Voice Search in Mobile Applications with the Rootvole framework

Felix Burkhardt

Deutsche Telekom Laboratories, Berlin, Germany

[Felix.Burkhardt@telekom.de]

Abstract

We present several Android apps that deal with voice based access to internet content on mobile apps, namely AskWiki, AutoScout24 search and the TV-guide app. They highlight several aspects of voice search strategies to match user query with vocabularies based on controlled, semi-controlled and linked open databases.

Index Terms: voice search, mobile applications, query interpretation

1. Introduction

We present several Android apps that deal with voice based access to internet content, screenshots are shown in figure 1.

a) AskWiki [1] deploys a heuristic approach to answer questions on semi-structured data without explicit modeling of vocabularies, but simply by trials to match keywords spotted in the query against Wikipedia article names. b) In contrast, the AS24 app [2] uses a very controlled database, the AutoScout24 used car data, to enable voice search for second hand cars. The main focus here is to detect numerical restriction in the query ("*from 3 to 5 thousand euro*"). c) The TVGuide app [3] can be seen as a combination of the two aforementioned approaches, by on the one hand matching against vocabularies extracted from the electronic program guide (EPG) and on the other expanding the queries by Freebase and DBPedia lookups. It is described in a separate article in this proceedings [3]. All of these are driven by a common framework for voice search named "*Rootvole*" developed in our labs.

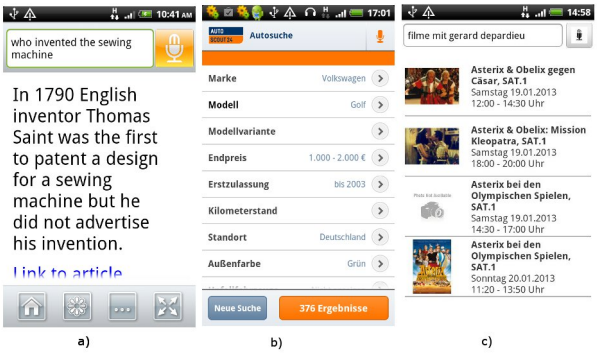

Figure 1: Screen shots of the app GUIs: a) AskWiki, b) AutoScout24 c) TV guide

Besides using different approaches towards vocabulary matching, the apps are differentiated by being either client-based a) or server-based b), i.e. the semantic processing takes place on the smartphone or on a server, both are supported by the Rootvole library. Figure 2 illustrates the architecture.

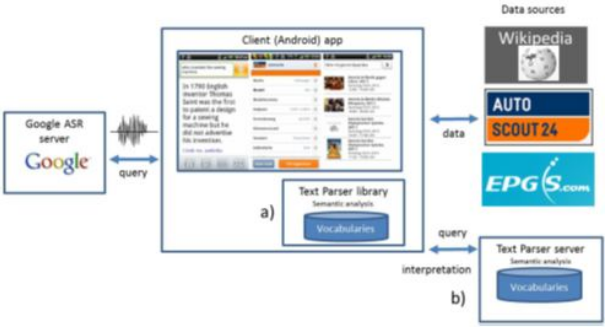

Figure 2: Two possibilities to access rootvole a) on the client, b) as a server implementation

For general processing of voice queries we developed a text parsing library (Rootvole, [2, 3]) that can be used to match text with semantic concepts. The algorithm was implemented in Java and can be described as a form of a parsing expression grammar, where we generate the expressions to be detected beforehand by regular expressions and store them in a vocabulary. Rootvole is planned to be published as open source software. Figure 4 shows the processing chain for the AutoScout24 App. Rootvole is designed especially to tag short queries with semantic concepts.

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 [5], categories, typed links and attributes are used to model a semantic structure between the Wikipedia articles. Some authors used Wikipedia in Question Answering systems to tackle the TREC and CLEF challenges, e.g. [6] 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. While we search with the AutoScout24 app in a structured database give an natural language, car ad-like expression, [7] describe a system that goes the other way round by extracting structured data from car ads.

2. The AskWiki app

The aim of the *AskWiki app* is partly to provide question answering and partly to facilitate search in a large Wikipedia[1] article with a small and limited device by presenting the crucial information as densely as possible. Wikipedia content as such

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

is not stored in a formally structured or machine readable form, but authors write texts that follow loose guidelines and suggestions. Still they contain structured information in info boxes which use a template mechanism, images depicting the article's topic, categorization of the article or subheadings.

As described in [1], several token combinations of the user's query are tested against the Wikipedia API, while allowing for additional "feature words" (figure 3). When an article is found, either the value of the table cell that matches the feature word or the first sentence of a corresponding subchapter is presented as an answer. A list of synonyms and related concepts is used to expand the feature words; it can be edited and extended by the user. For example, the answer to the query *Where is Trondheim located?* results in the first sentence of the subchapter about the geography of Trondheim.

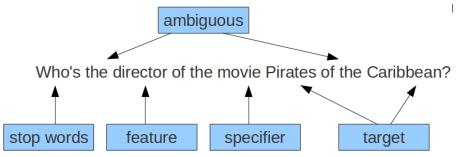

Figure 3: AskWiki: selection of keywords

AskWiki was introduced in the market early 2012 as, to our knowledge, the first question answering app in German, and has since been installed by more than 50.000 users. It has been ported to four additional languages (English, Czech, Dutch and Finnish), although informal tests show that it works best in German, the original target language of the approach. Today, the new developments of DBpedia and Wikipedia's Wikidata[2] would make a complete review of the approach worth while.

3. The AutoScout24 app

The *AutoScout24 app* [2] lets the user search for secondhand cars by choosing constraints from drop-down menus. With our enhancement, the user can now click a microphone button, say the search terms in natural language, and, after confirming the recognition results, the menu is filled with the selected values. The search can then be executed, or further selections can be made either by pointing or by voice.

An overview on the parser architecture is shown in figure 4. The input query is the result of the automatic speech recognizer (ASR). This means that some formatting constraints can already be assumed, for example the input always consists of lowercase letters and does not contain special characters.

A general preprocessing can be executed on the input query. As parsing involves finding a match between the input query and the vocabulary entries, both sides may be processed to add similarity between them via normalization. For example, the car model named "330i", might be translated by the speech recognizer to "3 30 i", "330 i" or "3 30i", depending on the prosody used by the speaker. In order to find a match in the vocabulary, we state that all combinations of several numerals followed by a non-numeral must be separated by a white space character and add a transformation rule to the preprocessor. Alternatively we could have added all these forms to the vocabulary, with the disadvantage that the context depth of the parser (the number of words to be taken as a single token) would have to be enlarged. This would not have helped in this case, because the model "330i" is not in the vocabulary (it is actually not a model but a model variant), but "330" is. Because of the inserted white space, this can now be matched.

After preprocessing, the input gets matched against the vocabulary by the island parser. Firstly, numerical values are detected. The exact algorithm is described in [3]. Numerical values in the context of car search are, for example, price, first registration, mileage or engine power. The words that were interpreted in this module are removed from the search string. Of course we wouldn't want a token like "300" to be interpreted both as a number and as a model name. This leads in German to errors for sentences like "fiat 500 bis 1000 euro" (*fiat 500 up to 1000 euro*), because "500 up to 1000 euro" gets interpreted as a price range, and the model "fiat 500" not recognized, but only the brand "fiat". But this is a natural ambiguity which cannot be avoided.

The remaining string gets matched against the vocabulary entries (entities), the algorithm is again described in [3]. Beneath the set of recognized entities, a rest string containing the words that were not interpreted is delivered, and logged by the system. This can later be used for vocabulary tuning when searching for missing synonyms. For example we saw in the logs that the string "meter" is not interpreted; it certainly is a mis-recognition from the speech recognizer of the word "kilometer", so we added this as a synonym.

The output of the two preceding modules is a set of values and entities like brand, model, or accessories. In this module we do some checks to ensure model-brand consistency, even adding a missing brand for a given model, which is necessary because the search on the AutoScout24 database cannot be executed without stating a brand. Because the AutoScout24 model and brand database is updated frequently when new models appear on the market, the generation of the parser recognition vocabularies from the database is done automatically by a set of extraction rules, which can be edited for maintenance, via a web frontend by AutoScout24 employees.

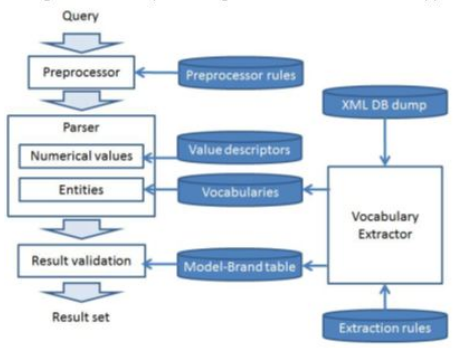

Figure 4: Parsing and maintenance process in the AutoScout24 app

4. Conclusions

We described three mobile apps that deal with voice search. AskWiki can be used to do question answering on Wikipedia content, AutoScout24 app enables the user to search for second hand cars and the TV guide app is used to search the TV program by voice. First promising steps were taken to use linked open data for query expansion, i.e. search the TV program for movies starring a specific actor.

[2]http://en.wikipedia.org/wiki/Wikipedia:Wikidata

5. References

[1] F. Burkhardt and J. Zhou, "Askwiki: Shallow semantic processing to query wikipedia," *Proc. EUSIPCO*, 2012.

[2] F. Burkhardt, J. Zhou, S. Seide, T. Scheerbarth, B. Jäkel, and T. Buchner, "Voice enabling the autoscout24 car search app," *Proc. of the ESSV, Elektronische Sprachsignalverarbeitung, Bielefeld*, 2013.

[3] F. Burkhardt and H. U. Nägeli, "Voice search in mobile applications and the use of linked open data," *Proceedings Interspeech Lyon*, 2013.

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

[5] M. Völkel, M. Krötzsch, D. Vrandečić, H. Haller, and R. Studer, "Semantic wikipedia," in *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23-26, 2006*, MAY 2006. [Online]. Available: http://www.aifb.uni-karlsruhe.de/WBS/hha/papers/SemanticWikipedia.pdf

[6] D. Ahn, V. Jijkoun, G. Mishne, K. Mller, M. de Rijke, and S. Schlobach, "Using wikipedia at the trec qa track," in *Proceedings of TREC 2004*, 2004.

[7] D. W. Embley, D. M. Campbell, and R. D. Smith, "Ontology-based extraction and structuring of information from data-rich unstructured documents," *Proceedings of the seventh international conference on Information and knowledge management*, 1998.