"ASKWIKI": SHALLOW SEMANTIC PROCESSING TO QUERY WIKIPEDIA

Felix Burkhardt and Jianshen Zhou

Deutsche Telekom Laboratories, Berlin, Germany

[Felix.Burkhardt | Jianshen.Zhou] @telekom.de

ABSTRACT

We describe an application to query Wikipedia with a voice interface on a mobile device, i.e. smart phone or tablet computer. The aim was to develop a so-called App that installs easily on an android phone and does not need large vocabularies. It can be used to either answer questions directly, if the information is contained in a table or matches some keyword syntax (like birth place), or get access to an article's sub chapter. An evaluation based on 25 test users showed the feasibility of the approach.

Index Terms— Wikipedia semantic modeling natural language understanding

1. INTRODUCTION

This article describes a so-called App named "AskWiki"; an application to query the German Wikipedia[1] with a voice interface in a mobile context, i.e. with a small mobile hand held device such as an Android based smart phone or tablet computer.

Besides providing for a textual interface, the in- and output can be given acoustically by using the build-in automatic Speech recognition (ASR) and Text-to-Speech synthesis (TTS) capabilities provided by the Android operating system. It is possible to query the system with natural language, i.e. words that don't carry semantic information are detected, although a query consisting only of the target keywords gets handled much faster by the system because the search space is smaller.

The idea for this application was motivated by the wish to access Wikipedia knowledge hands- and eyes-free, i.e. input as well as output should be given acoustically. A further constraint was that it should run completely on a mobile hand held device as a small, easy to download application without the need to operate a server. Application scenarios for this approach encompass, besides offering tourist information, the general advantage of accessing lexical information anytime anywhere, for example to submit queries for other movies featuring the main actor on the way home from a cinema visit.

We therefore looked for a solution that omits the use of large vocabularies as much as possible. The only vocabularies used in the current version are a short list of stop words and a set of synonym words and related concepts. No ontological knowledge is used, but the answer is simply found by repetitive trials to retrieve an article from Wikipedia based on the search words.

[1] http://de.wikipedia.org

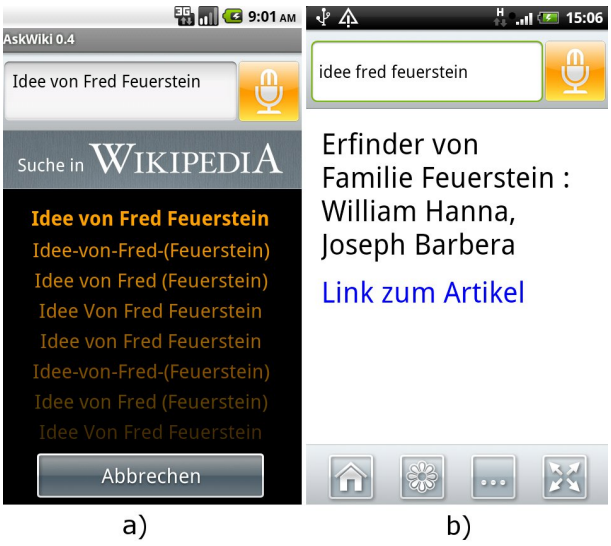

Fig. 1. *Screen shots of the graphical user interface. a) while searching, b) displaying the result*

This article is structured in the following way. Section 2 gives a short review on existing work from the literature. The following section 3 introduces AskWiki, explains the general approach and describes the algorithm to query Wikipedia. In Section 4, the semantic model of a natural language query mechanism is explained. Section 5 introduces the structure of an article. Section 7 describes the results gained from a user field test, Finally, in Section 8 we draw conclusions and discuss some possible future work.

2. LITERATURE REVIEW

The Wikipedia open domain lexicon was created by Jimbo Wales and Larry Sanger in 2001, its origins date from 1995. It is a collaboratively created encyclopedia based on the Wiki

content management framework introduced by Ward Cunningham in 1995. The Wiki framework used by Wikipedia is called MediaWiki, a Wiki software especially designed by the Wikipedia community.

In order to extract structured information from Wikipedia content, three approaches are possible.

- Add semantic tags directly to the Wikipedia content, e.g. [1].
- Extract structured information from Wikipedia and store as semantic data, e.g. formatted in a RDF triple store [2].
- Interface Wikipedia directly and gather information "on the fly", as done by the approach described in this article or in [3] and [4].

Approaches to enrich Wiki content with semantic annotation represented as RDF triples exist since longer time. In the case of [5], not only a syntax to describe RDF triples is described, but a Wiki browser including ontology visualization gets introduced. This allows queries like "Give me all movies from the 60ies with Italian directors".

For the interpretation of queries in a Question Answering application, in a first step the words must be processed by a natural language interpreting module. Such frameworks require large vocabularies and have a large footprint with respect to hardware resources and computing power. [4] use Google search and WordNet as additional information sources. In [1], categories, typed links and attributes are used to model a semantic structure between the Wikipedia articles.

DBPedia [2] is an international project to extract structured information from Wikipedia and to make this information available on the Web. Most of the articles described in DBPedia are classified in a consistent ontology. A query interface is realized in SPARQL[3] and can be accessed by several SPARQL engines. Also, several visual interfaces exist to the DBPedia data e.g. GFacet [4] or Fluidops[5]. The data also is used in numerous other databases as part of the linked data project[6].

Accessing DBPedia data in a mobile context has been described by [6]. Based on the current location of the user, nearby things described in DBPedia are presented and can be used as starting points for further exploration, e.g. in a tourist guide scenario. In contrast to this, the work described here simply concentrates on presenting the user with the most important information extracted from Wikipedia because displays are usually small in mobile gadgets, but does not take the current location into consideration.

[2]http://dbpedia.org/
[3]http://www.w3.org/TR/rdf-sparql-query/
[4]http://www.visualdataweb.org/gfacet
[5]http://iwb.fluidops.com
[6]http://linkeddata.org/

Virtually all approaches to add semantics to the Wikipedia content use RDF, the Resource Description Framework, as an underlying syntax to express semantic triples of subject, predicate and object.

Unlike [1], the approach described here does not require a modification to the Wikipedia content, but operates directly on the article sources. Semantic links are not yet interpreted, i.e. questions like "What is the capital of England" can only be answered if the answer appears in the article about England, but not via a link referral.

Some authors used the Wikipedia in Question Answering systems to tackle the TREC and CLEF challenges, e.g. [3] or [4], although they did not use the content to answer the questions directly, but to select the most probable answer from a set of possible candidates by comparison with the Wikipedia content.

3. THE "ASKWIKI" APP

Wikipedia information as such is not stored in a formally structured or machine readable content, but authors write texts that follow loose guidelines and suggestions. Although Wikipedia articles are not written with a formal syntax, they contain structured information by info boxes which use a template mechanism, images depicting the articles topic, categorization of the article, links to external web pages, intra-wiki links to other articles and inter-language links to articles about the same topic in different language.

The problem one encounters when extracting this information is that there is no formal syntax Wikipedia authors must follow when editing such information. We tackled this by focusing on the values in tables on the one hand and on the importance of the first sentence of a sub chapter on the other hand.

The aim of the AskWiki application is partly to provide question answering and partly to facilitate search in a large Wikipedia article with a small and limited device by retrieving the crucial information as dense as possible.

A screen shot of the GUI is shown in Figure 1. After a search query was entered by the user, the current search trials to Wikipedia get displayed (part a). The search can be interrupted if the N-Best results get noticeable worse.

When an article was found, either the value of the corresponding table cell or the first sentence of a sub chapter is presented as an answer (part b). By clicking the buttons at the bottom, a picture or more information can be viewed. Also a direct link to the Wikipedia article is provided.

In order to match the feature word of the query and the table entry or sub chapter title respectively, a list of synonyms and related concepts is given as a vocabulary which can be edited and extended by the user. Like this, the answer to the query *Where is Trondheim located?* results in the first sentence of the sub chapter about the geography of Trondheim.

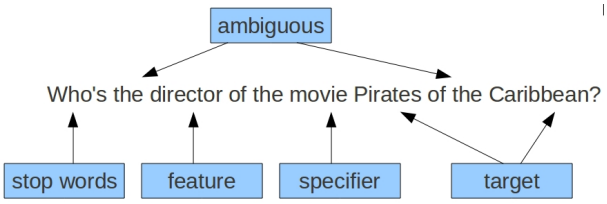

Fig. 2. *Tokenization of an input phrase.*

4. SEMANTIC MODEL

We model a query as consisting of tokens that might belong to up to five categories, see Figure 2 for an example.

- Stop words can be disregarded without losing information. They are identified by a vocabulary look up. From our experience, in German a short list of 20 to 50 entries containing the most frequently used words in questions is sufficient.

- Feature names can be used to query directly for a feature of the target article, e.g. the birth date of a person or the language spoken in a Country. Note that sub chapters are also modeled as features. This makes it possible to query directly for a sub chapter of an article, e.g. "*north wing castle Gottorf*".

- A specifier can be used to distinguish between ambiguous named entities, e.g. the name "Nashville" might mean the town or the movie by Robert Altman, for common names like in German "Klaus Müller" there are many different persons described in Wikipedia who might be distinguished by their profession.

- The target must exactly match the title of a Wikipedia article, even in capitalization. In contrast to specifier and feature, the target might be a multi-word sequence.

- Ambiguous words might be stop words but also might be part of a named entity. Examples are the words "von" (German for "of" and possibly part of a named entity) or "the". They are included in the search process in a later stage.

In Figure 2, an example of a query consisting of all possible parts is depicted. Note that the token "the" is ambiguous as it plays two roles, the first one is a stop word and must be ignored, the second is part of the article title and must be preserved and set in lowercase.

The grammar for a query is given in EBNF (extended Backus-Naur form) in the following lines.

$$
\begin{aligned}
< query > &:=< ftFrntQry > | < ftBckQry >; \\
< ftFrntQry > &:= [< feature >] < spcfdTarget >; \\
< ftBckQry > &:=< spcfdTarget > [< feature >]; \\
< spcfdTarget > &:= [< spcfr >] < target >; \\
< target > &:=< wrdseq > *; \\
< feature > &:=< wrdseq >; \\
< spcfr > &:=< wrdseq >; \\
< wrdseq > &:= [< stpwrds > *]token[< stpwrds > *]; \\
< stpwrds > &:=< stpword > | < ambig - word >; \\
< stpword > &:= \text{“}who\text{”}|\text{“}where\text{”}|\text{“}and\text{”}|\text{“}or\text{”}...; \\
< ambig - word > &:= \text{“}the\text{”}|\text{“}of\text{”}...; \\
< token > &:= [A - Za - z]*! =< stpwrds >;
\end{aligned}
$$

As described in Section 3, the program tries to find a match for a target article in Wikipedia for a given query by removing some words and changing the capitalization of other words. Because each trial results in a network connection attempt to Wikipedia, we can not try too many variants because the reaction time of the program is a critical issue in a real world scenario.

The feature can be in front of the target words ("*when was Einstein born?*") as well as behind ("*birthday of Einstein?*"). The specifier must be in front of the target ("*the movie Nashville?*"), i.e. "*Nashville the movie?*" would not be recognized.

In contrast to specifier and feature, the target may consist of several words ("*Pirates of the Caribbean*"). Stop words may occur everywhere in the query, they will simply be deleted.

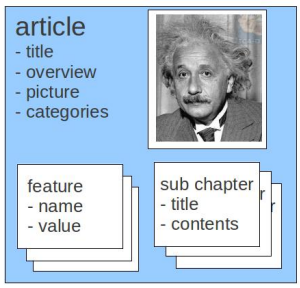

Fig. 3. *Model of an article.*

5. DATA MODEL AND EXTRACTION

In Figure 3, the data model of the implementation is depicted. An article has a title, a description, optionally a picture, a set of categories and a set of features as well as sub chapters, each

with a title and content. The description gets loaded by parsing all text before the first chapter headline. A preprocessing filter executing a set of pattern matching rules is used to filter out all sorts of Wikipedia Markup Language, HTML and other markup that can not be synthesized by a speech synthesizer in a meaningful way.

There is no formal syntax to mark an articles about persons, but if birth- and/or death dates and places are detected they get extracted by the pattern matching algorithm. Additionally, for all tables, the text in the first column cell for each row is taken as a feature name and all remaining text in the row as the value.

All content-lines starting with a section heading mark the beginning of a sub chapter. Parts with a header but missing content are treated as super chapters and their headers get appended to the following chapter headers. Take for example the sub chapter "*north wing*" for "*castle Gottorf*" with the super header "*buildings*", which gets saved as "*buildings north wing*" during the parse.

Similarly the chapter hierarchy is still visible although it is not really preserved. Because parts get treated in the following just like features, they can be used to search directly for information described in a sub chapter of a Wikipedia article. In fact, this is the only way to access this information from within the AskWiki application. The feature matching algorithm matches whole words, i.e. in order to get the chapter "*passenger elevators*" of the article about the Eiffel tower it is sufficient to say either "*passenger*" or "*elevators*".

6. SEARCH STRATEGY

The Google speech recognizer returns only lowercase tokens. Because the Wikipedia API[7] is very strict when matching article titles, all sorts of combinations regarding capitalization and hyphenation have to be tried out until an article gets returned, the article title is set as a title / specifier and the remaining word (at most the first or last word is used as a feature, see section 4) as feature description.

The following searches are run until an article is found for each of the N-Best recognition results.

1. search original query
2. search variants of original query
3. search variants of query without stop words
4. search variants while first word is feature
5. search variants while last word is feature
6. search variants while first word is feature keeping ambiguous word
7. search variants while last word is feature keeping ambiguous words
8. return no result

Each search consists of the following search variants.

1. try capitalization. e.g. "*Albert Einstein*"
2. try first word as specifier, e.g. "*Physicist Einstein*"
3. try to add hyphens, e.g. "*rhein-neckar-zeitung*"
4. try uppercase, e.g. "*BMW*"
5. try named entity capitalization, e.g. "*Werner von Siemens*"

Some of the variants exclude each other, for example all uppercase gets only tried if the string contains only one word whereas hyphenation does only make sense with multiple words. We carry out the searches in descending order of complexity and probability, so that less complex and more probable queries get answered faster than complex and rare questions.

Each query might lead to up about 40 Wikipedia API calls per N-best result and the user might have to wait for the answer, depending on network performance and query complexity, for up to approximately 10 seconds which is about the longest time acceptable for such an application. We did not include spelling variants or stemmers because this would prolong the execution time excessively.

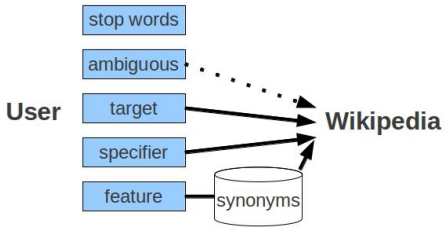

Fig. 4. *Some parts of the user's query get filtered.*

Up until now we implemented only one automatically inferred information because it appeared often in the test queries. The question "how old is X" can be answered for persons, cities, companies or countries and the like because a feature "age" is automatically added if features like "born", "founded" or "built" are encountered.

7. EVALUATION

During the development of AskWiki we conducted a field trial in order to test the usefulness of the App. 25 users selected randomly from the Berlin population were paid to use AskWiki for one week in their daily life and report their experiences. As part of the reports, they send us log files of

[7] http://en.wikipedia.org/w/api.php

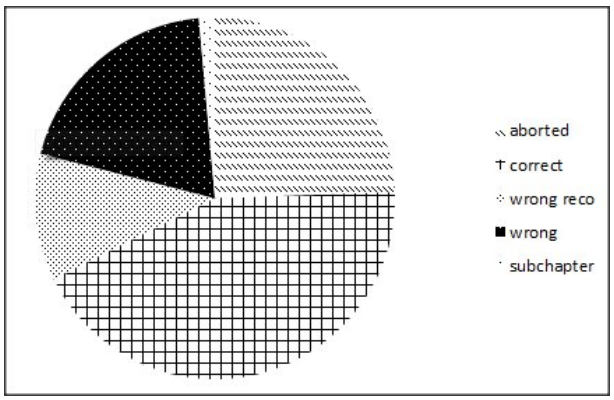

Fig. 5. *Results of the evaluation.*

the App containing the queries that were recognized by the Google speech recognizer (ASR). Figure 5 gives an overview of the outcomes. We collected 1052 queries and labeled them manually with the following categories.

- The search was aborted (24,4 %) by the user, which means probably that the ASR result was wrong.
- In most of the cases (43,16 %), the answer is correct when the App gets tuned, i.e. sometimes an acronym or synonym had to be added to the vocabulary.
- In 11.4 % of the cases, the ASR result was obviously wrong, i.e. no recognizable question could be detected.
- In 19.58 %, the answer was wrong or could not be detected.
- Only 1.43 % were queries for sub chapters.

This means, given a correct speech recognition result, the correct answer could be found in 68.79 % of the cases with a failure rate of 31.21 %.

At the time of writing this article, over 5176 people have downloaded AskWiki and 3764 of them kept the App on their device, which results in a retention rate of 72.72 %. 49 users evaluated the App with a star between 1 and 5 in the Android market and the mean value is 4.5 which is rather high.

We also tried the test queries of the 2003 CLEF campaign in German [7]. Of the 200 queries, only 48 were answered correctly, and most of them only after the question was reformulated to adapt to the telegraph style required by AskWiki, e.g. from *In which town was Emil Fischer born?* to *Birthplace Emil Fischer*. It shows that many of the questions involve more than one article to answer , e.g. *Which Elvis song was performed by Gilmore?*, while others deal with information specific to the US and thus not included in the German Wikipedia.

8. CONCLUSIONS AND OUTLOOK

We described an approach to retrieve Wikipedia information in a mobile context. It can be used to either answer questions directly, if the information is contained in a table or matches some keyword syntax (like birth place), or get access to an article's sub chapter.

An evaluation based on 25 test users as well as the comments in the Android market show that the approach gives indeed useful results to answer questions in the daily life.

The application can be extended in two directions: the synonym and stop word lists could be enlarged to cover a broader set of queries on the one hand, on the other the GUI could be much more complex to add for example more pictures, sub chapters or follow links directly.

9. REFERENCES

[1] Max Völkel, Markus Krötzsch, Denny Vrandečić, Heiko Haller, and Rudi Studer, "Semantic wikipedia," in *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23-26, 2006*, MAY 2006.

[2] 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 Semant.*, vol. 7, pp. 154–165, September 2009.

[3] David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Mller, Maarten de Rijke, and Stefan Schlobach, "Using wikipedia at the trec qa track," in *Proceedings of TREC 2004*, 2004.

[4] David Buscaldi and Paolo Rosso, "Mining knowledge from wikipedia from the question answering task," *Proceedings of the 5th International Conference on Language Resources and Evaluation*, 2006.

[5] David Aumueller and Sren Auer, "Towards a semantic wiki experience – desktop integration and interactivity in wiksar," in *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference*, Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, Eds., Galway, Ireland, November 2005, pp. 212 – 217.

[6] C. Becker and C. Bizer, "DBpedia Mobile-A Location-Aware Semantic Web Client," *Proceedings of the Semantic Web Challenge*, 2008.

[7] B. Magnini, S. Romagnoli, A. Vallin, J. Herrera, A. Peas, V. Peinado, F. Verdejo, and M. de Rijke, "Creating the disequa corpus: a test set for multilingual question answering," *Working Notes for the CLEF 2003 Workshop*, 2003.