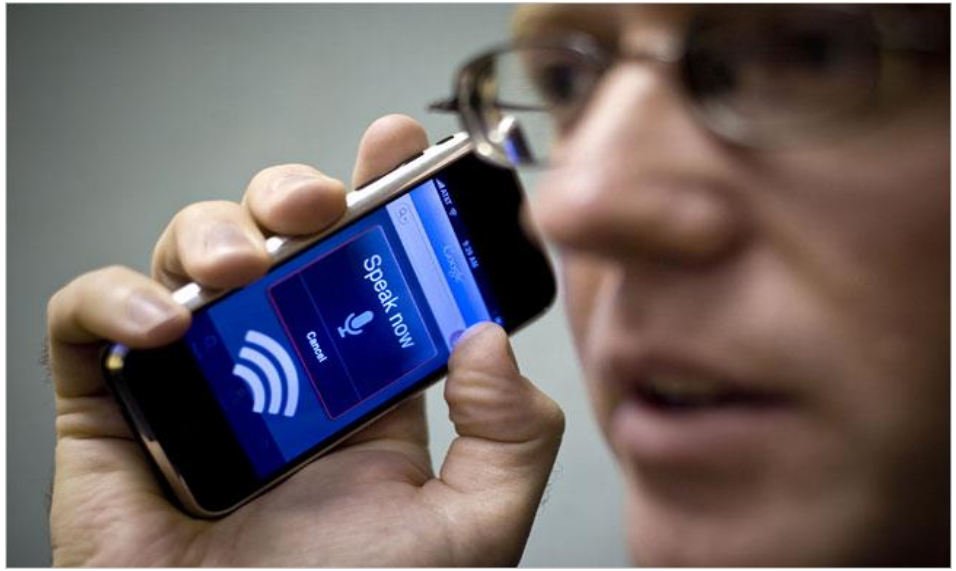
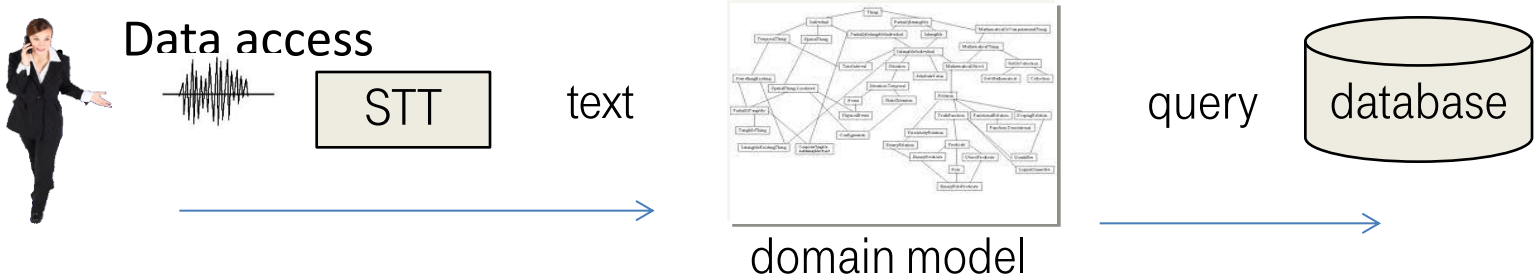
Search by Voice



Overview Search by Voice

- „Search by Voice “: content accessible by Voice
- Implies natural language processing
- Possibly includes acoustic content display (Text to Speech)
- Involves three steps:
  1. Speech to text
  2. Query expansion, semantic interpretation
  3. Data access



Excursion: Natural language parsing

- Old approach
  - Grammar based ASR
  - Recognizes only what is defined in the grammar
  - Fills slots defined in the grammar
- New approach
  - Statistical based ASR
  - Recognizes arbitrary text
  - Needs later text parsing for semantic interpretation

Voice interaction: the next big thing

- 1. Widespread use of mobile devices
  - Smartphones, Glass, Wearables, Implants…
- 2. TV becomes central entertainment center
- means the internet is 24/7 everywhere around us
- Keyboard is in many situations difficult to access,
  - device too small,
  - room too large
  - busy while driving
- Solution is voice control, speech recognition,
- beneath other multimodal inputs like silent voice, face recognition, body movements, biosignals, …

Evaluation measures: recall versus precision

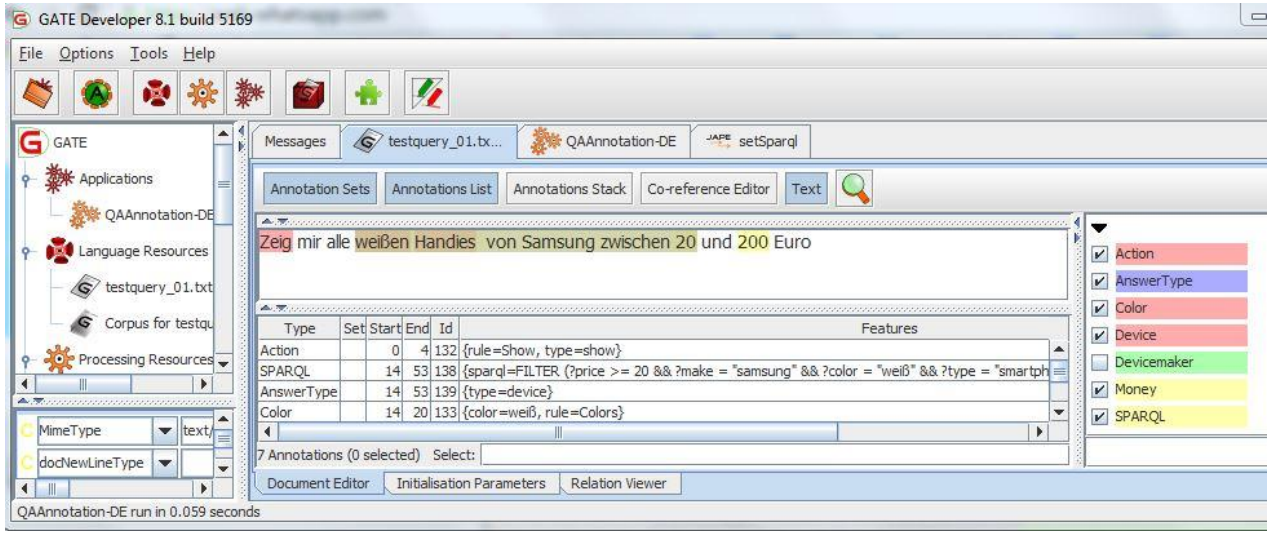
- Recall: how many results do I get?
- Precision: how often did I get the right result?

Example: fishing in a lake for trouts.

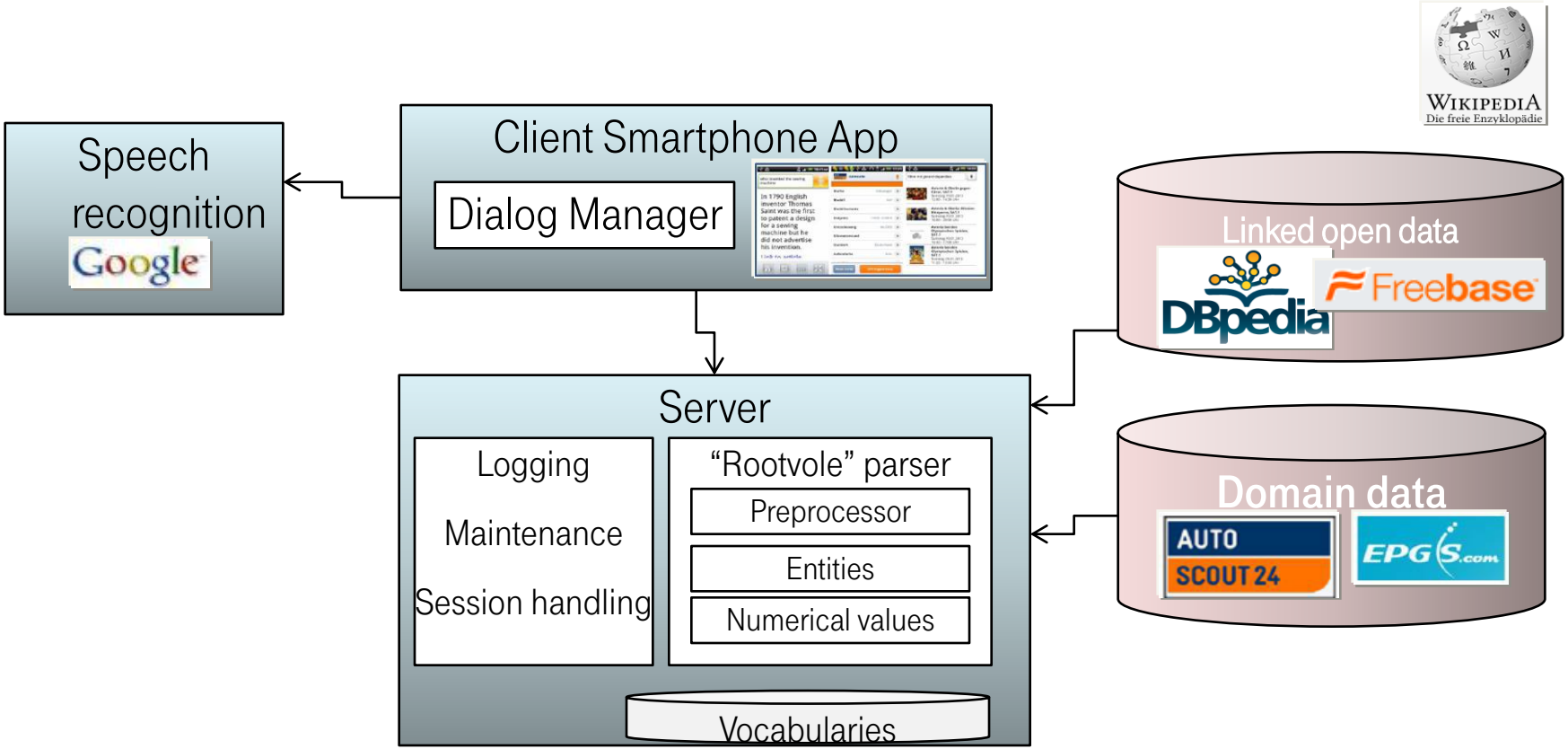
- Recall: relation of caught trouts to all trouts
- Precision: relation of caught trouts to number of catches
- F1 = 2* precision*recall/ (precision+recall)

Tool: Gate

- We use GATE to annotate terms in user queries, based on gazeteers.
- Each string gets annotated with Part-of-Speech, lemma and NER
- Via JAPE grammars, annotations can be expanded
- Machine learning is also possible

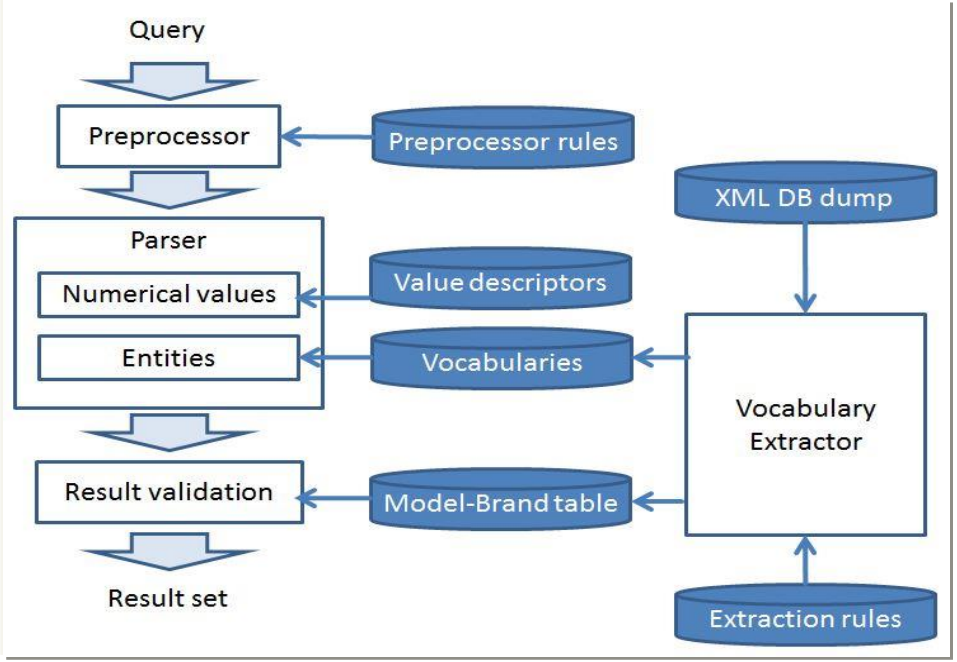


General architecture Mobile Voice Search



Text Parser library: “Rootvole”

- Java library for an island parser
- Parses input for
  - Entities (String)
  - Values (Values)
  - rest string



Rootvole 2

Numerical values are described by

Unit string, e.g. “euro”

Pre- or postfix unit, “Baujahr 2008” vs “300 euro”

Min- max string, e.g. “wenigstens, höchstens”

Can be a region, e.g. “between X and Y”

Rootvole 3

Entities are stored in vocabularies

Format

<id> <synonym_1>...<synonym_N>

Example parsing result:

```
parsing input krimi im zdf (krimi im zdf)
found: got 6 entities and 0 values, (rest: im ) : [genres: krimi (K ⏎
rimi)] [genres: krimi (Krimi-Serie)] [genres: krimi (Krimiserie)] [ ⏎
genres: krimi (Polizeifilm)] [senders: zdf (ZDF)] [actors: zdf (Ein ⏎
e pornographische Beziehung)]
```

RSSReader

General Work mode

- The news reader (RSS-Reader) is an Android Application that reads RSS-Feeds (short news items) from the Internet.
- Features:

– News items can be search for with (spoken) keyword spotting.

– Keyword recognition by Google ASR

– Text to Speech technology (provided by the phone) reads the articles.

– E-mail reading integrated (with K9 interface)

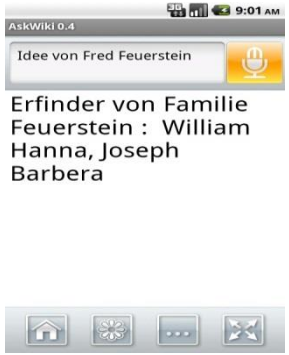
– Three categories: News, Sports, Culture

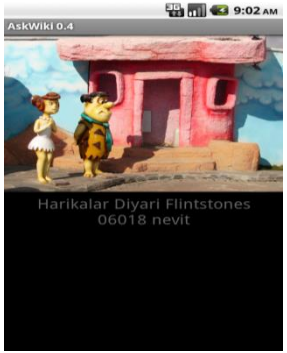
– No server connection needed (new items can be read off line)

AskWiki: overview

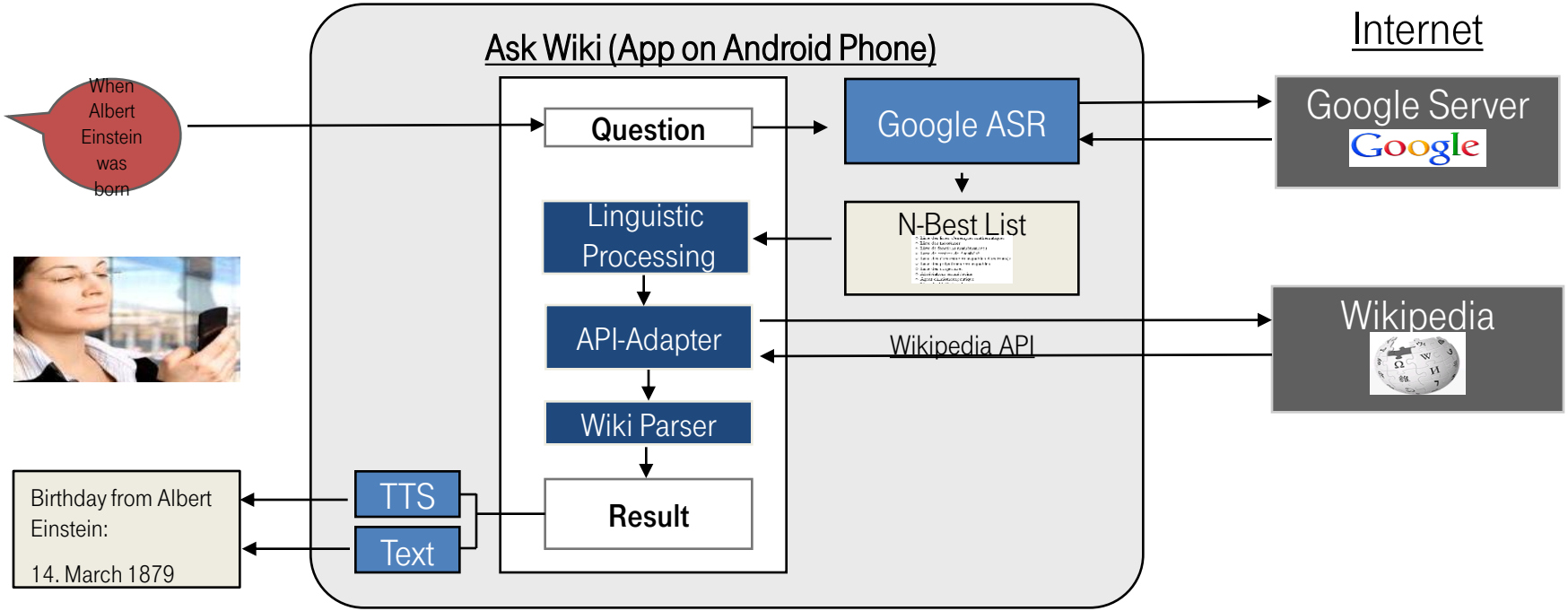
General Work mode

- AskWiki allows users to ask short factual questions.
- The Google Speech Recognizer integrated in the Android operating system allows for speech input.
- Sample queries would comprise
  - "Leiter von Siemens" (*Head of Siemens*)
  - "Vorwahl von Rumänien" (*area code of Romania*)
- It is possible to query for features (e.g. „*Bürgermeister* von Berlin") or sub parts (e.g. „*Nordflügel* Schloss Gottorf")
- Disambiguation can be resolved by naming the category, e.g. „Nashville *Film*"
- The application runs completely on the smartphone without the need to host an additional server





AskWiki: architecture



AskWiki: domain model

Query model

- A query consists of
  - Stop words that may be ignored
  - A feature, e.g. “date of birth”
  - A specifier, e.g. “physicist”
  - The target article’s title
  - Ambiguous words that can be stop words as well as part of the target

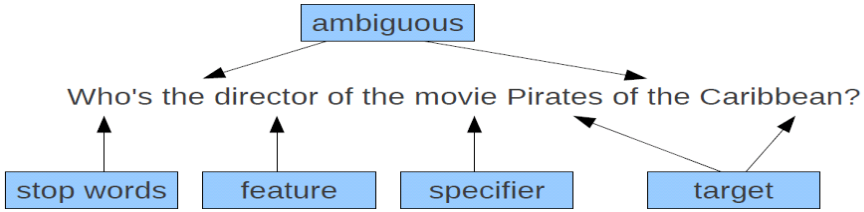


Fig 1: Tokenization of an input phrase

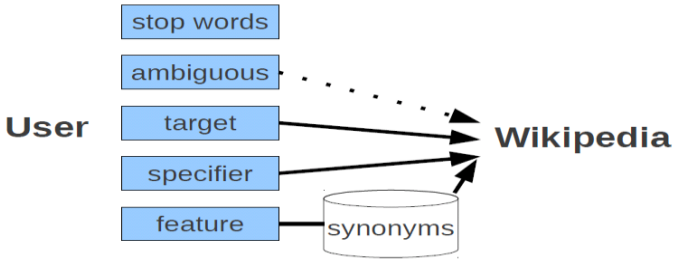
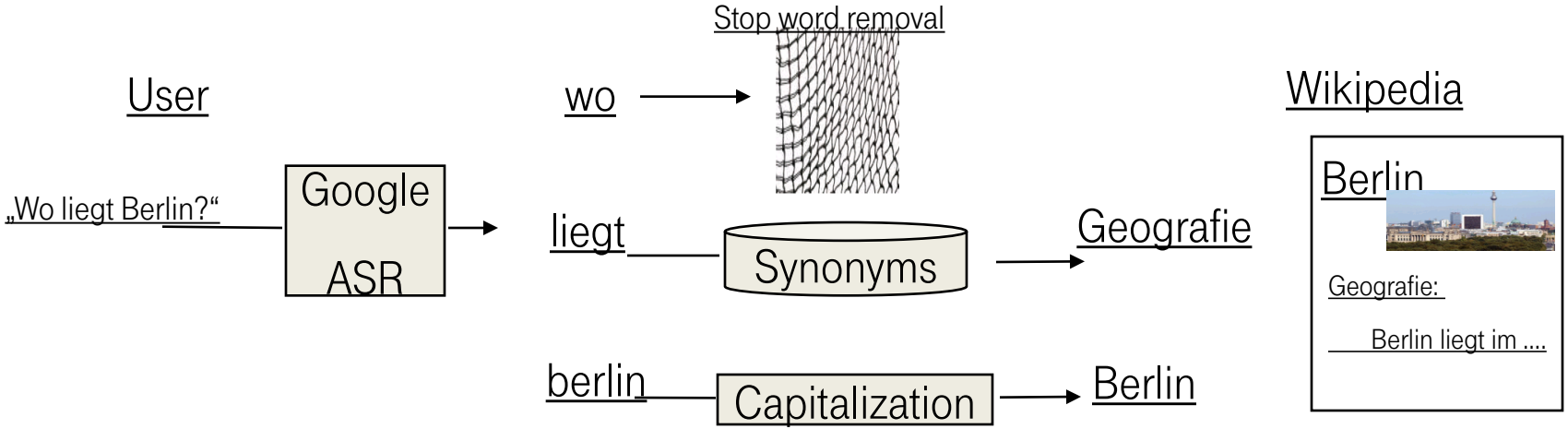


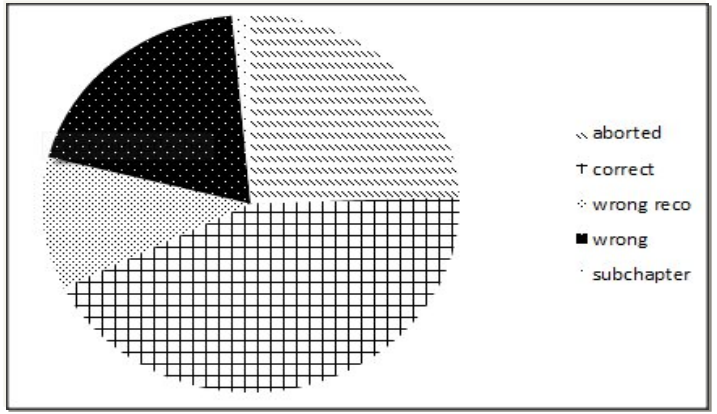
Fig 2: Some words get filtered, others get transformed

AskWiki: example



AskWiki: evaluation

- collected 1052 queries, labeled 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.
- In19.58 %, the answer was wrong or could not be detected.
- Only 1.43 % were queries for sub chapters.



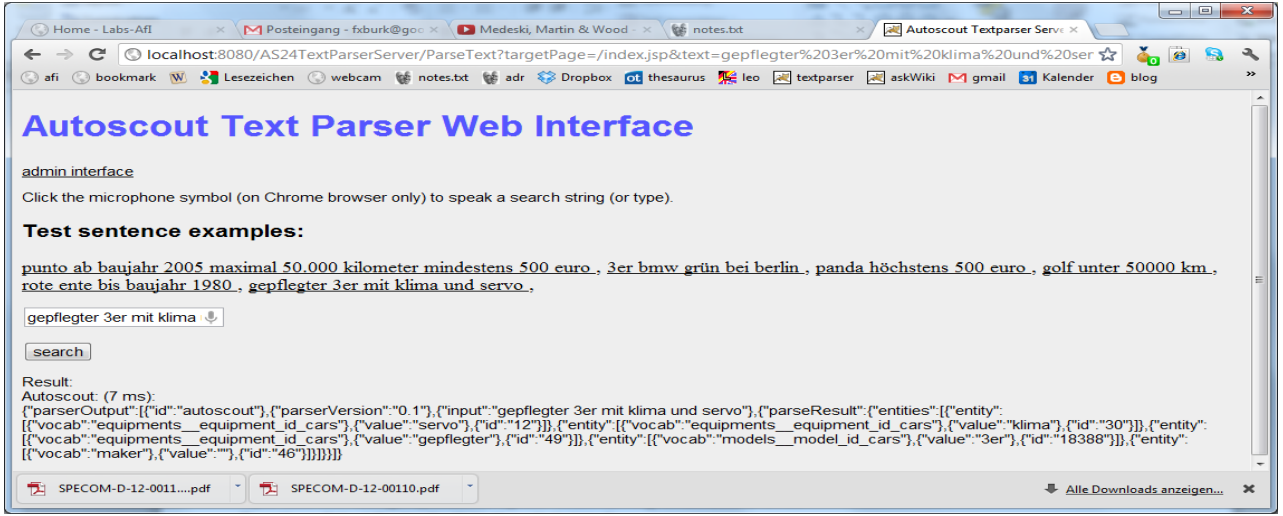
AutoScout24

AutoScout24: overview

- The AS24 Text parser parses natural language queries for car-related words to be used for a voice frontend to the AS24 car search App.
- The admin interface is used primarily to tune the vocabularies of the parser.

The user speech input is recognized by Google ASR (on Android Phone), gets interpreted by the Text Parser Server and the interpretation is then sent to the AS24 database to execute the search.

AutoScout24: overview test interface



- The test interface is used to enter queries and see the JSON response.
- Input can be speech, using the Google speech recognizer, when running in Chrome Browser
- From here, link at the top leads to the admin interface

AutoScout24: overview admin interface

- The administration interface is used to maintain the vocabularies
- There are two parts:
  - **Commands**: execute commands on the server
  - **Files**: edit / view configuration files

AutoScout24: files: AS24 XML file

Commands

Re-Initialize parser

Extract vocabularies from AS24 XML file

Refresh vocabs from disk

Files

- config file
- AS24 XML file
- transformation rules
- transformation vocab
- stopwords

suchparameter.xml write changes to disk


```
<?xml version="1.0" encoding="UTF-8"?>
<response>
        <code>0</code>
        <msg>OK</msg>
        <notice>Notice: You are using a permaToken as access key. This
is valid for development only! Access will be denied with this key
after end of development. If you are not in development please use a
regular access key.</notice>
        <searchparameter>
                <bikes>
                        <accident_free>
```


- AS24 XML file is the basis for the model, maker and other specific vocabularies. They get extracted from this file.

AutoScout24: files: transformation rules and vocab

test interface

Commands

Re-Initialize parser

Extract vocabularies from AS24 XML file

Refresh vocabs from disk

Files

- config file
- AS24 XML file
- transformation rules
- transformation vocab
- stopwords
- min words

vocabLoaderRules.txt write changes to disk

```
# (bla) -> bla
\(.*?\)=$1
# r 11 -> r11
(.+) (\d+)=$1 $2,$1$2
# x-y=x y
(\s)*-(\s)*=$1
# SUV/Geländewage
(\S+)\/(\S+)=$1,$2
# x / y=x y
(\s)*\/(\s)*=$1
^1$=1,1er
^2$=2,2er
```

- Transformation rules and vocab are in Java Pattern Matching syntax and are used to convert AS24 XML labels to vocabulary usable for a speech interface.
- Example
  - Input: SUV/Geländewagen
  - Output: suv,geländewagen

AutoScout24: files: min, max, miles, price

Commands

Re-Initialize parser

Extract vocabularies from AS24 XML file

Refresh vocabs from disk

Files

- config file
- AS24 XML file
- transformation rules
- transformation vocab
- stopwords
- min words
- max words
- miles words
- price words

maxWords_de write changes to disk

```
max maximal bis
```

- These words are used as keywords for boundaries and units.
- Postfix or Prefix notation for units is controlled in the config file:
  - E.g. „miles_postfix=true“ means
    - „3000 km“ gets recognized, but „km 3000“ not

AutoScout24: files: entity vocabularies

test interface

Commands

Re-Initialize parser

Extract vocabularies from AS24 XML file

Refresh vocabs from disk

Files

- config file
- AS24 XML file

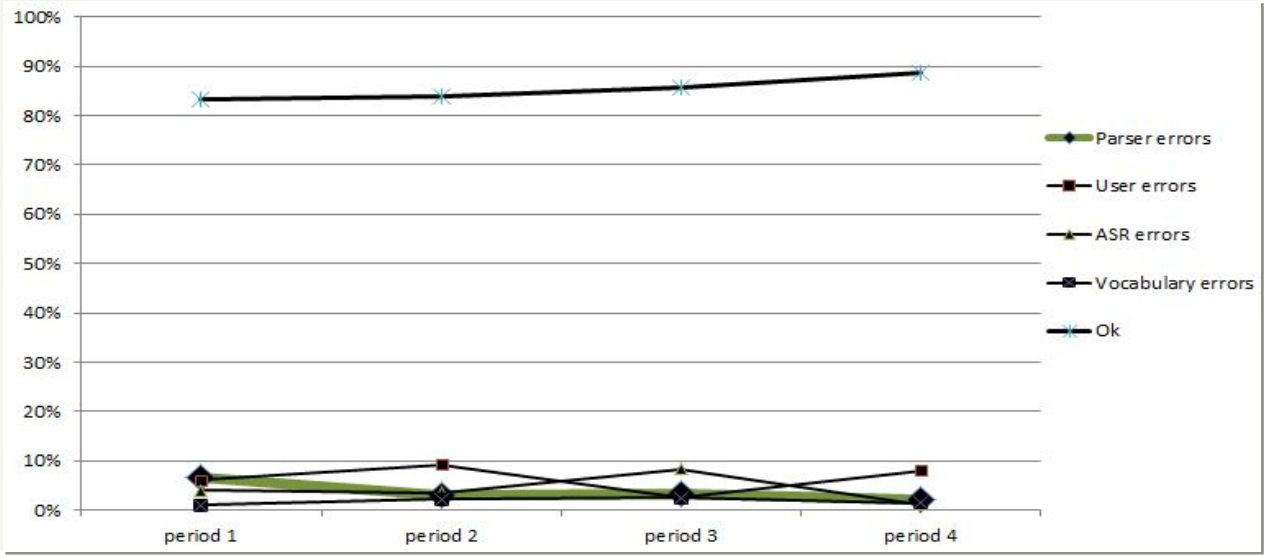
vocab/bodies__body_id_cars write changes to disk

```
1,kleinwagen
2,cabrio
3,coupé
4,suv,geländewagen
5,kombi
6,limousine
7,sonstige
12,van,kleinbus
```

- All other files were extracted from the AS24 XML file and can be viewed here for control.
- The vocabulary syntax is: <id>,<synonym 1>,...,<synonym N>
- If the vocabulary has to be changed, it should NOT be done here but by the pattern matching rules.
- Otherwise synchronization problems occur when the vocabulary (e.g. make or color names and/or IDs) from the AS24 database changes.

AutoScout24: tuning

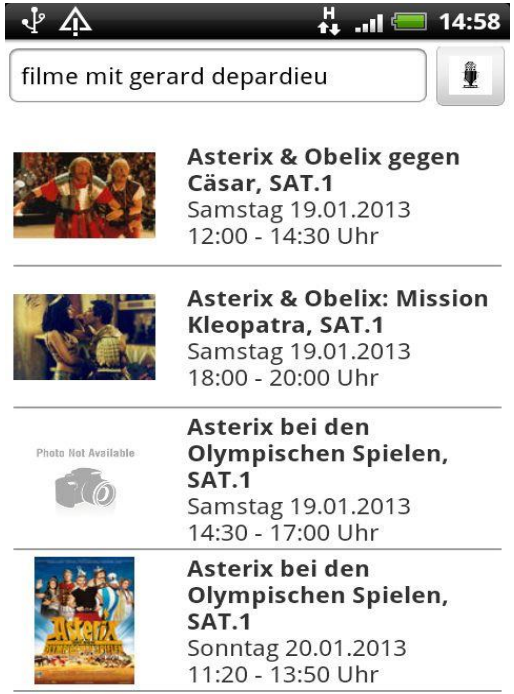
- Measured errors manually in four 1-week periods
- Tuned after each measurement
- Distinguished
  - Parser error
  - ASR error
  - User error
  - Vocabulary error



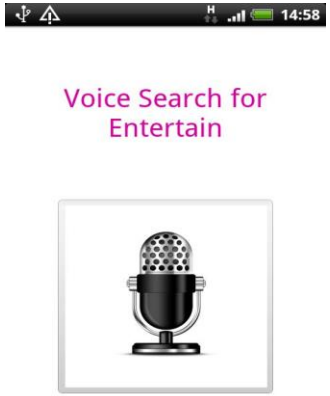
TV program guide: actors

Voice Search for Entertain

+++ Was kommt heute um 20:00 Uhr? +++

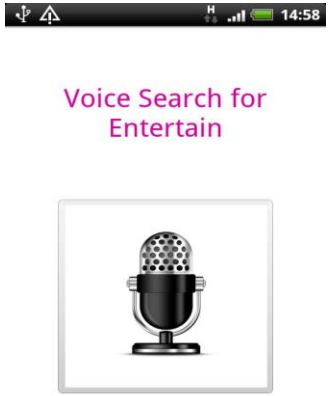


TV program guide: stations



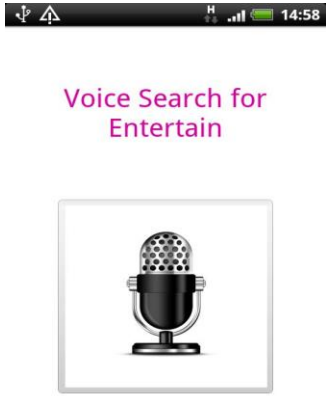
+++ Was kommt heute um 20:00 Uhr? +++

TV program guide: times

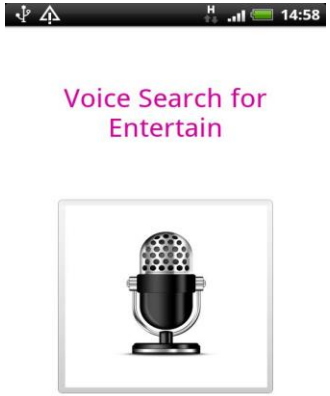


+++ Was kommt heute um 20:00 Uhr? +++

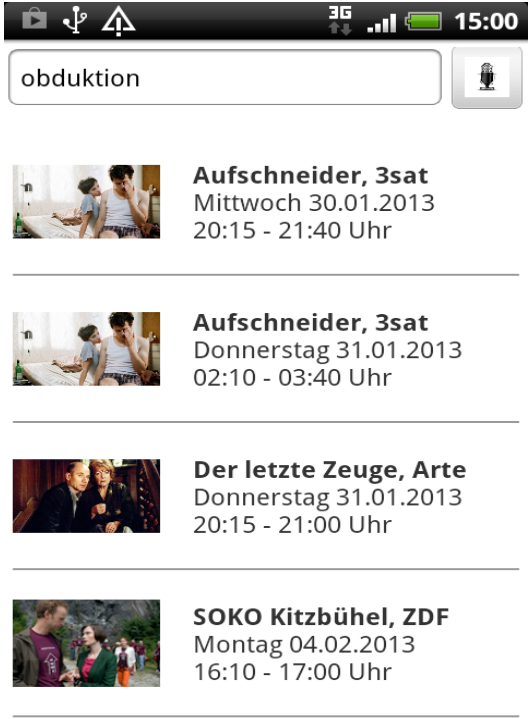
TV program guide: characters



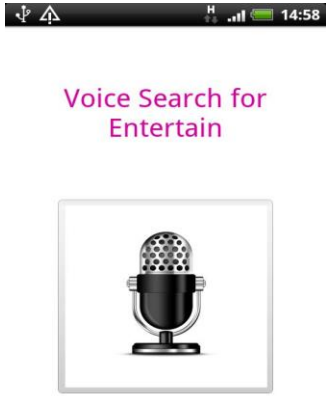
TV program guide: keywords



+++ Was kommt heute um 20:00 Uhr?
+++

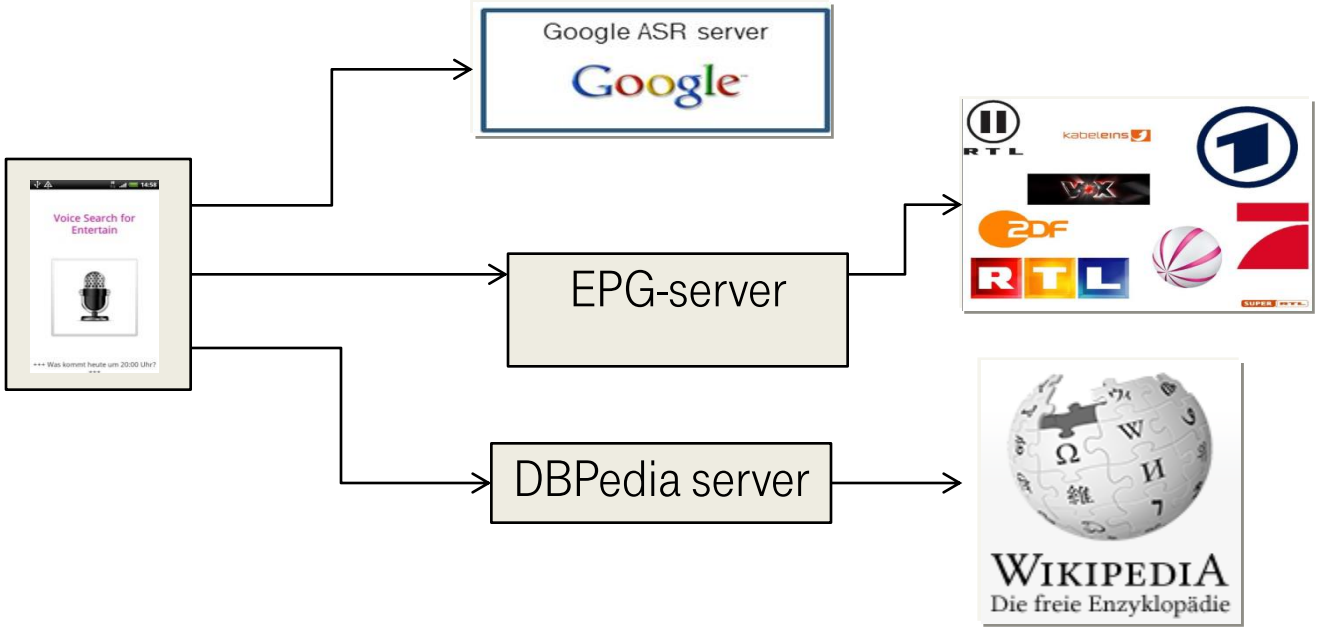


TV program guide: detail view

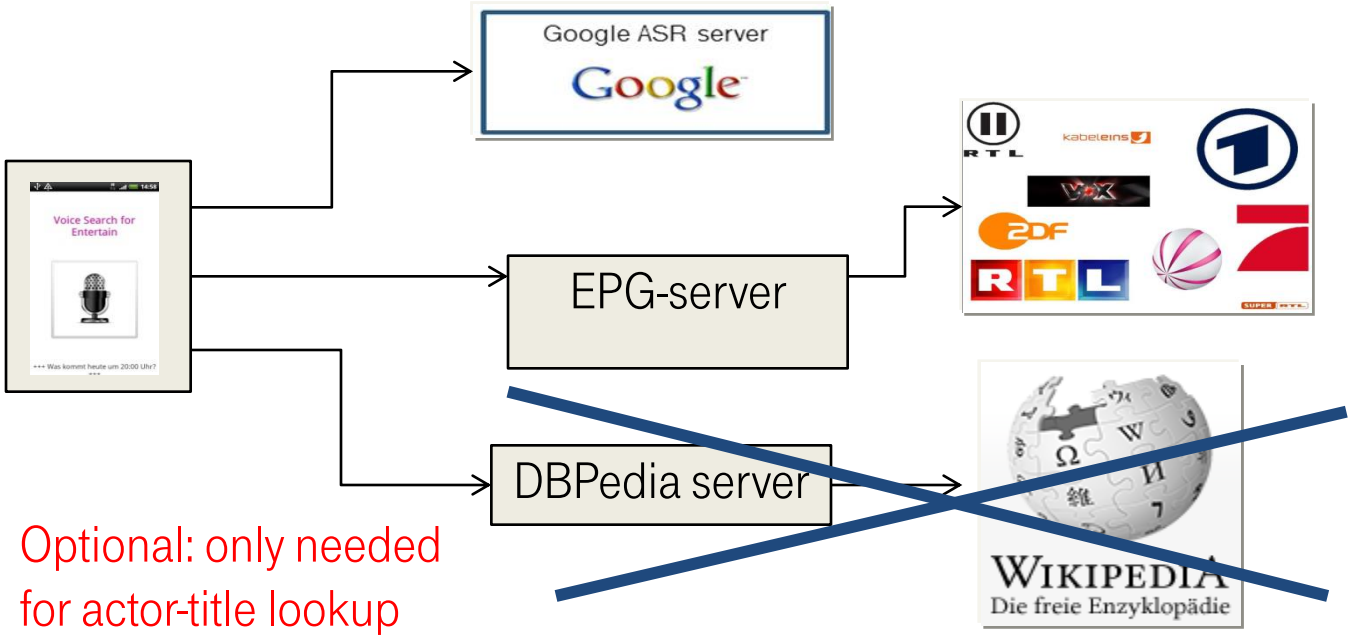


+++ Was kommt heute um 20:00 Uhr? +++

TV program guide: architecture



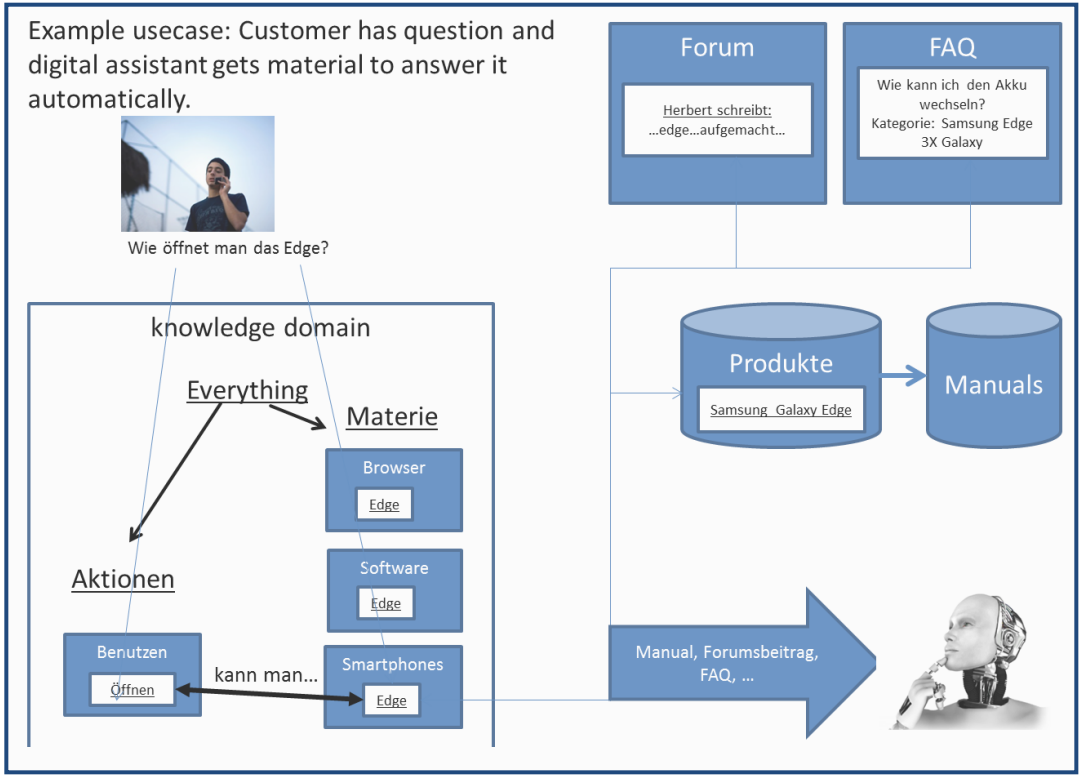
TV program guide: architecture



Quark: Architecture for a QA Machine: MOTIVATION

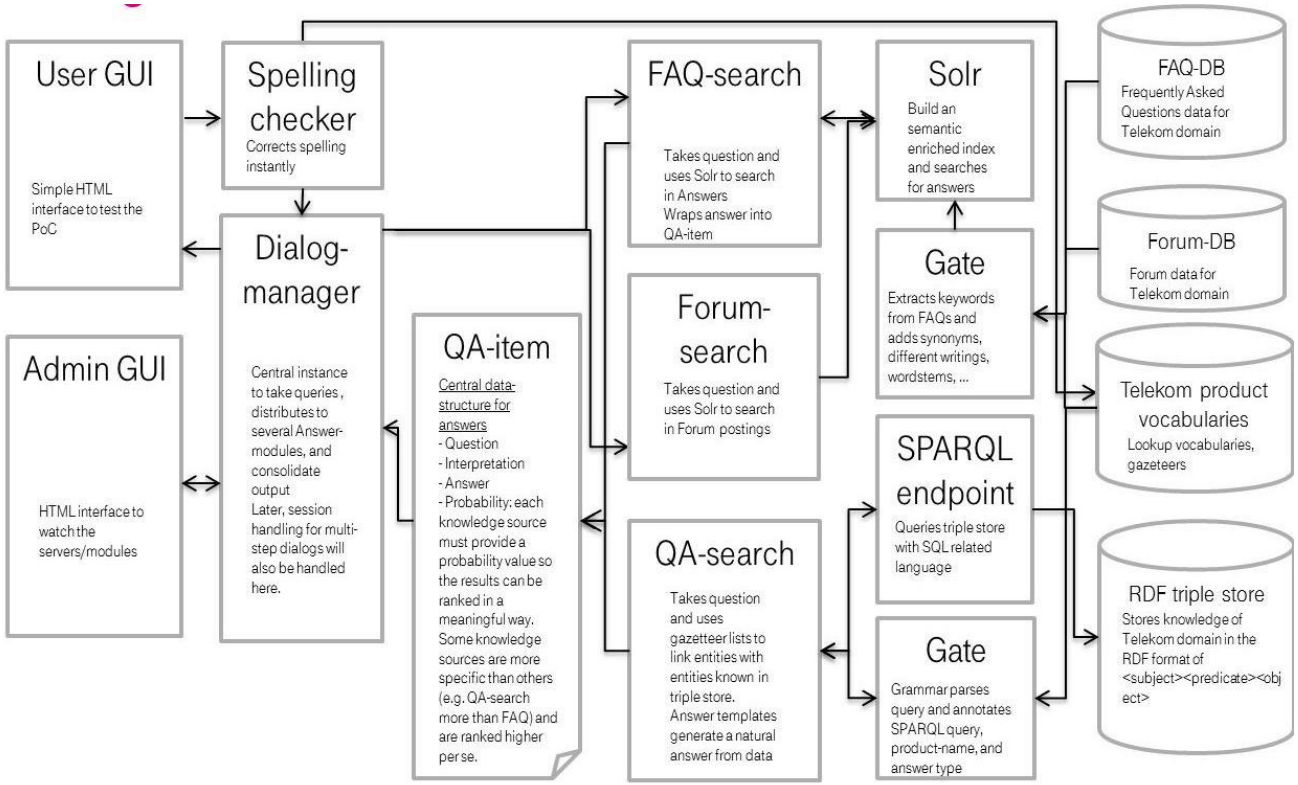
To ease the work of human agents and save costs automatic question answering systems are valuable

One example are so-called „chatbots“, i.e. automatic dialog systems



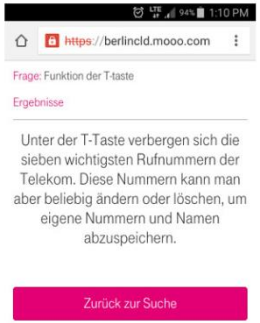
Quark: Architecture for a QA Machine: ARCHITECTURE

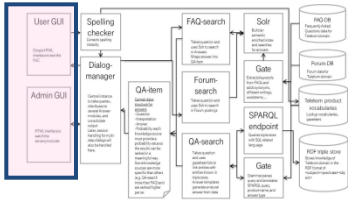
- We developed an architecture to develop, test and compare several components of such a question answering system.
- It is also used to build demonstrators for management

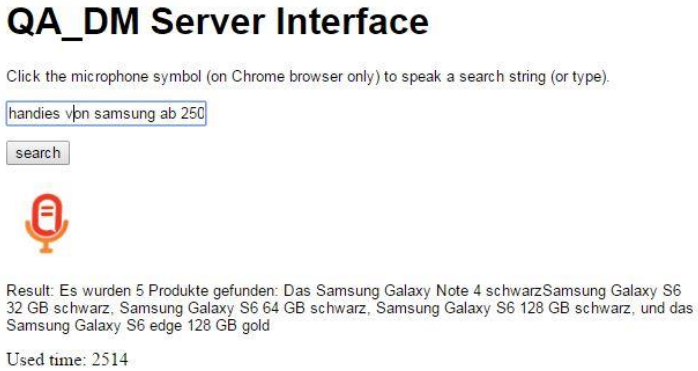


Quark: Architecture for a QA Machine: FRONTENDS

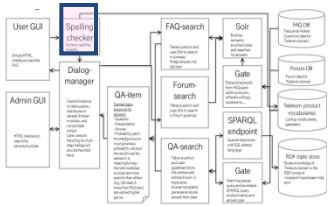
- There are several interfaces, e.g. mobile apps for demonstration
- Web interfaces for testing and maintenance



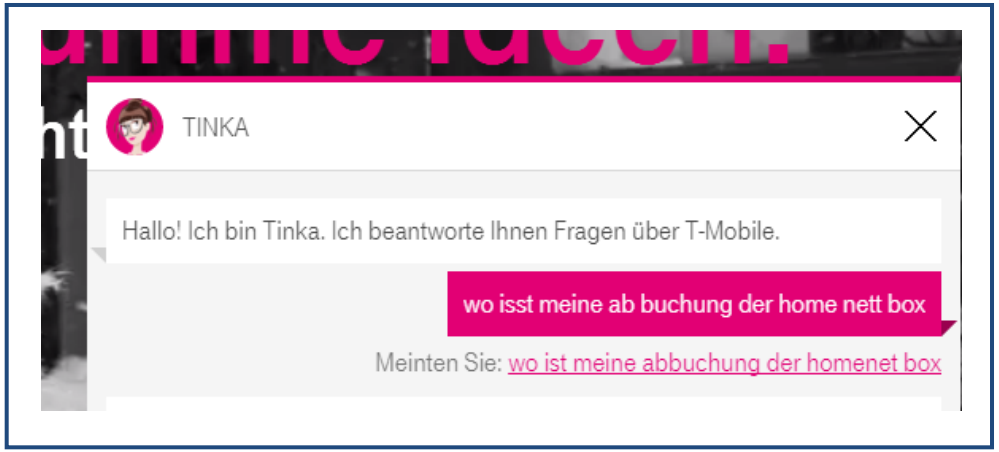




Quark: Architecture for a QA Machine: SPELLCHECKER

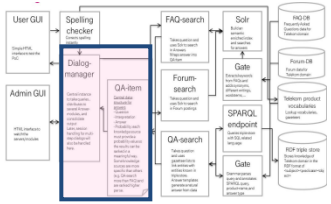


- In the current Tinka implementation we use the open source Hunspell spellchecker
- The API was algorithmically enhanced by supporting
  - Weighted user lexicon
  - Enabling recognition of bi-grams and tri-grams



Quark: Architecture for a QA Machine: DIALOGMANAGER

- Central instance
- Missing dialog model for slot-filling and ellipse handling
- Main challenge is currently the result list order, i.e. a quality measure for the answers from several moduls



Dialog-manager

Central instance to take queries , distributes to several Answer-modules, and consolidate output

Later, session handling for multi-step dialogs will also be handled here.

QA-item

Central data-structure for answers

- Question

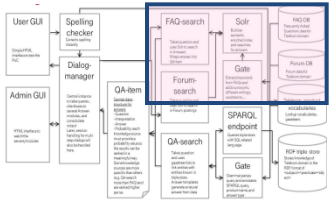
- Interpretation

- Answer

- Probability: each knowledge source must provide a probability value so the results can be ranked in a meaningful way. Some knowledge sources are more specific than others (e.g. QA-search more than FAQ) and are ranked higher per se.

Quark: Architecture for a QA Machine: SEMANTIC SEARCH

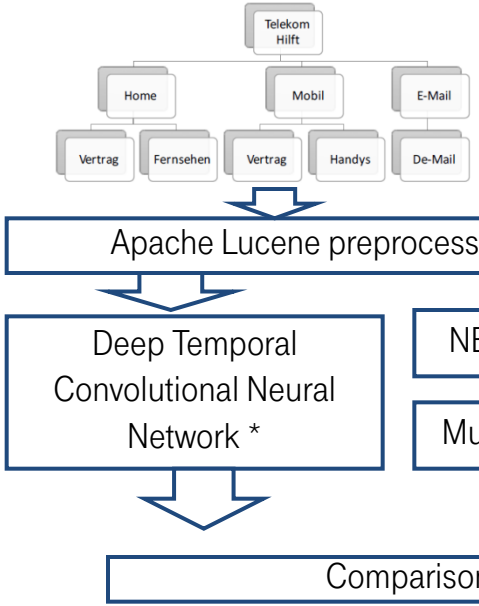
- With content that already contains answers, FAQ and extracted from forum, the user query must only be matched with the question.
- Finding important words and synonyms for „query-expansion“ can be done with GATE‘s term extractor, had to be adapted for German.
- The search index of a SOLR search engine can than be enhanced by these terms.
- The number of matching terms would be part of the quality criterion

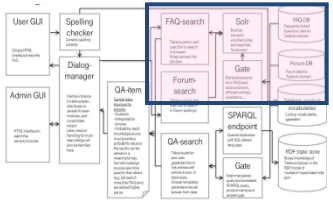


[9] MAYNARD, D. und W. LI, Y. AND PETERS: *NLP techniques for term extraction and ontology population*. In: BUITELAAR, P. und P. CIMIANO (Hrsg.): *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, S. 171–199. IOS Press, Amsterdam, 2008.

Quark: Architecture for a QA Machine:
TOPIC CLASSIFICATION OF FORUM DATA

- Together with the DAI (Distributed Artificial Intelligence) Labor of TU-Berlin we investigated the topic classification of „Telekom Hilft“ user forum
- Compared „classical machine learning“ with Deep neural nets.
- Both resulted in 55% accuracy rsp. 83% „one in three“
- Also investigated subclustering with DNN (4 subclusters per category)

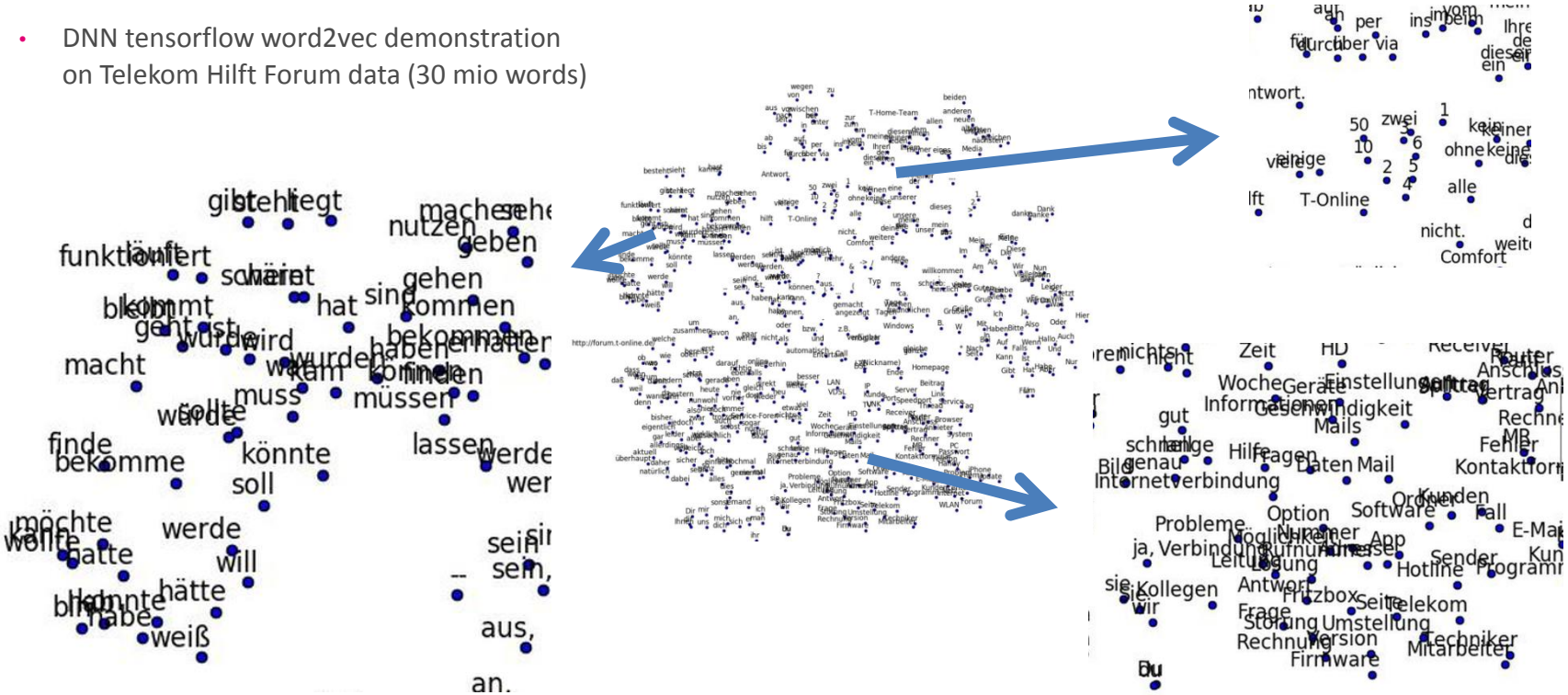




* Zhang, Zhao, LeCun, 2015

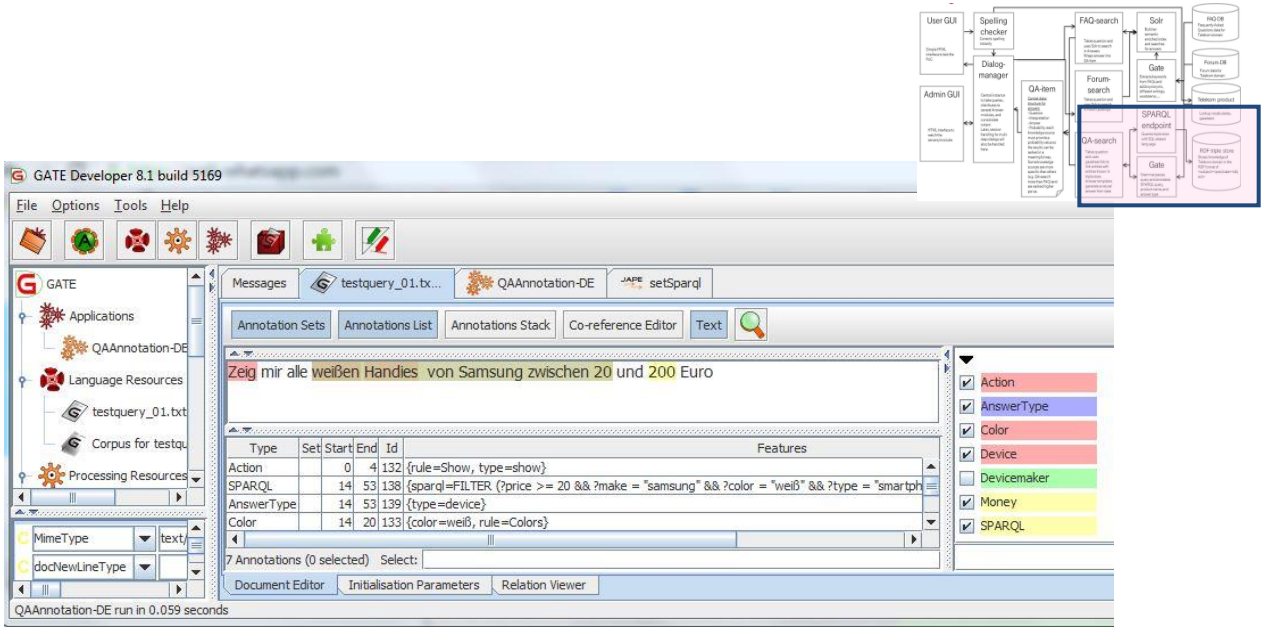
Quark: Architecture for a QA Machine:
TOPIC CLASSIFICATION OF FORUM DATA

- DNN tensorflow word2vec demonstration on Telekom Hilft Forum data (30 mio words)



Quark: Architecture for a QA Machine: ANSWER TEMPLATES

- We also use GATE to annotate terms in user queries, based on gazeteers.
- Each string gets annotated with Part-of-Speech, lemma and NER
- Disambiguation is not done yet
- Via JAPE grammars a pre-defined answer (and querstion) template is determined
- With this template, a SPARQL database is queried and the answer filled.
- SPARQL is the query language for RDF, a W3C suggestion for semantic annotation



[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.

Thanks

fxburk@gmail.com