvable in general.

The questions of the second test set had to belong to knowledge domains which are reflected by a corresponding profile since the basic assumption of my approach is that a user will mostly ask questions related to his fields of interest. To have a single clear cut criterion for choosing the questions for this test set I decided to take all questions from a single knowledge domain: sports. So the second test set consists of 200 questions (also chosen from the list of 8000 LogAnswer questions) which are more or less related to sports and I will therefore refer to it as the "sport set".

I used three different profiles for the evaluation. The first one is my own profile which is directly extracted from my NEPOMUK system. The following list which forms the profile includes all instances of the PIMO classes "Topic", "Organization" and "Location":

> Brettspiele, Informatik, Japan, Karlsruhe, Koblenz, Künstliche Intelligenz, Kyoto, Magic The Gathering, Rollenspiel, Schottland, Tischtennis, Wizards of the coast

The other two profiles were specifically created for the test runs with the sport set. The first one consists of only one keyword: "Sport". This profile reflects the criterion which was used to select the 200 sport questions. Since "Sport" is a very generic domain there was the possibility of it matching with many of the answer candidates which would distort the test results. Therefore I put this particular keyword in a separate profile to test how many answer candidates would actually match with it. Accordingly the

last profile I used contains keywords which represent more specific topics from the sports domain:

> Bundesliga, DTM, FC Schalke 04, Ferrari, Formel 1, Fußball, Leichtathletik, Olympische Spiele, Snooker

4.3 Results and Evaluation

4.3.1 Statistics

The first test run was carried out with the random set and my own user profile. In total 3719 answer candidates for the 200 questions were regarded (for each question the top 20 at most). 53 of these candidates matched with the profile. The matching candidates were divided on 29 questions which means that over 85% of the questions would not be affected at all by the use of profile information. So as a first conclusion I noted that if the other results showed that profiles should be used as a hard filter, one would have to implement it in a way that the filter is only applied for questions, where at least one candidate matches with the profile. Of the 29 questions where this is the case only one offered the potential for improvement based on the use of the profile. This question was also the only open question among the 29. Details about this example question and the potential upsides of my approach when dealing with open questions can be found in the next chapter (4.3.2). For all other questions using the profile would either change nothing or even lead to a worsening of the results. So these results indicated that my approach is not well suited for questions which do not correlate with the used profile.

In the other two test runs I conducted I used the sport set with the two sport profiles. For the sport set 3639 answer candidates had to be regarded (again for each question the top 20 at most). 140 of these matched with the keyword "Sport" and 481 matched with the profile containing the more specific topics. The 140 candidates matching "Sport" were divided over 61 questions. If the ranking process for these 61 questions was modified by just looking at matches with "Sport" 5 would potentially improve and 15 potentially worsen. The 481 candidates which matched with the specific keywords were divided over 116 questions of which 9 would potentially improve and 30 potentially worsen after integrating the profile information into the ranking. Combining the two profiles results in 533 candidates being matched which means that 88 candidates matched with both profiles. In total these 533 candidates are divided over 129 questions. The inclusion of profiles into the ranking would cause an improvement in 10 cases and a worsening in 33 cases. These results indicate that enhancing the ranking of answer candidates with profile information is not recommendable. However to understand why these results emerged and what could be done to improve the approach, one has to look at the results in detail. Therefore I

present some selected examples which showcase the observations I made during the evaluation of the test runs.

4.3.2 Positive Examples

Open questions for which no one single correct answer exists are one example for positive effects of the use of profile information. If several answers can be regarded as correct it is likely that some of them are more relevant for the questioner than others or that the question was asked with a specific answer in mind. So an answer which might be regarded as correct by one user could be regarded as wrong by others. Now if the user's profile can be used to select the answers that are actually relevant for the questioner this would offer a clear improvement to the answering system. Of course this is only true if the used profile is somehow related to the question and hence at least one of the answer candidates matches. For example the random set contains some open questions where none of the answer candidates matched with the profile during the test runs. However it also contains one question which showcases the potential for improvement when dealing with open questions: "Welche Schlösser am Rhein sind Dir bekannt?"[12]. The following text passage was extracted as one possible answer candidate: "Das Schloss Stolzenfels steht über dem nach ihm benannten Koblenzer Stadtteil Stolzenfels auf der linken Seite des Rheins, gegenüber der Lahnmündung."[13] With "Schloss Stolzenfels" this text passage obviously offers a correct answer to the question. The answer candidate was extracted from the Wikipedia article about the castle which is related to the following categories by the `Subject` relation in the DBpedia ontology:

```
Resource:Kategorie:Erbaut_im_13._Jahrhundert
Resource:Kategorie:Architektur_(Preußen)
Resource:Kategorie:Erbaut_in_den_1840er_Jahren
Resource:Kategorie:Kulturdenkmal_in_Koblenz
Resource:Kategorie:
   Kulturlandschaft_Oberes_Mittelrheintal
Resource:Kategorie:Geschütztes_Kulturgut_in_Koblenz
Resource:Kategorie:
   Neugotisches_Bauwerk_in_Rheinland-Pfalz
Resource:Kategorie:Schloss_am_Mittelrhein
Resource:Kategorie:Schloss_in_Koblenz
Resource:Kategorie:Museum_in_Koblenz
```

[12] Which castles located at the Rhine do you know?

[13] The castle "Schloss Stolzenfels" is located above the Stolzenfels district of Coblence (which is named after the castle) at the left side of the rhine, across the mouth of the Lahn.

Some of these categories denote the location of the castle: "Koblenz". As I study at the University of Koblenz my profile contains the keyword "Koblenz" and therefore matched with this answer candidate. Since I am (for whatever reason) interested in the city of Coblence and I asked a question about castles at the Rhine it is very likely that I am also interested in castles located near Coblence. In another fictive scenario a person might have included Coblence in her profile because she has a meeting there. Therefore she might be especially interested in castles around Coblence when asking that question. One could argue that a questioner with this intention would phrase the question differently (for example: "Welche Schlösser in der Nähe von Koblenz sind Dir bekannt?"[14]) But in the mentioned scenario the questioner might not even have the intention to find castles near Coblence when posing the question about castles at the Rhine. Maybe she forgot about the meeting or did not think about it at that particular moment and therefore she could still be interested in castles near a location where she will stay. This answer candidate was ranked 4th which means that it is very likely that the resulting answer "Schloss Stolzenfels" would be ranked in the top 5 too (given that LogAnswer manages to extract the answer from the passage in the reasoning step). This means that in this case the use of profile information does not offer an improvement in the sense that a correct answer was found which was not found before, since the answer would have been presented to the user anyway. However it is still an improvement if the correct answer is ranked higher since the top ranked answer should ideally also be the best. For this example in particular it is to note though that another answer candidate which also contained a correct answer was not matched with the profile. This candidate was extracted from the Wikipedia article "Rheinhöhenweg" and contains this text passage: "Am Rheinhöhenweg liegen zahlreiche Burgen und Schlösser wie die Godesburg, Burg Lahneck, Schloss Stolzenfels, Burg Rheineck, Burg Rheinstein, Drachenburg und viele andere."[15] This candidate also contains "Schloss Stolzenfels" as an answer and additionally a whole list of castles located at the Rhine. While this candidate was only ranked 5th it is fair to say that this is a perfect answer to the question. However it was not matched to the profile since the DBpedia only contains the following categories as related subjects for the article "Rheinhöhenweg" and the title itself did not match either:

```
ResourceKategorie:Rhein
Resource:Kategorie:Wanderweg_(Hessen)
Resource:Kategorie:Wanderweg_(Rheinland-Pfalz)
```

[14] Which castles located near Coblence do you know?

[15] Numerous castles like the "Godesburg", "Burg Lahneck", "Schloss Stolzenfels", "Burg Rheineck", "Burg Rheineck", "Burg Rheinstein", "Drachenburg" and many others lie at the "Rheinhöhenweg".

```
Resource:Kategorie:Wanderweg_(Nordrhein-Westfalen)
```

While this answer candidate should clearly be ranked very high (likely it should be ranked first) one could argue that a user who is particularly interested in castles around Coblence expects an answer containing only those castles. Therefore an answer containing a list of castles which has to be sorted out by the user afterwards may be inferior to one that contains just the name of one castle which is near Coblence. Though again in this case the questioner might have phrased the question differently. But if we look again at a fictive example where a user is interested in hiking (profile contains keyword "Wandern") we see that this candidate would match. It is also very likely that a hiker would ask a question like this when searching for interesting travel destinations. In this scenario the use of profile information would improve the answering process by offering the perfect answer to the user.

Another type of questions for which the use of profile information is suited well are questions which contain homonyms or other ambivalent phrases. Such an ambiguity issue often makes it difficult to determine the domain which is targeted by the question. If one of the possible meanings of the question belongs to a domain which the user is interested in, it is very likely that he had this meaning in mind when asking the question. Now if this domain can be identified using the keywords from the user's profile, the corresponding answer candidates can be ranked higher to solve the ambiguity problem. Of course it is always possible that the user asks a question about something which is outside of his usual domain of interest in which case the use of profile information could lead to worse results. While this is a general problem of my approach it is less of an issue when dealing with ambiguity since it is unlikely that a user will often ask questions targeting topics with the exact opposite meaning than a topic contained in the user's profile.

A typical example which came up several times during my test runs where questions regarding persons whose name is not unique in the knowledge base. For example for the question "Wo wurde Ben Johnson geboren?"[16] answer candidates regarding the athlete as well as candidates regarding the actor were found by the LogAnswer system. This question was part of the sport set and was therefore tested with the detailed sport profile which contains the keyword "Leichtathletik"[17]. So in this case the goal is to find the birthplace of the athlete Ben Johnson because the profile clearly indicates that this is the domain the questioner is interested in. The only text passage which contains this information was ranked second and it did match with the profile. So if the profile information was included in

[16] Where was Ben Johnson born?
[17] athletics

the ranking the answer would likely be ranked first which is an obvious improvement. Regarding the fact that the first and third ranked answer candidates refer to the actor Ben Johnson, this question also offers a good example for the utilization of profiles as a filter. While the third ranked candidate also contains a technically correct answer to the question, it is very likely not relevant for the questioner. If both answers are displayed, the user would have to look at the text passages from which the answers were taken to decide which is the one he actually asked for. However if this answer was dismissed because it did not match the profile, only one correct answer would be presented to the questioner which is the optimal case.

Another typical example are inaccurate questions like for example: "Wie heißt der deutsche Nationaltrainer?"[18]. This question gives no indication about which sport it refers to. Again the user profile can be used to "guess" which domain the questioner had in mind. In the test run with the sport set one answer candidate for this question which was ranked 4th matched with the detailed sports profile. The text passage contained the name of the German national soccer team coach "Joachim Löw" and it matched with the keyword "Fußball". Assuming the question was targeting this information the use of the profile would again lead to an improvement. If a user was generally interested in information about the soccer domain the utilization of the profile as a filter would again make sense. In this case only the one candidate which contains the relevant information would pass the filter.

4.3.3 Negative Examples

There are many scenarios where the usage of user profiles can lead to a worsening of the answering process. Some of the most obvious ones (like for example users asking questions about domains which are not reflected by their profile) were already mentioned in this paper. However in this chapter I will present some examples where the usage of profile information should in theory be practicable or where the reason for the worsening is not obvious.

In chapter 3.2 I talked about special disadvantages when dealing with locations (i.e. names of cities or countries) in the user profiles. The following example showcases these problems: "Wie hoch ist der Fernsehturm in Stuttgart?"[19]. For this question one answer candidate extracted from the article "Fernmeldeturm_Koblenz" matched with my profile (keyword "Koblenz"). While this example showcases a general problem of my approach which is that the profile may accidentally match with wrong answer

[18] What is the name of the German national coach?
[19] How tall is the television tower in Stuttgart?

candidates the test runs showed that this problem occurs especially often when dealing with names of cities or countries. The reason for this is that statements about locations are often phrased in a similar way. Therefore the LogAnswer system will likely select pretty similar answer candidates which just differ in the name of the location they refer to. As in this example almost all the answer candidates contained information about a tower.

At first glance the next example question, taken from the sport set, seems to be perfectly suited for the use of profile information: "Welche Mannschaftssportarten kennst du?"[20]. It is an open question with many possible correct answers and the profile could in theory be used to provide the most fitting answers to the user. On top of this the sports profile which was used for the test run fits the domain of the question which should guarantee some positive results. However only two answer candidates matched with the profile and both did not contain a correct answer. The first one was taken from the Wikipedia article "Eckstoß"[21] and contains the following text passage: "Auch in anderen Mannschaftssportarten existiert der Eckstoß."[22]. It should be mentioned that in the next sentence of the article the team sports which this text passage refers to are named (soccer and fieldhockey). So if the LogAnswer system was able to corelate information across the boundaries of one sentence this text passage could deliver a correct answer. But since this is not possible at the time of writing this paper it has to be ignored in the evaluation. This candidate matched with the keyword "Fußball" from the profile because the article is part of the category "Fußballregel". The second matched candidate was taken from the article "Sønderjysk Elitesport" and contains the following text passage: "Der Sønderjysk Elitesport (SønderjyskE) ist ein dänischer Sportverein, der in mehreren Mannschaftssportarten aktiv und erfolgreich ist.". This candidate also matched with the keyword "Fußball" because the article is part of the category "Dänischer_Fußballverein". These two candidates were ranked 5th and 12th. Eight answer candidates on the ranks 1,8,9,16,17,18,19 and 20 contained correct answers which means that the use of profile information would have lead to a worsening of the ranking in this case. The correct answer candidates were for example extracted from articles about hockey and polo which did not match with the profile. This example showcases a general problem regarding the use of profiles: Even if a profile technically fits the domain of the question, certain keywords might be needed to match the correct answer candidates which were extracted (e.g. hockey and polo). Even if the keywords contained in the profile might in theory match with potential answers to a question this does not help if these answers are not contained in the 200 extracted text passages.

[20] Which types of team sport do you know?

[21] corner kick

[22] The corner kick also exists in other team sports

The next example is again a question from the sport set: "Wie ist der aktuelle Weltrekord im 100m-Lauf?"[23]. While seven answer candidates for this question matched with the detailed sport profile none of those contained a correct answer. However the answer candidate which was ranked first contained a correct answer. This candidate was extracted from the Wikipedia article "August 2009" which contains a summary of important events and facts related to that month. Obviously this article did not match with the sports profiles and since it is not particularly related to the sports domain it certainly should not. There are many cases where the correct answer to a question can be found in an article which is not directly related to the domain of the question. The reason for this is that the knowledge contained in the Wikipedia is not strictly sorted by topic. While it is structured in different articles there are many articles containing information from various domains and each article also contains information which does not directly refer to its main topic. This means not only that answers for questions can sometimes be found in unexpected articles but also that relevant information for a user can sometimes be found in articles which do not match with his profile.

In a few cases unexpected results which would potentially worsen the ranking occured because the matching function (see chapter 3.3.2) did not return the expected result. For example it may be expectable that the two strings "Fußball-Weltmeisterschaft 1982" and "Fußball" are regarded as a match. However the matching function only returned a value of 0.448 which did not pass the threshold of 0.5. Though removing the hyphen in the first string would cause the matching value to raise to 0.757 because then "Fußball" is regarded as a separate token which can directly be matched with the second string. Another example are the two strings "Olympische Sommerspiele 1980/Basketball" and "Olympische Spiele" which should be regarded as matching but the matching function returned 0.0 which indicates no match at all. These examples show a potential for improvement in the matching function. However dealing with this type of natural language issues is a general task of the question answering field and therefore not directly related to my approach.

4.3.4 Problems with the DBpedia

One major problem I discovered during the test runs are problems with the DBpedia ontology which are the cause of some of the bad results in my work. The related subjects for the Wikipedia articles are not used consistently. The following examples showcase this flaw and the effect it has on my work. The first example I will look at is the question: "Wer ist Bastian Schweinsteiger?"[24]. The text passage which presumably contains the best

[23] What is the current world record in 100-metre dash?

[24] Who is Bastian Schweinsteiger?

answer to this question is taken from the Wikipedia article about Bastian Schweinsteiger. The DBpedia ontology relates the following subjects with this article:

```
Resource:Kategorie:Mann
Resource:Kategorie:
   Fußballnationalspieler_(Deutschland)
Resource:Kategorie:Deutscher_Meister_(Fußball)
Resource:Kategorie:Kategorie:Geboren_1984
```

This article does not match with the keyword "Sport", since this phrase neither appears in any of the categories nor in the title (the article has no redirects). Another answer candidate (taken from the article "TSV_1860_Rosenheim") which also contained a correct answer matched with "Sport" but this answer was not as precise. Though when I looked at another example question from the sport set and compared it to this one I recognized an inconsistency. The question "Wer ist Lukas Podolski?"[25] is a pretty similar question and in this case again the best answer candidate was taken from the main Wikipedia article about the person in question (i.e. Lukas Podolski). For my test run in particular both articles matched the user profile since the detailed sports profile I used also contains the keyword "Fußball. However looking at the related subjects of this article it is obvious that this time the candidate matches with the keyword "Sport":

```
Resource:Kategorie:Mann
Resource:Kategorie:Pole
Resource:Kategorie:Person_(Schlesien)
Resource:Kategorie:
   Fußballnationalspieler_(Deutschland)
Resource:Kategorie:Deutscher_Meister_(Fußball)
Resource:Kategorie:Kategorie:Geboren_1985
Resource:Kategorie:Sportler_(Köln)
```

All the categories listed for the article about Bastian Schweinsteiger have an equivalent in this list of categories but there are also some additional ones. While it is obvious that different persons also have different attributes, the inconsistency resulting from using some type of attribute only for some of the resources where it would be applicable, harms the usability of the ontology. In this example the resource "Kategorie:Sportler_(Köln)" identifies the related resource (i.e. Lukas Podolski) as a sportsman from Köln. The DBpedia contains many analogical resources referring to different cities (e.g. "Kategorie:Sportler_(Berlin)") which could all be used to

[25] Who is Lukas Podolski?

classify sportsmen. However since this type of categorie is not used consistently for all sportsmen it does not offer a reliable criteria. Other examples confirm this observation as for instance the article "UEFA Women's Champions League" is related to the category "Kategorie:Frauenfußball" while the article "UEFA Champions League" is only related to the category "Kategorie:UEFA_Champions_League". Of course this effect is not restricted to the sports domain. For example the article "Mont Blanc" can be matched with "Geographie" while the article "Mount Everest" can not. It is fair to say that my approach would have delivered better results if these inconsistencies did not exist.

5 Conclusion and Future Work

The raw numbers presented in chapter 4.3.1 suggest that the approach which I presented in this paper does not offer a potential for an improvement of the system. Some of the negative effects which are showcased in the examples can not be circumvented. For example the fact that relevant information for the user can sometimes be found in articles which do not match with his fields of interest is directly based on the structure of the Wikipedia. Since knowledge taken from sources in natural language will never be strictly sorted by topic this problem is unavoidable. Also the fact that the profile can accidentally match with wrong answer candidates can hardly be circumvented. However the positive examples show that the approach still has the potential for improvement. From the beginning of my work it was clear that the approach would be well suited for open questions and this was backed by the test results. The problem here is that most of the questions in the LogAnswer system and hence most of the questions in my test sets were questions targeting factual knowledge. Using profiles can often lead to a worsening of the results for this type of question. Thus in a different scenario where open questions are more frequent, the utilization of profile information could be very useful to filter relevant answers and thus improve the quality of the system. Another problem with the current LogAnswer application is that users will ask questions from various domains. The test results in my work showed that the use of profile information is not effective at all if the profile does not directly correlate with the asked questions. Here again in a different scenario, where each user will only ask questions about some specific domains which are explicitly represented by his profile, my approach would be of much better use. However in the end the conclusion must be that my approach is not suited for the common area of application in which LogAnswer is currently used.

My suggestions for future work are primarily related to the usage of the profile. In my approach I used a very simple type of profile consisting of just a list of keywords. Using a more detailed profile which could be extracted from a different source could potentially reduce the amount mistakenly matched answers. Another possibility could be to create a profile based on the last questions posed by a user and use this information as a heuristic. If a user asks multiple questions about the same or related topics, this approach could be used to identify the corresponding domain currently researched by that user. In my paper I also mentioned some possible enhancements for my approach. Reevaluating the results of my approach with a more consistent ontology and with an improved matching function could definitely lead to better results in the future.

References

[CB09] CHRISTIAN BIZER, Georgi Kobilarov-Sören Auer Christian Becker Richard Cyganiak Sebastian H. Jens Lehmann L. Jens Lehmann: DBpedia A Crystallization Point for the Web of Data. In: *Journal of Web Semantics: Science, Services and Agents on the World Wide Web Issue 7, Pages 154 - 165* (2009)

[DFW10] DAVID FERRUCCI, Jennifer Chu-Carroll James Fan David Gondek Aditya A. Kalyanpur Adam Lally J. William Murdock Eric Nyberg John Prager Nico S. Eric Brown B. Eric Brown ; WELTY, Chris: Building Watson: An Overview of the DeepQA Project. In: *AI Magazine* (2010). `http://www.stanford.edu/class/cs124/AIMagzine-DeepQA.pdf`

[FGHP08] FURBACH, Ulrich ; GLÖCKNER, Ingo ; HELBIG, Hermann ; PELZER, Björn: LogAnswer - A Deduction-Based Question Answering System. In: *IJCAR 2008 - 4th International Joint Conference on Automated Reasoning, Sydney, Australia, 10th - 15th August, 2008, Proceedings, to appear*, Springer, 2008 (Lecture Notes in Computer Science)

[FGHP10] FURBACH, Ulrich ; GLÖCKNER, Ingo ; HELBIG, Hermann ; PELZER, Björn: Logic-Based Question Answering. In: *KI - Künstliche Intelligenz* 24 (2010), Nr. 1, S. 51–55. `http://dx.doi.org/http://doi.acm.org/10.1007/s13218-010-0010-x`. – DOI http://doi.acm.org/10.1007/s13218–010–0010–x. – Special Issue on Automated Deduction

[GHM+07] GROZA, Tudor ; HANDSCHUH, Siegfried ; MOELLER, Knud ; GRIMNES, Gunnar ; SAUERMANN, Leo ; MINACK, Enrico ; MESNAGE, Cedric ; JAZAYERI, Mehdi ; REIF, Gerald ; GUDJONSDOTTIR, Rosa: The NEPOMUK Project - On the way to the Social Semantic Desktop. In: PELLEGRINI, Tassilo (Hrsg.) ; SCHAFFERT, Sebastian (Hrsg.): *Proceedings of I-Semantics' 07*, JUCS, 2007, pp. 201-211

[Glö07] GLÖCKNER, Ingo: University of Hagen at QA@CLEF 2007: Answer validation exercise. In: *Results of the CLEF 2007 Cross-Language System Evaluation Campaign, Working Notes for the CLEF 2007 Workshop. Budapest, Hungary*, 2007

[Glö08] GLÖCKNER, Ingo: Towards logic-based question answering under time constraints. In: *Proceedings of the IAENG International Conference on Artificial Intelligence and Applications (ICAIA-08), S. 13-18, Hong Kong*, 2008

[Glö09] GLÖCKNER, Ingo: Finding Answer Passages with Rank Optimizing Decision Trees. In: *Proceedings of the Eighth International Conference on Machine Learning and Applications (ICMLA-09), IEEE Press, 2009, pp. 208-214*, 2009

[GP08] GLÖCKNER, Ingo ; PELZER, Björn: Exploring Robustness Enhancements for Logic-Based Passage Filtering. In: *Proceedings of the 12th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES-08), Zagreb*, September 2008

[Har02] HARTRUMPF, Sven: *Hybrid Disambiguation in Natural Language Analysis*. Hagen, Germany, FernUniversität Hagen, Fachbereich Informatik, Diss., Juni 2002

[Hel06] HELBIG, Hermann: *Knowledge Representation and the Semantics of Natural Language*. Springer, 2006

[PW07] PELZER, Björn ; WERNHARD, Christoph: System Description: E-KRHyper. In: PFENNIG, Frank (Hrsg.): *Automated Deduction: CADE-21* Bd. 4603, Springer, 2007 (LNAI), 503–513

[RLJL97] ROGERS, Seth ; LANGLEY, Pat ; JOHNSON, Bryan ; LIU, Annabel: Personalization of the Automotive Information Environment. In: *Proceedings of the workshop on Machine Learning in the real world; Methodological Aspects and Implications*, 1997, S. 28–33

[Sau07] SAUERMANN, Leo: Semantic Web am Desktop. In: *Entwickler Magazin* 2008 (2007), 12, Nr. 1, 119-122. `http://entwickler-magazin.de/zonen/magazine/psecom,id,17,ausgabe,234,p,0.html`. – Aimed for Popular Science and Developers

[SED07] SAUERMANN, Leo ; ELST, Ludger van ; DENGEL, Andreas: PIMO - a Framework for Representing Personal Information Models. In: PELLEGRINI, Tassilo (Hrsg.) ; SCHAFFERT, Sebastian (Hrsg.): *Proceedings of I-Semantics' 07*, JUCS, 2007, pp. 270-277

[WBW+02] WAGNER, M. ; BALKE, W.-T. ; WAGNER, Matthias ; BALKE, Wolf tilo ; HIRSCHFELD, R. ; KELLERER, W.: *A Roadmap to Advanced Personalization of Mobile Services*. 2002

6 Appendix

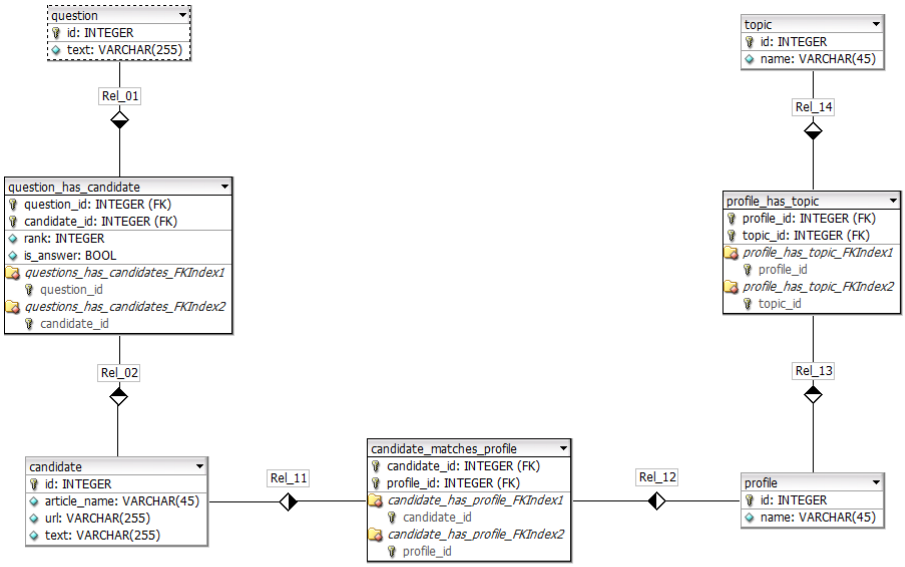

Figure 5: Database architecture for the implementation of the test runs

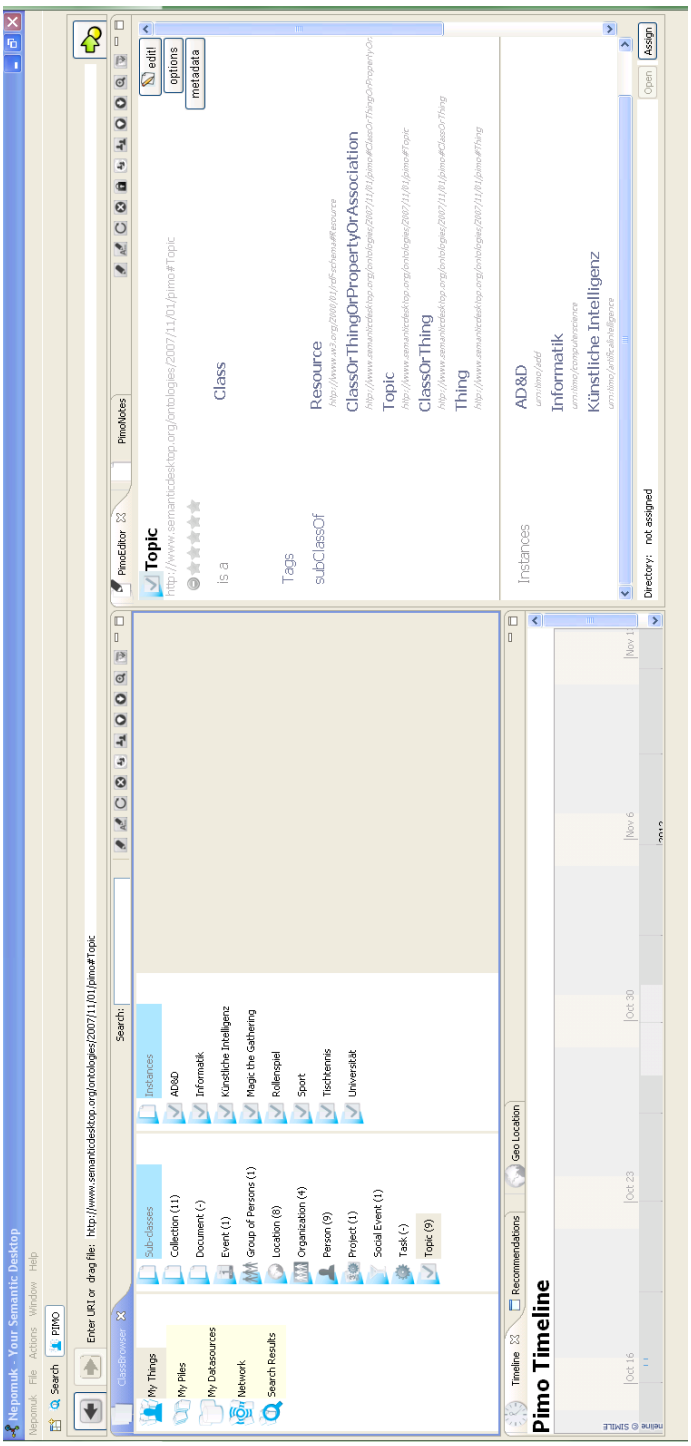

Figure 6: Screenshot of the NEPOMUK Semantic Desktop PIMO perspective