QUARK: ARCHITECTURE FOR A QUESTION ANSWERING MACHINE

Felix Burkhardt

Telekom Innovation Laboratories
Felix.Burkhardt@telekom.de

Abstract: We present QUARK (QUestion Answering Rendering Knowledge), an architecture and a prototypical first partial implementation to answer customer questions in the telecommunication domain from several knowledge sources. The system features a central dialog manager instance that collects answers for a given question from several knowledge sources and selects the most probable based on a combination of confidence value and a weight for each knowledge source "per se".

1 Introduction

We present QUARK (QUestion Answering Rendering Knowledge), an architecture and a prototypical first partial implementation of a system that can answer customer questions in the telecommunication domain from diverse knowledge sources. The system features a central dialog manager that gives answers for a given question by directing it to several knowledge sources. The most probable one, or a ranked order, is selected based on a combination of confidence value and a weight for each knowledge source "per se".
In a first version the knowledge sources are

- A given set of FAQs (Frequently Asked Question) for a customer help web page.
- A database of product information, dummy version
- Domain-specific Forum postings, dummy version

These knowledge sources are very diverse in nature and need different NLP (Natural Language Processing) modules to

- Extract important keywords that can be be used for matching
- Build ontologies from in-domain texts and product vocabularies
- Analyze the questions semantically
- Search for document candidates as answers to a query
- Summarize answers that are too long for speech output
- Render texts from database knowledge

Many of these modules are not fully self-developed but use open source software such as GATE[1], SOLR[2] or JENA[3]. This article describes the architecture, discusses the role of the sub

[1] https://gate.ac.uk/
[2] http://lucene.apache.org/solr/
[3] http://jena.apache.org/

modules and presents the first partial implementation done in Java. Several front-ends, such as web pages or Android applications, were developed. A user evaluation or comparison with other questions answering systems has not been done yet, as the project is in its early stages.
The motivation behind this endeavor is threefold:

- To evaluate NLP modules from companies for our customers.
- To be able to fast build demonstrators in the NLP domain.
- To benchmark the performance of NLP systems, i.e. compare our system, based on open source software, with commercial ones.

2 Related literature

Since the acquisition of Siri by Apple in 2010 and successful market introduction, voice enabled search becomes more and more natural to smart phone users. Google voice search provides a similar service. Both offer, in addition to internet search, services like voice dictation, for example to dictate SMS, command and control ("*Call Peter at home, book a table in the Italian restaurant!*"), and question answering ("*What's the capital of Romania?*").
Beneath these general services, voice search systems for restricted domains have been developed. Song et al [11] describe a system to voice search for media data and incorporate a phonetic similarity model to increase recall on their data. In [8], Voice search is used to generate robust SMS text detection by matching the search query against a database of SMS template snippets. Voice search generally might be a technology to overcome problems like illiteracy and information access in the developing world, as described in [2].
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. [5] use Google search and Word Net as additional information sources. In [13], 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. [1] or [5], 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. [7] describe a system that extracts structured data from car ads.
We implemented two question answering systems in the last years. With Rootvole we queried the Autoscout24 database to voice enable the mobile app [3]. AskWiki was a general domain question answering architecture that allowed based on Wikipedia [4].
Based on this experiences we now present a new system that combines several knowledge sources, or sub-domains, in a centralized question answering interface.

3 QUARK Architecture

In Figure 1 we present the architecture of QUARK. QUARK consists of a number of web services that implement specific functionalities and communicate via HTTP Rest APIs , using JSON and XML as data formats.
Before we describe each sub-module in detail in the following subsection, we give a quick run-through of the architecture.
The input to QUARK is text. It could also be speech, but then a speech-to-text recognizer, also named ASR (automatic speech recognition), would have to be integrated as a pre-step to the

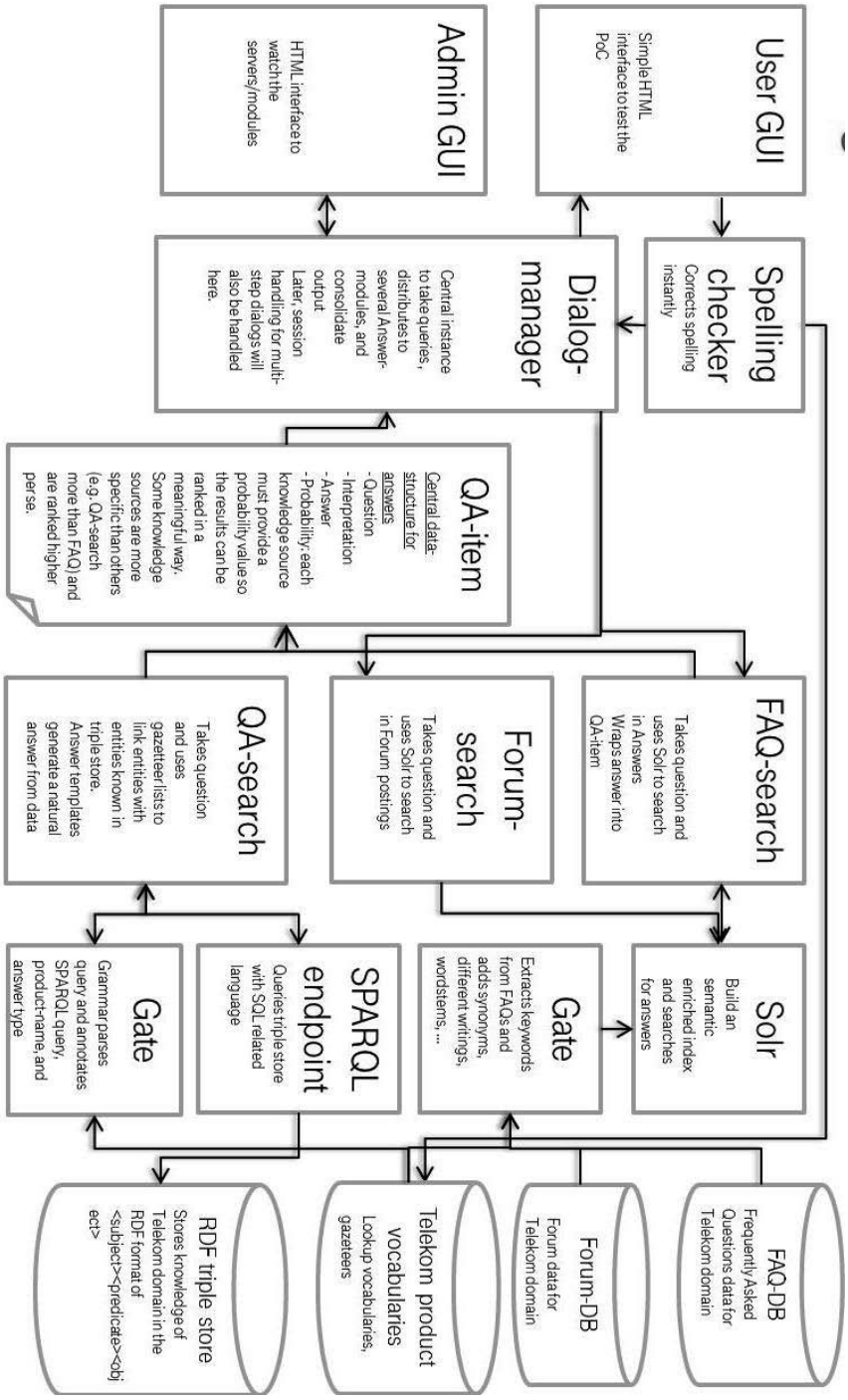

Figure 1 - Overview on the architecture of the question answering system

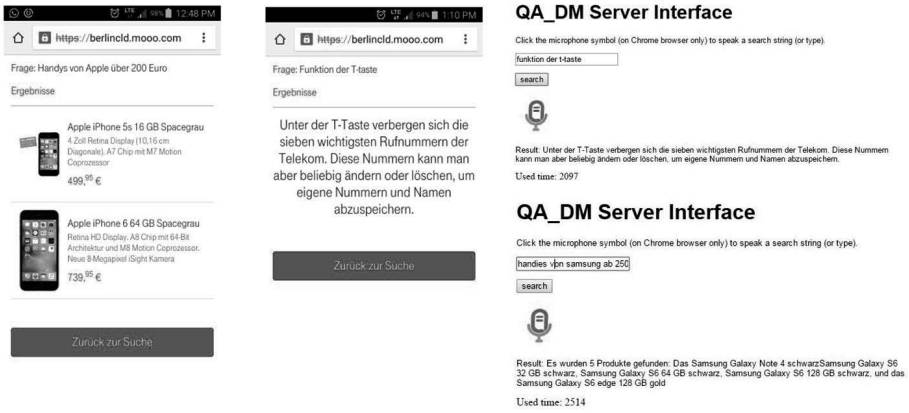

Figure 2 - Mobile app and web interfaces to the system

system. As it's beneficial for the ASR language model to know about the words that are likely to occur, the vocabulary databases of QUARK should then be used to extend the ASR language model.

For first tests of the system we implemented a simple web interface and a mobile Web app (web application implemented in Javascript) to access QUARK. Some screenshots of the current user interfaces for testing and demonstration purposes can be seen in Figure 2. On the left hand side the mobile app is depicted, which also has graphical output. It was developed mainly to demonstrate the system to management. On the right hand side of Figure 2 the web frontend is shown for two input queries.

The web frontend also integrates a maintenance and configuration interface. Beneath the user interface each of the sub modules that run on a web-server has its own administration web interface that is used to view log files, edit configuration files or re-start the module.

The user query gets firstly analyzed by a spell checking module that should protect the user from spelling mistakes. Beneath a German background dictionary domain specific terms like Telekom product names must be added to the vocabulary. The query can be replaced automatically, or the user could be presented with a clickable status "*did you mean ...*".

The dialog manager is the central module to distribute the query to several knowledge sources. In a future implementation dialog functionality would be implemented here. For example with elliptical statements (*"How many i-phones do you sell?" - "And which one is the cheapest?"*, the second one can not be resolved yet.

Crucial for the dialog manager is the confidence value that each potential answer should have. Based on this, a meaningful ranking for a set of answers from different modules can be done, but this means it has to scale equally between modules, i.e. for example 0.8 must be an equally good (or bad) answer across modules. This global quality measure is still an open issue.

The dialog manager gives the query to several potential query answerers that try to match questions from the database, the forum and the FAQs, respectively.

All of them use GATE applications to annotate question and textual content semantically. SOLR is used to do fast indexing and searching. RDF and its query language SPARQL is used to search in data that's distributed over several RDF stores. This is also the way to gap the bridge to linked open data. For example we could use concepts from DBPedia in the annotations. For example, the Wikipedia link to „'mobile phone": https://de.wikipedia.org/wiki/Mobiltelefon, could then be used to identify the concept and retrieve semantic knowledge from it.

4 Modules

This section describes the modules denoted in Figure 1 in more detail.

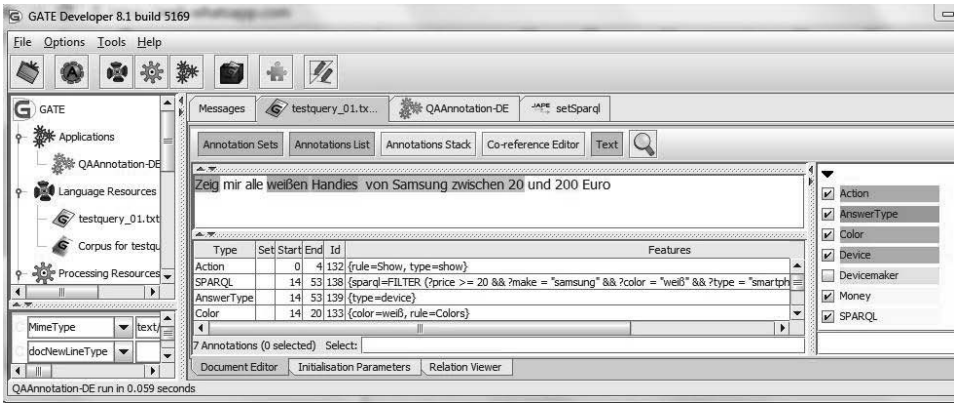

Figure 3 - The Gate user interface annotating a user query

4.1 Spellchecker

The spellchecker is based on the open source spelling checker Hunspell[4]. It provides support for pattern expansion to expand morphological stemming. A context can not be taken into account, this means that the distinction between “That’s reel nice” is wrong but “The reel is broken” can not be made. Several vocabularies can be searched. We use a background general German vocabulary and a specialized, domain-specific vocabulary compiled from Telekom product names. In order to give the specialized vocabulary more weight, both vocabularies get searched and, if they both suggest a correct spelling, the Levenshtain distance of the corrections gets compared. If it’s the same but both suggestions differ, we use the one suggested by the specialized vocabulary.

4.2 Query analysis

This subsection refers to the module “QA-search” in Figure 1.

To analyze the user query with respect to semantic content, we use Gate [6]. Gate (General Architecture for Text Engineering) is a framework to process text linguistically. It is developed and maintained since many years at the University of Sheffield.

Figure 3 shows a screenshot of the Gate Developer GUI, which can be used to develop and test Gate applications before deploying them in a larger context. As can be seen, for the input query “*Zeig mir alle weißen Handies zwischen 20 und 200 Euro von Samsung*”, the semantically important terms are recognized and semantically annotated. This happens by processing the text with a combination of gazetteer lists and Jape grammar files. Jape is Gate’s grammar rule interpreter.

Based on the concepts that appear in the query, a Jape rule assigns a SPARQL statement. This SPARQL search query is then sent to the RDF data store and the result set of smart phones can be retrieved by JENA. JENA is a library, developed originally by Hewlett Packard, to manage RDF data.

[4]https://github.com/dren-dk/HunspellJNA

In the same manner the possible question types are determined by manual JAPE rules. The only thing that can be learned by machine-learning are the terms that denote concepts, e.g. new smart phone names or colors.

4.3 Document semantic annotation

Another possibility to give the user a Telekom-product related answer is to look the question up in the FAQ database. As a first step we imported 2400 FAQ articles from the Telekom help site. These were analyzed with the Gate Termraider application [9]. It uses diverse strategies and measures, like term-frequency/inverse document frequency (TF-IDF), to extract important terms from documents.

As this was originally developed for English, we had to do some adaptation of the processing resources to support German. We used the Stanford Part-of-Speech tagger [12] with a German grammar. As the Stuttgart tree tagger [10], which also does lemmatization, was too slow for our application, we simply added a Gate "Feature Gazetteer" with a German lemma lexicon. Telekom product names were added to the gazetteers.

Figure 4 shows a screenshot of Gate after the Termraider app was run. Important terms from the FAQ documents have been found and can be added as relevant keywords to boost semantic search in the SOLR search engine.

The same process will be done with posts from Telekom Help forum but we haven't come to that yet. As this content is much less controlled than FAQ articles, several more interesting classification problems arise here; for example whether the post is an answer or a question, whether the sentiment is negative, etc.

Figure 4 - The Gate user interface with the Term Raider application

4.4 Document search

The documents that can be presented as answers are then indexed by a SOLR search server. It is possible to add synonym lists to the underlying search engine (Lucene[5]). But in our architecture

[5]https://lucene.apache.org/

it would be difficult to maintain this list. We add synonyms at an earlier stage to the keywords that were extracted from the documents (see section 4.3). The keywords are then used for the search with a higher weight than the full text search.

5 Conclusions and Outlook

We presented a framework that is used to answer textual questions in the Telekom domain. It is based strongly on open-source projects like Gate, JENA, Weka or Lucene.

We use it for research and development as a framework to test third-party software, do benchmarks on semantic technology and build demonstrators and proof-of-concept installations.

The framework is currently in development and will be extended in the future, the aim is a modularized end-to-end system (user to data) that can be used to test and integrate specific components (also from commercial vendors) in the architecture depicted in Figure 1.

Forum search has not been integrated yet. This is an especially interesting topic as the content is much less controlled than information coming from FAQs or database tables. Beneath the problem to identify posts that are semantically related to a given query, it involves possibly type-of-text classification and sentiment analysis.

One of the most pressing problems in the current system is the problem of defining a consistent evaluation measure across the sub modules for the quality of the answers. As the sub modules operate independent from each other it is very difficult to determine which of the given answers from several modules is “better” than another one.

As stated previously, dialog functionality has not be been implemented yet. This means that questions must contain all information to retrieve the correct answer, as additional information slots can not be retrieved by the system or get stored.

References

[1] AHN, D., V. JIJKOUN, G. MISHNE, K. MLLER, M. DE RIJKE und S. SCHLOBACH: *Using Wikipedia at the TREC QA Track*. In: *Proceedings of TREC 2004*, 2004.

[2] BARNARD, E., J. SCHALKWYK, C. VAN HEERDEN und P. MORENO: *Voice search for development*. Proc. Interspeech, 2010.

[3] BURKHARDT, F.: *Voice Search in Mobile Applications with the Rootvole framework*. In: *Proceedings Interspeech, Lyon*, 2013.

[4] BURKHARDT, F. und J. ZHOU: *AskWiki: Shallow Semantic Processing to Query Wikipedia*. Proc. EUSIPCO, 2012.

[5] BUSCALDI, D. und P. ROSSO: *Mining Knowledge from Wikipedia from the question answering task*. Proceedings of the 5th International Conference on Language Resources and Evaluation, 2006.

[6] CUNNINGHAM, H., D. MAYNARD, K. BONTCHEVA und V. TABLAN: *GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications*. In: *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[7] EMBLEY, D. W., D. M. CAMPBELL und 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.

[8] JU, Y.-C. und T. PAEK: *A Voice Search Approach to Replying to SMS Messages in Automobiles*. Proc. Interspeech, 2009.

[9] MAYNARD, D. und W. LI, Y.AND PETERS: *NLP techniques for term extraction and ontology population*. In: BUITELAAR, P. und P. CIMIANO (Hrsg.): *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, S. 171–199. IOS Press, Amsterdam, 2008.

[10] SCHMID., H.: *Improvements in Part-of-Speech Tagging with an Application to German*. Proceedings of the ACL SIGDAT-Workshop, Dublin, Ireland, 1995.

[11] SONG, Y.-I., Y.-Y. WANG, Y.-C. JU, M. SELTZER, I. TASHEV und A. ACERO: *Voice Search of Structured Media Data*. International Conference on Acoustics, Speech and Signal Processing, 2009.

[12] TOUTANOVA, K. und C. D. MANNING: *Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger*. Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), S. 63–70, 2000.

[13] VÖLKEL, M., M. KRÖTZSCH, D. VRANDEČIĆ, H. HALLER und 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.