

# Master Thesis

Porting an English Question-Answering System to German

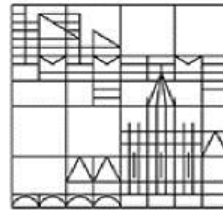
Presented by:

**Kalouli, Aikaterini-Lida**

Student number: 01/815883

at the

Universität  
Konstanz



in cooperation with



For the degree:

Master of Arts

in the field of study ‘Speech and Language Processing’

Evaluated by:

1. Prof. Dr. phil. Miriam Butt, University of Konstanz
2. Prof. Dr. phil. Maribel Romero, University of Konstanz

Konstanz  
14<sup>th</sup> of March 2016

## **Declaration of independent work**

I hereby affirm that I have independently written the attached Master's thesis on the topic "Porting an English Question-Answering System to German" and have not used any other aids or sources other than those I have indicated.

For parts that use the wording or meaning coming from other works (including the Internet and other electronic text and data collections), I have identified them in each case by reference to the source or the secondary literature.

Furthermore, I hereby affirm that the above mentioned work has not been otherwise submitted as a thesis for a Master's examination. I further understand the pending completion of the review process I must keep the materials available that can prove this work was written independently.

After the examination process has been completed, the work will be submitted to the Library of the University of Konstanz and catalogued. It will thus be available to the public through viewing and lending. The described data given, such as author, title, etc. are publicly available and may be used by third parties (for example, search engine providers or database operators).

As author of the respective work, I consent to this procedure.

A current confirmation of my enrolment is attached.

---

(Kalouli, Aikaterini-Lida)

Konstanz, 14<sup>th</sup> of March 2016

## Acknowledgments

At this point, I would like to express my sincere gratitude to Nuance Communications Deutschland GmbH and Nils Lenke for giving me the unique opportunity to conduct my Master Thesis within the company. Especially, I would like to give my warmest thanks to my team, for all the help and support, without which this thesis would not have been completed. I would like to thank Dick Crouch, who with his experience and expertise offered me excellent guidance and valuable help throughout this work. I am thankful to Marisa Ferrara Boston whose deep linguistic understanding, dynamic personality and continuous encouragement and support were a unique driving force for this thesis. I would also like to thank Peter Stublely for providing a priceless source of motivation and for making me part of this great team. I also thank Erdem Ozcan whose technical expertise and support were more than valuable and helpful. I would like to thank each and every one of them for giving me the opportunity to learn so much and to have unforgettable four months. Special thanks are in order for Peter Stublely, Dick Crouch, Charlie Ortiz and Nils Lenke for reviewing this thesis and improving it with their helpful comments.

I would also like to express my deep thanks to my supervisors, Prof. Dr. phil. Miriam Butt and Prof. Dr. phil. Maribel Romero for their supervision and guidance throughout the thesis and for motivating me to choose such an exciting subject from which I had the opportunity to learn a lot and to progress.

Last but not least, I am grateful to my parents for their support and encouragement, my dear brother for all the great advice and help and my beloved Stephan for all his love, patience and understanding.

## Table of Contents

Declaration of independent work.....	i
Acknowledgments.....	ii
Table of Contents.....	iii
1. Introduction.....	1
1.1. Motivation.....	1
1.2. Research Questions and Goals.....	2
1.3. Structure of the Thesis.....	3
2. An introduction to the Question-Answering (QA) systems.....	4
2.1. What is a QA system?.....	4
2.2. Relations to an Information Retrieval System and a Passage Retrieval System.....	4
2.3. The dimensions of the QA task.....	6
2.3.1. Applications.....	6
2.3.2. Users.....	7
2.3.3. Question Types.....	7
2.4. Approaches for the QA systems.....	7
2.5. Components of a QA System.....	10
2.5.1. Question Analysis.....	11
2.5.2. Document collection preprocessing.....	12
2.5.3. Candidate Answer Document Selection.....	12
2.5.4. Candidate Answer Document Analysis.....	13
2.5.5. Answer Extraction.....	13
2.5.6. Response Generation.....	14
2.6. Evaluation of a QA System.....	15
3. Modern QA systems.....	17
3.1. Brief history of the first pioneering QA systems.....	17
3.2. SynTactic Analysis using Reversible Transformations system ( <i>START</i> ).....	19
3.3. Ephyra.....	20
3.4. AskMSR.....	23
3.5. AnswerBus.....	24
3.6. The DeepQA Project.....	25
3.7. Other approaches for the implementation of QA systems.....	27
3.7.1. Ontology-based QA systems.....	27
3.7.2. QA systems using information from previous questions.....	30
4. The Nuance Communications Deutschland GmbH QA system (the ‘thesis version’)....	32
4.1. Introduction to the ‘thesis version’ of the Nuance QA system.....	32
4.2. Components and architecture of the ‘thesis version’ Nuance QA system.....	34
4.2.1. Document preprocessing.....	35
4.2.2. Question analysis.....	37
4.2.3. Passage Search and Retrieval.....	39

4.2.4.	Answer Extraction .....	40
4.2.5.	Answer Ranking/Filtering.....	41
4.2.6.	Answer presentation.....	41
5.	Implementation of the German QA system .....	42
5.1.	Methods and implementation tools .....	42
5.2.	Implementation.....	43
5.2.1.	German document preprocessing.....	43
5.2.2.	German question analysis .....	49
5.2.3.	Other components of the pipeline .....	54
6.	Testing and evaluating the German QA system .....	56
6.1.	Testing the system.....	56
6.1.1.	Testing the functionality of the system.....	56
6.1.2.	Testing the overall performance .....	57
6.2.	Evaluating the system.....	62
6.3.	Discussing the performance of the German system .....	64
7.	Summary and Future work.....	68
	List of Abbreviations .....	69
	List of Figures .....	70
	List of Tables .....	72
	References.....	73
	Appendices.....	76
	Appendix A: English Test set .....	76
	Appendix B: German Test set.....	81

## 1. Introduction

### 1.1. Motivation

Have you ever watched Star Trek and wondered whether all that science fiction technology could ever actually become reality? Today many of those “futuristic” innovations are indeed working applications of our daily life. Representative examples are the portable computers, the tablets, used as Personal Access Data Devices (PADDs) in the series, or the 3D printers reminding of Captain Picard’s tea replicator or even the multi-language translation systems resembling Star Trek’s universal translator. Another such innovation is the “understanding and talking” computer. Although there are already quite a few automatic speech recognition and natural language understanding systems that recognize voice and execute the desired action (e.g. *Siri*, *Microsoft’s Cortana* or *Google Now*), there is still a long way to go until we can interact with the computer as Lt. Commander Data is doing.

The research on and development of Question Answering systems (QA) is a step towards the direction of natural language understanding. Such systems are able to process a question which is stated in natural language, and give back an answer, also in natural language. To the question “*When was J.F. Kennedy killed?*” such a system would return the sentence “*J.F. Kennedy was killed on the 22th of November 1963*” (tested with *Microsoft’s Cortana*). This is already a major difference to the answers that one would get by the modern search engines, such as *Yahoo* or *Google*. These engines would return a number of documents where the answer can be found and we would have to look ourselves for the exact answer within those. Nevertheless, there are simple natural language questions that can be answered directly by such search engines. To the question “*What is the capital of Germany?*” *Google* would indeed return “*Berlin*” as its first result. Since the number of natural language questions that can be directly answered by a search engine is limited to the ones not containing complex natural language features, the modern search engines cannot (yet) be considered QA systems. These are not only capable of delivering one exact answer to a query, but also take context and linguistic features into account in order to deliver the best answer possible.

Escaping science fiction, the development of such systems can contribute to the quality of services in various fields and eventually to the quality of life. A good example is the one of public services or companies. With high performance QA systems, one could imagine of a time in the future, when the employee of a company or of the tax authority or even a doctor in a hospital would just need to type in the desired question and then the system would give

back a reliable, complete and accurate answer based on some predefined documents. This could speed up procedures, save resources and ensure consistency of acting. Such systems could additionally be helpful for people in their daily life, for example when needing a specific information as fast as possible, or for researchers and scientists, who can save time by just getting back one answer instead of having to read large documents of data. On the other hand, such clever systems would be a real assistance for impaired people. Combined with a clever “talking” computer, such systems could be used by people who cannot see or read; the desired query would be formulated orally, the system would execute the search and the answer would be returned orally as well. There are many more applications of a QA system than just those, but these already show the wide range of usage of such systems.

The QA systems originate from the Artificial Intelligence field of the ‘60s. Already in Simmons (1965) there is a survey about fifteen English QA systems that were implemented in the preceding five years. Those first systems, like *LUNAR* (Woods, 1978) or *BASEBALL* (Green, Wolf, Chomsky, & Laughery, 1961) analyzed below, relied on large domain-specific databases, which had to be manually created by experts of the time. Although modern systems contain databases as well, they tend to be less domain-specific, more automated and more flexible as far as the natural language processing is concerned, like the *START* (Katz, 1997) system, also analyzed below. Nevertheless, there are also modern QA systems, which rely on the live replies of other users or experts, such as *Wiki Answers* or *Yahoo! Answers*.

### 1.2. Research Questions and Goals

Apart from presenting the modern QA systems, the main goal of this thesis is to contribute to the QA field by implementing a state-of-the-art German QA system. After an exciting internship in Nuance Communications Deutschland GmbH (Nuance from now on), I was given the opportunity to collaborate with the Nuance QA Team and develop such a system on the basis of one of the versions of the English QA system of Nuance (‘thesis version’ from now on). The questions that need to be addressed within this effort are mainly three. Firstly, it should be shown whether the existing Nuance system, which is based on a partially language independent pipeline, can be ported from English to German by just adjusting the language specific parts of it. If this is so, the goal of the existing QA system not to be too language dependent in its implementation will have been met. In this case, the architecture and the implementation of a QA system, which facilitates its use for multiple languages, will have been proven possible. The next question that needs to be investigated is whether the retrieved

results from the German system are satisfactory enough compared to the English ones. Such a finding would mean that the algorithms of the system are also language-flexible. If these two goals are fulfilled, then the overall implementation of a state-of-the-art German QA system will also be successful.

With the automotive domain flourishing and the desire for “clever” cars getting more and more intense, the QA system to be implemented will be applied on a German user manual for Audi A3 found online. This will then make it comparable with the domain and use case that the English system was applied on. This domain of application has also the advantage of mainly involving factual questions, which are a good start for such a natural language understanding system. The use case is described as following: the driver wants to look up some kind of information or warning message that the car is displaying. In order to do so, he/she would have to pull off to look it up in the manual book. If, however, the user manual is in a digitalized form in the software of the car, the driver can just pose the question (and with the prerequisite that automatic speech recognition is enabled) orally and he/she will get an answer from the system. The current QA system implemented does not incorporate the automatic speech recognition component; the questions are fed into the system in a written form and the answers are also given back in a written form.

### **1.3. Structure of the Thesis**

In section 2 I will give some background information on QA systems by referring to the differences between a QA system and other information retrieval systems, to the dimensions of the QA task and to the different approaches for the QA systems. I will then analyze the components of a QA system. I will also present the measures used to evaluate them. Section 3 will give a brief history of the first pioneering systems and introduce the most popular modern state-of-the-art QA systems. It will also present some other uncommon methods that have been proposed and tested in research QA systems. Section 4 will offer a presentation of the ‘thesis version’ English QA system of Nuance. In the section following I will describe the implementation of the German QA system. I will refer to the methods and tools used for implementing the system and I will then describe its actual implementation. Section 6 will present the performance results of the two systems and this performance will be discussed. The last section will make a summary of my work and proposals for future work.



## **2. An introduction to the Question-Answering (QA) systems**

### **2.1. What is a QA system?**

A question-answering system is a system which receives a natural language question as input and produces a natural language answer as output. The produced answer is ideally concrete and accurate and proof for that is the passage or passages which are also being returned and which confirm why the system gave this answer. In most cases, there are more than one answers returned. The one being considered as the best match is returned first and the others follow in descending order.

### **2.2. Relations to an Information Retrieval System and a Passage Retrieval System**

What is the difference between an information retrieval (IR) system, a passage retrieval system (PR) and a QA system? In the first case the user just gets back documents that seem relevant to the query (Sindhu & Sasikumar, 2014) like the modern search engines work. If those documents are further processed in order to identify the exact passages that contain the answer the user is looking for, then a passage retrieval system is in action (Roberts & Gaizauskas, 2004). Such a system returns the exact passages of the retrieved documents that most probably contain the answer to the query of the user. Systems that go beyond that and return the exact answer to the question are QA systems. QA systems can return either an exact answer as found within a passage – that is the case for the most state-of-the-art QA systems – or create themselves a natural language answer which cannot be found as such within any passage. The latter is the case for QA systems based on databases, where the database entry has to be “translated” to natural language, or for answers that require synthesis and therefore the system has to merge different elements into one grammatical, well-formed natural language answer.

Therefore, one could say that there are three separate systems, but at the same time each of them is a prerequisite for the next one. Sometimes, PR is embedded into IR, if IR is already passage-based. Also, sometimes state-of-the-art QA systems are considered the same as PR systems in the sense that they give back parts of the passages as answers and do not create themselves the answer. (Sindhu & Sasikumar, 2014). Of course, something like that cannot be argued for QA systems coming from databases or requiring synthesis, as explained above. Taking all this into account it is clear that the efficiency of the IR system is of great importance for the quality of the PR and this one is critical for the efficiency of the QA

system. For example, for a simple question such as “*Where is the Taj Mahal?*” a small section of one document would be sufficient. The retrieval of many more documents could actually have the opposite result from the desired one because at the stage of the QA task there would be plenty of chances to get “mishits”. So, in this case, it is crucial that the IR component supplies the PR with limited-length “best” documents and that this only forwards small “best” passages to QA (Roberts & Gaizauskas, 2004).

IR has been researched a lot since the mid-1950s and the designed systems show high performance (Roberts & Gaizauskas, 2004). Representative examples are the search engines, like *Google* or *Yahoo*, which are able to look for information among very large amounts of documents, often using keywords algorithms or similar techniques. The queries that are put into the IR systems need not be grammatically correct or syntactically well-formed and sometimes the search might even return worse results if the query is stated as a grammatical natural language question (Hirschman & Gaizauskas, 2001). The IR community has developed some very good evaluation procedures for these systems, the most famous one being the *Text Retrieval Conference* (TREC), run by the US National Institute of Standards and Technology (Hirschman & Gaizauskas, 2001), where different systems are evaluated based on specific corpora and questions. The techniques used in IR were adapted to retrieve not just relevant documents but also relevant passages, in that way introducing the field of the PR, which is still less known and researched. For this reason, in order to get a small taste of the different approaches that PR is using, the categorization of Roberts and Gaizauskas (2004) might be presented. The approaches differ as to the way they consider a passage and its size and therefore to the way they split a document into passages. There are semantic, discourse and window-based notions of passage. In the semantic ones the boundaries of each passage are marked by the shift of the topic. Whenever a new topic is introduced, a new passage is introduced. For this kind of PR semantic analysis is necessary. The discourse ones are based on the discourse markers of the document itself, meaning its physical boundaries. Such examples are the paragraphs of a document or even its title. The window-based notions of a passage define a standard size of a passage in terms of fixed byte spans; in other words in terms of the number of words contained. The last method is simpler to use since the passages have a standard size while the other models have to get adjusted to the variable size of each passage. The advantage, however, of the first two models is that they return logical and coherent pieces of the document, something very essential if they are used for tasks like QA (Llopis, Vicedo, & Ferrández, 2002). A more original approach of PR is the one proposed by

Llopis et al. (2002). The approach is based on the window-based model with a slight difference. They are not using the passage terms as a measure but the passage sentences instead. This means that the size of every window is defined based on the number of sentences contained in it. This makes their approach different from the classic window-based one in that the window does not always have a standard predefined number of words. Although the number of sentences is standard, the number of words contained in every sentence can be varying. In this way, they are exploiting the advantage of the predefined size passages –in the sense of number of sentences – while at the same time the passages returned are not as incoherent as in the classic window-based model since the sentences are not cut in the middle.

Having briefly presented the approaches and methods of a PR system, there will be no further discussion on those systems, as this would exceed the purpose of this thesis. The focus is to be laid on the state-of-the-art QA systems.

### **2.3.The dimensions of the QA task**

The QA task is a very complex one because it must face multiple problems and combine their solution in a whole. Therefore, it is worth mentioning some of the many dimensions that a QA task should be able to handle (Hirschman & Gaizauskas, 2001). Some of those dimensions are: the kind of applications a QA system involves, the potential users of the system and the questions types being handled by the system. Those characteristics will be analyzed in the subsections following according to Hirschman & Gaizauskas (2001).

#### **2.3.1. Applications**

Firstly, the applications of question answering can be distinguished based on the organization and structure of the source of the answers. The sources of the answers can be completely unstructured like web pages are, semi-structured like the comment fields in databases or completely structured like database entries. The more structured the application, the easier it is to build a QA system upon it but at the same time the less universal the system can be (Hirschman & Gaizauskas, 2001), as it can deal only with this structured data and not with sources of whatever nature. Concerning the range of the search documents a QA system utilizes, they can be anything from fixed collections, as the TREC corpus mentioned before, to widely open collections, like a search over the Web. Finally, there are QA systems that can be applied on only specific domains, in the sense that they can only answer questions coming

from a specific field, e.g. the *BASEBALL* (Green, et al., 1961) system introduced below could only answer questions concerning the US baseball league, and there are also systems without such a restriction which can answer to questions on a wide variety of topics. Depending on whether the systems can be applied to only one specific domain or if they are universal, they can be divided into the domain-specific ones and the domain-independent ones or open-domain, respectively.

### 2.3.2. Users

As argued in Hirschman & Gaizauskas (2001), the users of a QA system can vary from casual users to “professional users”, who might be using the system for their work and thus be very demanding. The type of questions, the type of answers, the value of the correctness of the answers and the overall use of the system are different depending on the type of the user. In the example of the company employee mentioned before, where the person needs the answers to assist his/her customers or solve an issue, the correctness of an answer is definitely more crucial than for someone who just wants to know what the ingredients for the frosting of a cake are.

### 2.3.3. Question Types

Questions can be divided according to the answer type anticipated. There are questions anticipating a factual answer, others asking for an opinion or others expecting a summary or a listing of things. The questions can be further divided according to their type, e.g. *yes/no* questions, *wh*-questions, indirect requests or commands. For example, systems that are based on the use of *wh*-questions will fail to answer to a query which is stated as a command, e.g. “*Who is the president of the US?*” stated as “*Name the President of the US.*” (Hirschman & Gaizauskas, 2001).

## 2.4. Approaches for the QA systems

A QA system will have to address the above mentioned as well as other less significant dimensions of the problem and will have to decide which of those and how it will implement them. In any case, an overall approach concerning the way of processing the question and analyzing the given documents will have to be adopted. There are four approaches to be named according to Dwivedi & Singh (2013) and Sindhu & Sasikumar (2014).

The first is the linguistic one and is making use of Natural Language Processing (NLP) techniques such as tokenization, part-of-speech tagging (POS-tagging) and parsing. The linguistic techniques are applied on the user's question so that the question can be reformulated to a query which is appropriate to extract the relevant answer from a given source, e.g. from a document or a database. In other words, there is some kind of canonicalization taking place, so that the question is turned to a standard query appropriate for the corresponding source. Such a source or "knowledge base" (often organized in logic frames, templates and ontology and semantic networks) contains most of the times information concerning a specific field, which means that such a system would be limited to the information stored in the given document or database. Two of the most famous first QA systems, *BASEBALL* (Green, et al. 1961) and *LUNAR* (Woods, 1978), were using this approach and an example of a modern QA system, relying on the linguistic approach, is *START* (Katz, 1997); all described below. An important benefit of such systems is that their results are good since the natural language is taken into account, however they have long reaction times for large data and they are limited to the specific knowledge bases and domains. Table 1 below gives more details on the aspects of the linguistic approach.

The second approach is the statistical one where the system "learns" in a statistical way through a large amount of data (machine learning) and uses algorithms to compute predictions and similarities. Some of the most popular statistical techniques which are used to make predictions about the expected answer type to a question are the support vector machine classifiers, the Bayesian classifiers and the maximum entropy models. For analyzing the similarity between the question and the document/passage retrieved other statistical techniques are applied, such as the  $n$ -gram mining, the sentence similarity model, the Okapi similarity etc. As supported in Dwivedi & Singh (2013), the statistical approach is beneficial because it can deal with a large amount of data and their heterogenous nature. Such systems are independent of databases, structured query languages or specific documents and the questions in natural language do not need to be canonicalized. Another advantage of this approach is that the method can be adapted for any field since there is no specific knowledge source. However, one of the main disadvantages is that the statistical approaches treat each token of a query independently, without identifying any linguistic features and detecting any linguistic relations between the words of the query. Therefore, the plain similarity-based analysis can sometimes lead to poor natural language results (*cf.* Table 1). One of the pioneer

systems in this field was *IBM's* statistical QA system, whose descendant, *Watson*, will be analyzed below.

Another approach is the pattern-matching one. In this approach text patterns are supposed to replace the sophisticated techniques of the other approaches. Due to the use of patterns, this approach is sometimes called Information Extraction (IE) (Hirschman & Gaizauskas, 2001). To make this pattern-logic clearer, Dwivedi and Singh (2013) refer to the question “*Where was the Cricket World Cup 2012 held?*”, in which there is a pattern “*Where was <Event Name> held?*”. Its answer pattern would be “*<Event Name> was held at <Location>*”. These QA systems just learn such text patterns from passages (machine learning) and avoid the more complex linguistic techniques requiring parsing, ontology classification, named-entity recognition, etc. Most of those systems use the surface text patterns while some of them use templates to generate the answer. The surface pattern based approach relies on a large list of patterns. A candidate answer is analyzed on the basis of similarity between its semantic pattern and the pattern of the question. The patterns are a kind of regular expressions. Designing such patterns is very time consuming and requires a lot of human skill; nevertheless, such systems have a very high precision. The template based approach on the other hand uses predefined templates for the questions. The templates have some entity slots, which are missing information concerning the concept of the question. When those slots are filled, a query template will be generated and this will fetch the corresponding response from the database. The template based QA systems are very similar to the automated Frequently Asked Questions (FAQ) systems, which respond to predefined questions with predefined answers; only that the template based QA systems fill the questions templates dynamically at real time and do not rely on predefined questions. The pattern-matching approach is poor in semantic understanding and reasoning but it is proved useful for handling heterogenous Web data. Different aspects of this approach can be seen in Table 1.

From what has been discussed so far, it is clear that none of the approaches can be considered as the ideal one universally; each approach performs well in its application area. This has led to the development of a fourth approach, which combines all others or some of the others and leads to a hybrid QA system. Various modern QA systems are trying to combine the strengths of the individual approaches and are therefore hybrid in their nature.

Table 1: Comparison between the three main QA approaches (Dwivedi & Singh, 2013:422).

Aspect	Linguistic Approach	Statistical Approach	Pattern Approach
Question type handled	Factual questions	Complex non-factual along with factual questions	Factual questions, definitions, acronyms, birth dates, etc.
Semantic understanding	Deep	Shallow	Less than all other competing approaches
Heterogeneous data handling	Quite difficult as knowledge bases are generally designed only to handle their prestored data type	Statistical similarity measurements are used to integrate data	Easily possible as patterns aid in wrapper generation
Reliability	Most reliable because answers are extracted from self-maintained knowledge bases	Reliable as most of these systems use the supervised approach	Depends on the validity of knowledge resource
Scalability	Quite complex as new rules have to be introduced in the knowledge base for every new concept	Most suitable for handling large data once properly trained	Less because new patterns have to be learned for each new concept
Evaluation Technique/Test Collections	Domain-specific manually developed test collections	TREC, CLEF* test collections	Domain-specific manually developed test collections
Application area	Systems that have long-term information needs for specific domains	Quite suitable in handling large volume data e.g. web	Small and medium size websites

\*CLEF is an EU-project for evaluating and improving IR systems for European languages.

### 2.5.Components of a QA System

Every QA System has its own architecture and pipeline depending on the approach it is using and on the way of dealing with the above mentioned dimensions of the question answering task. However, there is an underlying architecture that can be detected in every QA system and understanding its principles is of importance in order to understand its implementation. Let us see these general components of a QA system, as they are being described in Hirschman & Gaizauskas (2001). Not all QA systems implement all of those components. The components as analyzed in Hirschman & Gaizauskas (2001) are the following: Question



Analysis, Document Collection Preprocessing, Candidate Document Selection, Candidate Document Analysis, Answer Extraction and Response Generation. The general architecture of a QA system is illustrated in Figure 1 (the subsections following correspond to the various boxes of Figure 1).

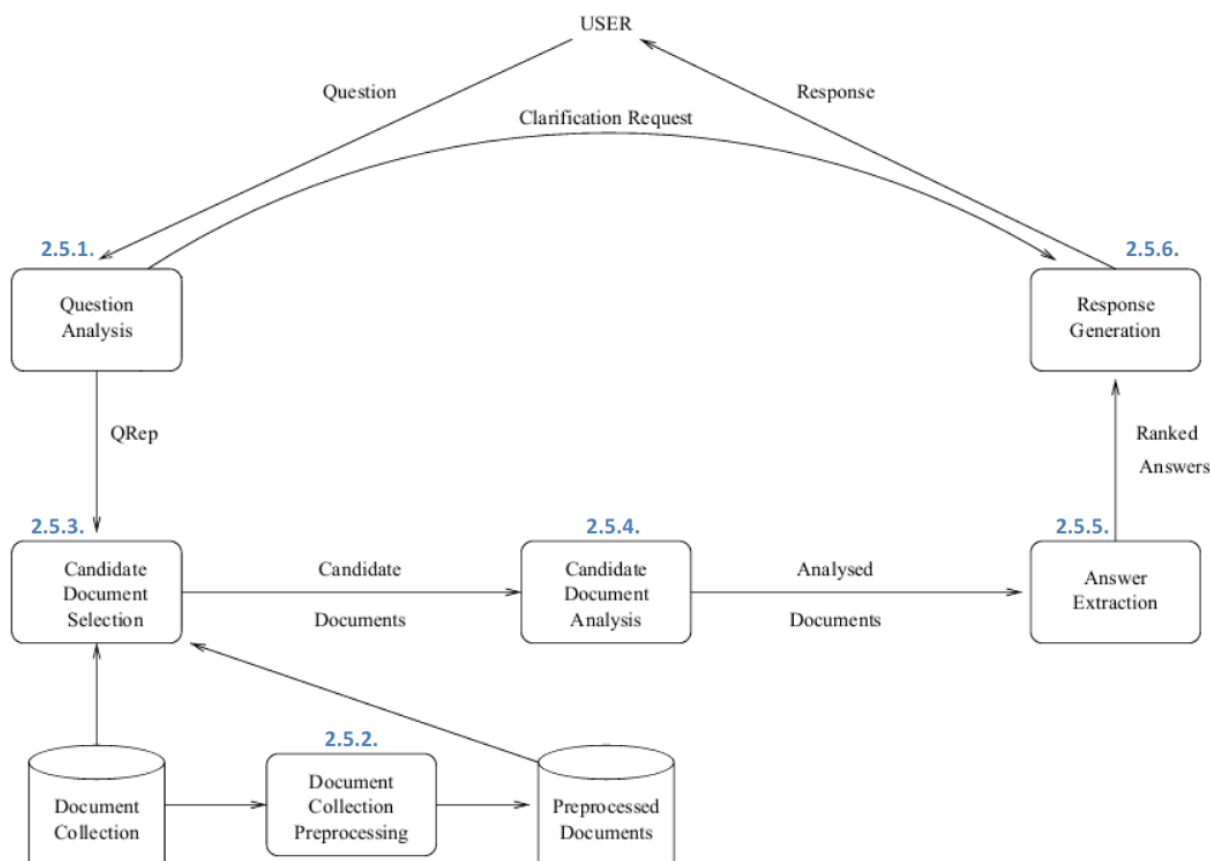


Figure 1: Generic Architecture of a QA system as presented in Hirschman & Gaizauskas (2001) – slightly modified.

### 2.5.1. Question Analysis

During this first stage the natural language question needs to be analyzed and transformed (question reformulation) to whatever form is recognized and needed from the subsequent parts of the system. The question analysis relies first of all on classifying the semantic type of the entity triggered by the question, in other words what is the keyword of the question. For example, *when* seeks a date or a time, *where* seeks a location, etc. This means that the desired answer entity (type) for which it will be searched should match a date and a location, respectively (see *Classification Request* in Figure 1). In questions without *wh*-words finding the central question type is trickier and requires linguistic parsing of the question already at the beginning. Secondly, the question analysis has to define additional constraints on the



answer entity. These could be other keywords of the question which should be matched in the candidate answer-bearing documents. With the power of lexical semantics the keywords can also be expanded with synonyms and other morphological variants in order for the coverage of the question analysis to be as broad as possible. Also, additional constraints on the answer entity could be the syntactic or semantic relations that ought to be true between the answer and the question. These are usually identified with linguistic techniques, such as POS-tagging, parsing and semantic matching. Thus, with the classification of the triggered answer type and the definition of the additional constraints on the question, the question analysis has been completed. The question can now be represented as a query (*QRep* of Figure 1) recognizable by the consequent components of the system and bearing all the extra information it acquired during this stage.

In the described way, systems can process the explicit questions they get as input. For systems, however, allowing on-going dialogue, it might be the case that there is also implicit input in the form of context, which should also be taken into account, e.g. ellipsis or anaphora phenomena. Such systems have to follow similar procedures which will analyze not only the question itself but also its context.

### 2.5.2. Document collection preprocessing

For a QA system to have real time responses against gigabytes of data, the document preprocessing step is more than essential. Such preprocessing can just mean indexing of the given documents or can also include more complex methods. Such methods belong most often to the field of NLP. Common methods used are the semantic analysis of the document, e.g. building the logical representation of the document, as stated in Hirschman & Gaizauskas (2001), or some kind of a shallow linguistic parsing of the document, e.g. POS-tagging, chunk-parsing, etc. Not all modern systems include this stage, which however proves more and more critical for response times. Systems that do implement the document preprocessing do not have to carry out any document analysis at the later stage of the process.

### 2.5.3. Candidate Answer Document Selection

At this stage the candidate documents for containing the answer are being selected. An important decision at this point is whether a system is going to use a boolean or a ranked answer search engine to filter out those candidate documents. Ranked answer engines are the ones that return the documents with the best scores based on machine-learning algorithms or

on statistical algorithms like the  $n$ -gram mining, the sentence similarity model, the Okapi similarity etc. Yet again, it must be decided how far down the ranking will go. Boolean engines are the engines that return results that contain the query as a boolean representation of tokens and boolean operators, e.g. for the question “*where is the Mac Donald’s in Konstanz?*” a boolean engine would return all documents containing *Mac Donald’s AND Konstanz*. According to Hirschman & Gaizauskas (2001), the ranked answer approaches have been returning higher results in the IR evaluation; nevertheless some TREC QA participants have maintained that the use of boolean engines is better for QA systems. As argued by Prager (2001, referred to in Hirschman & Gaizauskas, 2001:289), combining different kinds of search engines can improve performance. It should also be noted that if this stage allows not only for document retrieval but also for passage retrieval, then the PR approach to be used and the parameters (e.g. passage length) should also be defined as discussed in section 2.2.

#### 2.5.4. Candidate Answer Document Analysis

This step is not necessary for systems that have already applied some document preprocessing because it actually resembles the document preprocessing stage in that it applies similar linguistic techniques on the documents selected as candidates. Typically, at this stage there is at least some named entity identification, which classifies strings as names of persons, as dates, as locations etc. Other linguistic processes that could take place at this stage are sentence splitting, tokenization, lemmatization, POS-tagging and chunk parsing. There are also systems that go beyond that and apply a full syntactic analysis on the candidate documents. Then, this syntactic structure is mapped to a logical representation (semantic analysis), which can be further used for the matching to the question’s logical representation.

#### 2.5.5. Answer Extraction

At this stage the representation of the question – acquired in the first stage of the question analysis – and the representation of the candidate answer-bearing documents/passages – acquired either in the stage of the document preprocessing or during the document analysis – are matched to each other and there is a set of candidate answers produced. Each answer of this set carries a score signaling the likelihood of correctness. This matching could have different forms. Most often it means that the candidate answer text should contain a string whose semantic representation matches the one of the expected answer. For covering cases of

lexical relations such as hypernymy or hyponymy, there are lexical resources such as WordNet (WordNet: A lexical Database for English, 2015) used at this stage. Example 1 shall illustrate a simple matching process.

(1) Question: Are whales animals?

First Order Logic Representation:  $\forall x [\text{WHALE}(x) \rightarrow \text{ANIMAL}(x)]$  (simplified version of the corresponding declarative sentence *All whales are animals*: generic, prototypical representatives of WHALE assumed)

Document excerpt: All mammals are animals.

First Order Logic Representation:  $\forall x [\text{MAMMAL}(x) \rightarrow \text{ANIMAL}(x)]$  (simplified version: generic, prototypical representatives of MAMMAL assumed)

WordNet entry: *mammal* is a hypernym of *whale*.

In this example, the semantic representation of the document excerpt will be matched to the one of the question because with the use of WordNet it can be found that *whale* is a *mammal* and since *mammals* are *animals*, it follows that *whales* are *animals* and that therefore the two representations are equal (note that the matching in this example is a very simplified one).

Once the matching has been completed and a candidate answer has been selected, there are additional constraints applied to it corresponding to the constraints that were set during the question analysis stage. These additional constraints might be *absolute* which means that their failure would exclude the candidate answer immediately or they can just be *preferences*, which would give the candidate answer a higher score for the ranking but would also not exclude it if the constraint is not present. According to Hirschman & Gaizauskas (2001) most QA systems do not set these constraints as *absolute* but just as *preferences* because although they can guarantee a correct answer when they match, to set them as strict absolute criteria for the matching would be too demanding. As Hirschman & Gaizauskas (2001:291) claim, “It would sacrifice too much recall for precision”.

#### 2.5.6. Response Generation

After the candidate answers have been extracted, the final response can be generated. According to the TREC QA evaluations, the response that most QA systems generate is a ranked list of the top five candidate answers. Many systems deliver as well the evidence to those answers, meaning the passages from which the answers were extracted.

## 2.6. Evaluation of a QA System

In order to evaluate a QA system, there are two stages to be completed. First of all, we should be able to evaluate the quality of each answer retrieved for a question separately. Some of the criteria to do so can be found in Hirschman & Gaizauskas (2001):

- **Relevance:** the answer should be a relevant response to the question.
- **Correctness:** the answer should be factually correct.
- **Conciseness:** the answer should not contain irrelevant information.
- **Completeness:** the answer should be complete, i.e. a partial answer should not get full credit.
- **Coherence:** an answer should be coherent, so that the questioner can read it easily.
- **Justification:** the answer should be supplied with sufficient context to allow a reader to determine why this was chosen as an answer to the question.

Until now, in TREC an answer is considered correct when it is a relevant response to the question (relevance) and there is enough evidence showing that the document from which the answer originates indeed contains the relevant information (justification) (Roberts & Gaizauskas, 2004). The other measures are not taken into account yet. Unsupported is an answer which is a valid response to the question but is not originating from the source document. All other answers are considered incorrect (Roberts & Gaizauskas, 2004).

Given a test set with various questions and based on the evaluation of each answer of every question of the set, we can proceed to make conclusions about the overall performance of the PR component of the system. For each question of the given test set we can extract the following metrics, which can then be combined to one overall evaluation for the whole PR component:

- a: Number of retrieved relevant hits
- b: Number of retrieved irrelevant hits
- c: Number of relevant hits, which were not retrieved

Two of the core measures that we can calculate based on the metrics  $a$ ,  $b$  and  $c$  are:

- **Precision:** returns the number of retrieved passages/answers that are indeed relevant to the query or with the mathematical notion:  $\text{precision} = \frac{a}{a+b}$  (simplified from Roberts & Gaizauskas, 2004).
- **Recall:** returns the extent to which all relevant passages/answers are being retrieved or, in other words, the fraction of the retrieved relevant passages/answers to all (retrieved and unretrieved) relevant passages/answers or with the mathematical notion:  $\text{recall} = \frac{a}{a+c}$  (simplified from Roberts & Gaizauskas, 2004).

Recall and precision are two measures which are normally considered together, as looking at only one of them is not reliable. In other words, it is easy to achieve high recall (e.g. by retrieving all passages as a response to any query) but this is not a representative measure if the number of irrelevant retrieved passages is unknown. Therefore, in this case the computation of the precision is necessary. A measure that combines those two core values is the f-measure:

- F-measure: is defined as the harmonic mean of precision and recall or with the mathematical notion:  $f\text{-measure} = \frac{2 * recall * precision}{precision + recall}$  (simplified from Wikipedia: Information Retrieval, 2016).

Other modern measures for evaluating the quality of a PR system are:

- Precision at k or P@k: returns the number of relevant passages/answers within the top  $k$  passages/answers retrieved. For example, P@3 returns the number of relevant answers found among the top three answers retrieved (Wikipedia: Information Retrieval, 2016).
- Normalized Discounted Cumulative Gain or nDCG: measures the usefulness or the *gain* of a passage/answer based on its position in the result list. A highly relevant answer which is at the top of the result list gets a higher score while a highly relevant answer that is found further down in the list is penalized. In other words, nDCG takes into account the ideal ranking (if all best results were ranked in descending order), and compares this to the ranking that is actually returned. For more details, see Yining, et al. (2013).

### 3. Modern QA systems

In this section the most popular modern QA systems will be briefly described and discussed. Manual QA systems will not be presented because they do not meet the purpose of this thesis. Manual QA systems are systems in which the answers are given from other people. The other people can either be other users of the platform (e.g. *WikiAnswers*) or experts who are paid to answer to the questions (e.g. *Google Answers* 2002-2006). The performance of such systems depends on the users who reply to the queries.

Before introducing the modern QA systems, it is worth getting to know the origins of those systems.

#### 3.1. Brief history of the first pioneering QA systems

In Simmons (1965) there was already a review of the first experimental QA systems. These were systems of the so-called “first generation” (Simmons, 1970). These first QA systems were not based on sufficient linguistic models and were also written in low level languages such as IPL. With the introduction of (at that time) higher level languages such as SLIP and ALGOL, the picture began to change. Proof for that change is the development of the popular *ELIZA* system by Weizenbaum in 1966. This SLIP program was a first attempt to develop a natural language conversation system between a human and a computer. *ELIZA* simulated a Rogerian psychiatrist who would try to help a patient by asking back questions and thus creating a dialogue with the human. The system was based on the substitution of key words of the input of the user into canned phrases. It was the first system using a kind of the pattern-matching approach presented in 2.4. In order to make *ELIZA*'s behavior clearer, let us see example (2) tested on <http://www.masswerk.at/elizabot/> :

```
(2) ELIZA: How do you do. Please tell me your problem.
YOU: I am feeling very sad and stressed all the time.
ELIZA: Do you think coming here will help you not to be sad?
YOU: Yes, I wanted to get some advice from you.
ELIZA: You seem to be quite positive.
YOU: Why can it be that I am sad all the time?
ELIZA: I'm sure it's not pleasant to be sad.
Etc...
```

Sometimes these substitutions were so successful that the users were misled to think that they have a real doctor in front of them. Even though *ELIZA* was the first natural language dialogue program which gave back answers that seemed to make sense, it used no linguistic methods whatsoever to make the dialogue. The first QA systems that seemed to exploit the

properties of the language appeared later in the 60s. Two of the most popular ones are the *BASEBALL* (Green, et al. 1961) and the *LUNAR* (Woods, 1973) systems.

The *BASEBALL* system was a QA system that could take as input questions related to the results of the US baseball League and give back the relevant answer. As presented in Green, et al. (1961), the questions had to be in ordinary English language and they were read from punched cards. The program was written in IPL-V, a language suitable for information processing, making use of list structures to represent information. Briefly, the main components of the system were the following. The *Question Read-in* component turned the question into a sequential list and fed it in this form to the system. In the next phase of the *Dictionary Look-up* every word was being looked up in a prestored dictionary and was then matched to its meaning. Then the system took care of the syntax by applying POS-tagging. After that followed the *Content analysis*, where the meanings of each word and its syntactical characteristics were considered together in order to form a canonical form for the further processing. The *Processor* would then define which part of the stored data matches this canonical form and come up with a list of results as an answer. This list was then presented to the user. The structure of the *BASEBALL* system shows that it was based on database retrieval, where the results stored in the database were made accessible through natural language queries. This made it very limited in terms of its domain – only baseball – although it made a sophisticated use of syntax and semantics (Hirschman & Gaizauskas, 2001).

Another well-known system of that time was the *LUNAR* system by Woods (1973). The logic of this system was similar to the *BASEBALL* one. This system answered questions regarding “the geological analysis of species of rocks that were flown from the lunar surface by the Apollo expeditions” as described in Lapshin (2012). The answers were based on the information contained in a geology database. As described in Woods (1973), there were three phases that led to the answer. In the first one a syntactic analysis of the query was taking place. This syntactic representation was then mapped to a semantic one in order to produce a formal representation of the meaning of the query. The formal representation was then passed on to the retrieval which matched it to the query language of the database and produced a relevant answer. Thus, the system was again database based and therefore domain limited, just like *BASEBALL*.

Except for *ELIZA*, the two other most popular early QA systems, made a precise use of natural language to achieve their goal. Their architecture has many similarities with the

modern QA systems. However, their main disadvantage was that they were of restricted domains in the sense that they answered only questions belonging to a specific domain because their information retrieval components were based on specific databases only. Therefore, the need to implement some domain independent QA systems, the so-called open-domain QA systems, became even clearer. The modern systems that are going to be presented now are not domain-limited and make use either of large databases or of the World Wide Web.

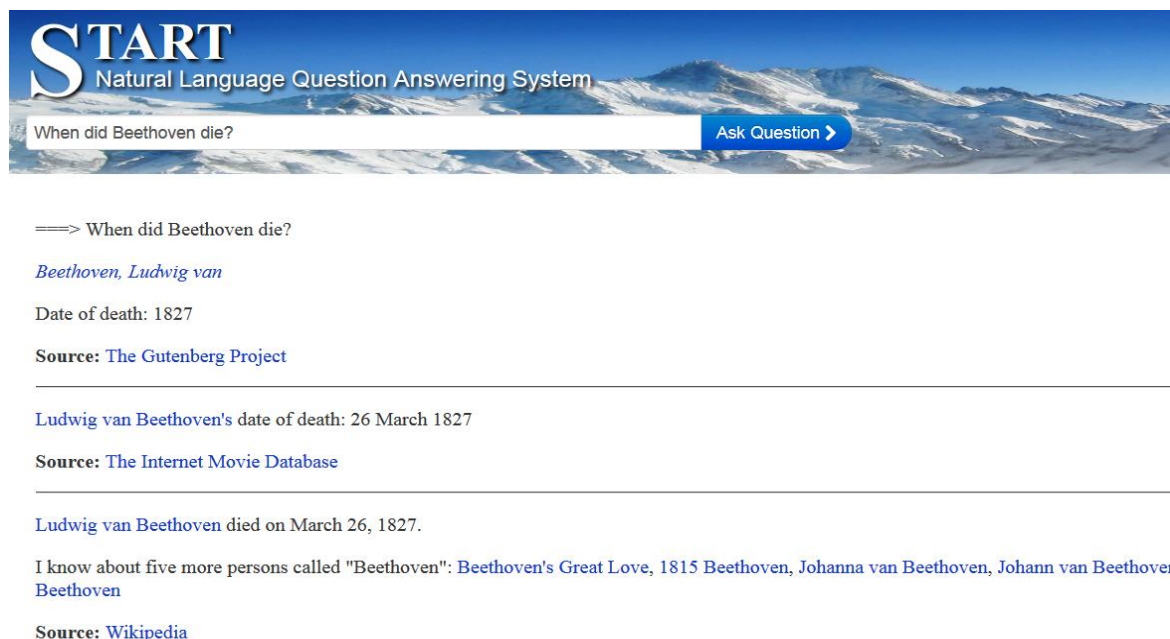
### 3.2. SynTactic Analysis using Reversible Transformations system (*START*)

*START* was developed by Boris Katz at MIT's Artificial Intelligence Laboratory. In 1993 it became available on the World Wide Web and was therefore the first QA system to answer to questions of thousands of users and is still one of the oldest still running QA systems. The system makes use of knowledge bases/databases. Initially it just contained a knowledge base with information about the faculty of the MIT Artificial Intelligence Laboratory (Katz, 1997). Since then the *START* system can answer to a wide variety of questions related to various topics such as geography, weather forecasts, maps, countries etc. Nowadays, it uses other large sources such as the Bosnia Information Server, the Google map collection, the CIA database and many more (Katz, 1997).

The system consists of two modules as described in Katz (1997). The first one is called the “understanding module”. This module integrates the information provided by a given document, database or website into *START*'s knowledge base. In order to be able to integrate so heterogeneous data, it needs a kind of a parser, which does the information extraction, meaning the *web scraping*. The retrieved texts and pictures are then assigned some metadata, which describe further the contents of the retrieved information – the process of the so-called Natural Language Annotation. The “generating module” comes into action when a user asks a question. The question is being recognized and reformulated until there is a positive match with the metadata of *START*'s knowledge base. In this way, *START* is generating an answer from the extracted information annotated with the metadata. The best matches are returned first and other possible answers are listed below.

Let us see how the system responds to the question “*When did Beethoven die?*” (Figure 2), tested on <http://start.csail.mit.edu/index.php>:





START  
Natural Language Question Answering System

When did Beethoven die? [Ask Question >](#)

---

====> When did Beethoven die?

*Beethoven, Ludwig van*

Date of death: 1827

Source: [The Gutenberg Project](#)

---

[Ludwig van Beethoven's](#) date of death: 26 March 1827

Source: [The Internet Movie Database](#)

---

[Ludwig van Beethoven](#) died on March 26, 1827.

I know about five more persons called "Beethoven": [Beethoven's Great Love](#), [1815 Beethoven](#), [Johanna van Beethoven](#), [Johann van Beethoven](#), [Beethoven](#)

Source: [Wikipedia](#)

Figure 2: Example of a *START* answer to the question "When did Beethoven die?".

*START* is performing quite well for queries that correspond to the information that is contained in its knowledge base. Questions that cannot be found in the knowledge base cannot be dealt with because the system does not attempt a search in a search engine.

### 3.3.Ephyra

*Ephyra* is an open-source QA system created by Schlaefter, Gieselmann, Schaaf and Waibel in 2006. The system can be easily integrated in various NLP applications, especially because it can be adapted to other languages than English by adjusting the language-dependent components. The system is based on the training of question-answer pairs and is using the web search engines to fetch the documents suitable for pattern extraction as described in Schlaefter, et al. (2006) (Figure 3). In the first stage of the pipeline the question is normalized, e.g. the punctuation marks are eliminated. Then, a query generator creates one or more queries out of the question. These can be simple queries, a *bag of words* approach, or also more complex representations which rephrase the question and in that way define the form of the answer anticipated. These queries are then passed on to the search module which executes two kinds of search. Knowledge miners use simple informational retrieval techniques to search in unstructured data (e.g. use *Google* to search the web) and the knowledge annotators integrate resources, such as websites, that already provide some semi-structured information (e.g. the *CIA World Factbook*). The search results are then passed on to a set of filters that

check the results for specific characteristics and rank higher only those results that contain these characteristics. After all filters have been applied, the results with the best scores are returned.

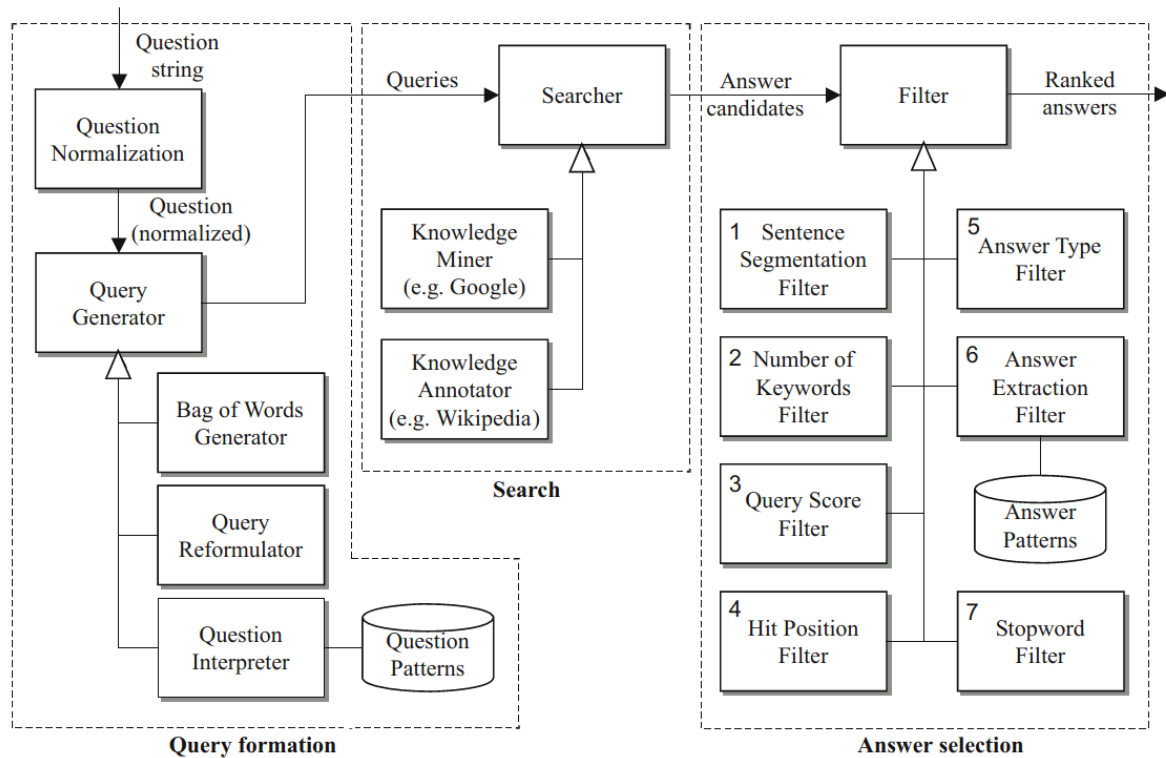


Figure 3: Overview of the *Ephyra* system as presented in Schlaefer, et al. (2006).

The special characteristic of *Ephyra* is that it uses question and answer patterns. Each question pattern can be split in three parts: property, target, and context. This is shown with the example sentence used in Schlaefer, et al. (2006) “*How many calories are there in a Big Mac?*”, where the property is the *number*, the target is the *calories* and the context is the *big mac*. *Ephyra* can distinguish between 70 different properties like *name*, *date*, *location*, etc. For every property there are different frequent question patterns associated with it (in form of regular expressions). A question is interpreted after applying all the question patterns and finding the pattern that best matches the question.

On the other side, the answer patterns extract answer snippets in terms of the recognized search query. Each answer pattern is again split into the same three parts as the question pattern and this splitting relies again on regular expressions. Whenever a snippet matches a pattern, the part of the snippet that corresponds to the *property* part of the pattern is being

extracted. This creates a list of all snippets matching the answer patterns, with every snippet having a different score. The ones with the highest score are the most promising ones.

Although the Ephyra Framework has been giving some promising results, it should be noted that it works only for factual questions concerning locations, numbers, persons etc. Such questions can be easier formulated as patterns than questions asking for reasons or opinions. Generally, such systems using patterns can never capture all possible question and answer patterns since there are endless ways of expressing the same notion in natural language.

Let us see in example (3) how *Ephyra* processes the query “*What is the capital of Germany?*” tested within *OpenEphyra* downloaded from <https://sourceforge.net/projects/openephyra/>:

(3)

Question: What is the capital of Germany?

+++++ Analyzing question (2016-01-26 13:29:56) +++++

Normalization: what be the capital of germany

Answer types:

NElocation->NEcity->NEcapital

Interpretations:

Property: CAPITAL

Target: Germany

Predicates:

-

+++++ Generating queries (2016-01-26 13:30:01) +++++

Query strings:

capital Germany

"the capital" (Germany OR "Federal Republic of Germany" OR Deutschland OR FRG)

"Germany" capital Germany

"is the capital of Germany"

"the capital of Germany is"

+++++ Searching (2016-01-26 13:30:01) +++++

+++++ Selecting Answers (2016-01-26 13:30:01) +++++

Filter “AnswerTypeFilter” started, 0 results 2016-01-26 13:30:01

...

...

Answer: The capital of Germany is Berlin.

### 3.4. AskMSR

The *AskMSR* system is a web based QA system developed by Brill, Dumais and Banko (2002). It is distinguished for its simplicity and efficiency in using the web as a large data resource and avoiding sophisticated linguistic procedures. The system is built upon four stages as explained in Brill, et al. (2002) (*cf.* Figure 4 for the overall architecture). During the first stage, the query is reformulated and each rewrite string gets a different weight. The rewrite strings are actually substrings of the declarative answers to the question. The rewrite strings are of various complexities; from complex ones reflecting the initial concept of the question to the simplest one containing the conjunction of all non-stop words of the question. Since not all rewrite strings will match any documents, the presence of also less precise reformulations of the query gives a greater chance of finding matches. The second stage is the *n*-gram mining. In this stage all rewrite strings generated are being formulated as search engine queries and the search engines return page summaries as results to the search. From these summaries *n*-grams are collected as possible answers to the question. The *n*-grams extracted are then scored according to the weight of the query rewrite that retrieved them. The *n*-grams are then filtered and reweighted according to how well each candidate answer matches the expected answer type. The answer type is predicted after the query has been analyzed and assigned to one of seven question types such as *who*-question, *what*-question, *how-many*-question, etc. The final stage of *n*-gram tiling merges similar answers and creates longer answers out of smaller overlapping answers.

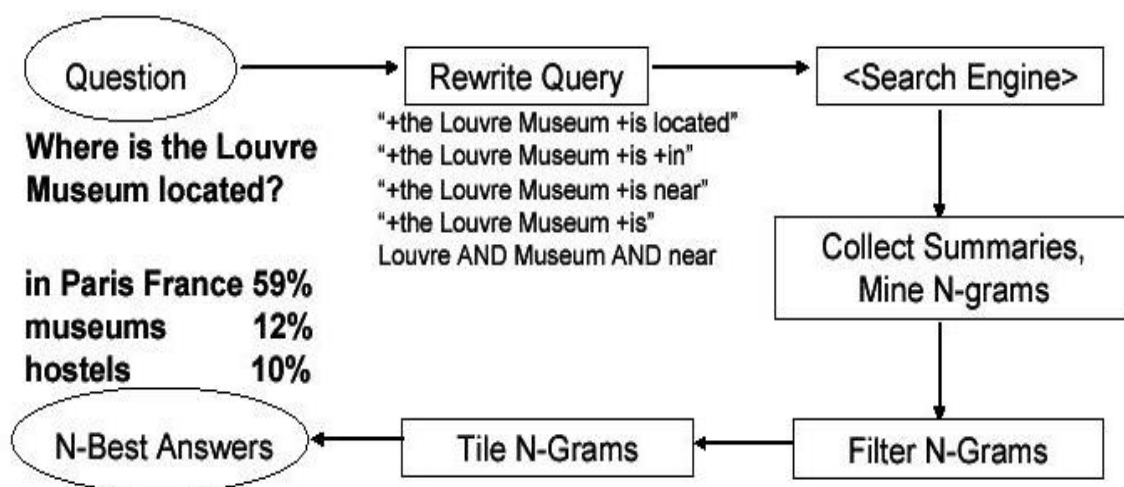


Figure 4: Overview of the *AskMSR* system as presented in Brill, et al. (2002).

As was the case for *Ephyra*, *AskMSR* is able to answer factual questions but proves less efficient when it is confronted with reasoning or more complex questions. One reason for that is that natural language as such is not being taken into account and relations between the words of a question are not being investigated. On the other hand, the amount of rewrite queries has to be finite while the ways of answering a question in a declarative sentence can potentially be infinite, which means that the rewrite queries can never capture all the different possibilities.

### 3.5. AnswerBus

*AnswerBus* is an open-domain QA system implemented by Zheng (2002). It is based on sentence level web information retrieval. Its unique feature is that it can accept natural language questions in English, German, French, Spanish and Italian and return their answers in English. Its special characteristic is that its reaction is a very fast one; this was also one of the priorities when building the system, meaning that it provides not the perfect answer to a query but a good enough answer as fast as possible. Five web engines are used to retrieve the relevant answers from: *Google*, *Yahoo*, *WiseNut*, *AltaVista* and *Yahoo news*. The questions that are not in English are sent to the *BabelFish* translating tool and their English translation is fed back into the system.

According to Zheng (2002) the functionality of *AnswerBus* is based on four consecutive stages (Figure 5). In the first one, two or three search engines among the five are selected in order for the information retrieval to be applied on them. The choice of the search engine is based on the performance of each of them concerning different kinds of questions. For example, *Yahoo News* is probably the most suitable for news queries. The performance was tested beforehand with test queries which were sent to the engines and received a different number of hits in each engine. So, the three search engines get a query as input: the query is the original question reformulated. The reformulation is based on the deletion of functional words, on the frequency of each word and on the modification of the word form. Then, the documents referred to at the top of the hit lists of the search engines are being retrieved. The documents are parsed into sentences and it is determined which sentences could candidate as answers. In order to do so, each word of a sentence is classified as matching or non-matching word. The sentences that could potentially suit as answers are being extracted. These answers are then ranked and the top choices are being returned to the user along with the contextual URL link.

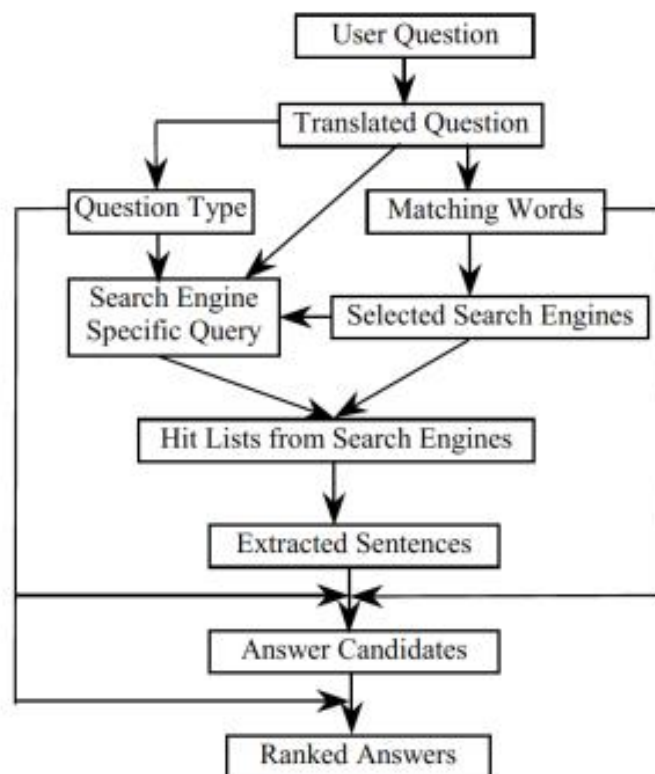


Figure 5: Overview of the *AnswerBus* system as presented in Zheng (2002).

*AnswerBus* was evaluated with 200 questions of the TREC 8 corpus (Zheng, 2002). In comparison with other systems such as *START*, which were also tested with the same questions, it had a better performance. However, one disadvantage is that the answer always consists of one sentence, which is maybe enough for factual knowledge but not enough for questions that require more complex answers.

### 3.6. The DeepQA Project

One of the most popular QA systems currently is the *DeepQA* project or *Watson* as it is broadly known – after the founder of the IBM Corporation, Thomas Watson (Lapshin, 2012). In 2005 the IBM Corporation launched the *DeepQA* project; a project aiming at an open-domain QA system able to be adapted to various fields. The main goal of the project was to create a system able to participate in the famous American TV show *Jeopardy!*. In 2011 the developed QA system took part in several *Jeopardy!* shows, defeating the best human contestants.

As presented in Lapshin (2012) the special characteristic of *Watson* is that it is able to adapt, in other words to add new components, at each stage of the operation. The process consists of

the following stages (Figure 6). Firstly, during the question analysis, the query is classified, meaning that its category is being determined. The categories to which a question can belong are defined within the *Watson* system itself. After that, the program determines the focus of the question, which is the part of the question which replaced by the answer will make the question into a sensible statement. At last, the lexical type of the question has to be determined, meaning that the part of the question which characterizes the answer type has to be found, e.g. if the question is a *when*-question, then the answer type will have to be a time or date. After the analysis of the question has been completed, *Watson* decomposes the query into smaller parts, using rule-based and statistical machine-learning algorithms to do so. The next step is the generation of answer hypotheses. To do this, knowledge sources are searched. Knowledge sources can be anything from unstructured webpages to semi-structured ones (e.g. *Wikipedia*) and to structured knowledge bases (e.g. *RDF storage*). At the beginning, an initial search inside these sources takes place in order to generate some candidate answers; some hypotheses. These hypotheses are then formatted to fit the answer type. The different sets of hypotheses for each question are then processed further with the “soft filtering”. The “soft filtering” applies simple, fast algorithms on those hypotheses to create a shortlist with candidate answers. One such algorithm is computing the likelihood of a candidate answer to be of the lexical type defined during the question analysis. In that way, the number of candidate answers to be forwarded is reduced and the accumulation of errors is avoided. These hypothetical answers are then fed into evidence-scoring algorithms, whose objective is to obtain evidence from the knowledge sources that this is indeed the right answer and to give a score to each of those potential answers. At the last stage of the *Watson* system, the similar hypotheses are combined into one and this one is normalized in order to be grammatical. Machine-learning algorithms are then responsible for the ranking of each of those answers. The one with the highest ranking is the answer to be returned. The modern *Watson* system can process a query within 3 to 5 seconds – of course, this depends strongly on the resources; the “normal” open-source *Watson* could take longer.

*Watson* is considered to be one of most promising QA systems. Its main advantages of being open-domain, relying not only on databases but also on the web and being easily adjustable for different operations are the features which make it unique.



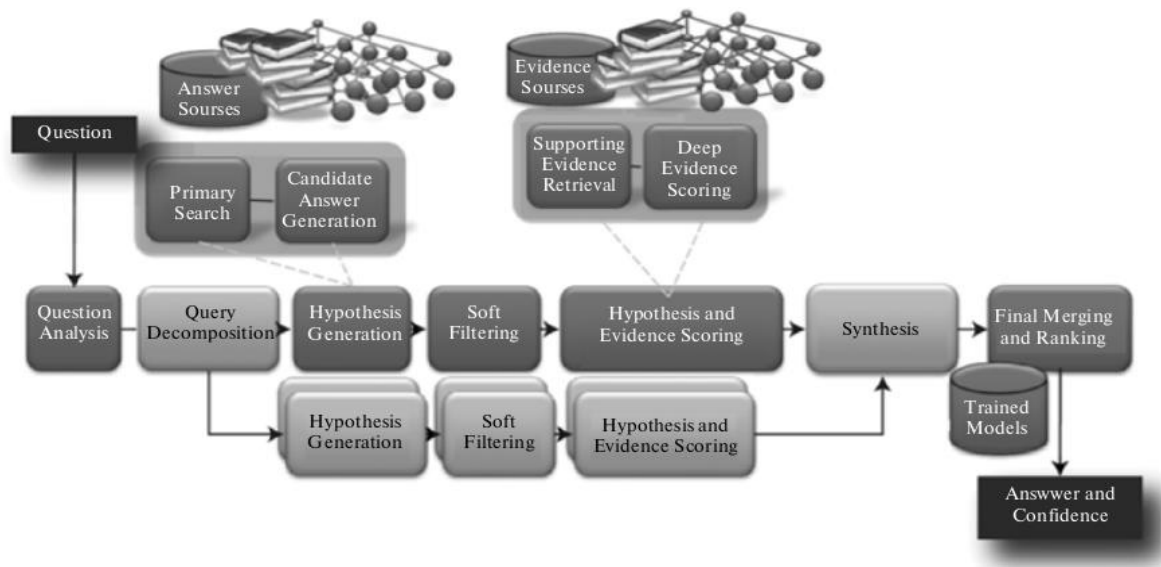


Figure 6: Overview of the *Watson* system as presented in Lapshin (2012).

### 3.7. Other approaches for the implementation of QA systems

The implementation of a QA system is a complex task requiring the stepwise development of its components and their harmonic combination into a working system. As it was made clear before, there are different approaches how a QA system can be implemented and there are also various ways of dealing with the different aspects of the question answering task. An ideal system would be the one that combines the advantages of all systems in one and keeps the limitations as few as possible. Therefore, in this section it is worth presenting some systems that use other, less-common ways for implementing different parts of the QA system. These ideas might be more suitable for certain types of QA systems than for others.

#### 3.7.1. Ontology-based QA systems

As already described most QA systems make some kind of semantic analysis at the question analysis stage or at the document preprocessing/analysis stage or at both. Linguistically, this semantic analysis can roughly be distinguished in two categories: the lexical semantics and the structural semantics analysis. Lexical semantics are for example the synonyms that are extracted for every word of the query while structural semantics would be the logical representation of the whole query in terms of First Order Logic or some other semantics-logic theory. For the part of the lexical semantics one can make use of a word net, e.g. to find out the synonyms, or make use of an ontology. Ontology refers to the sets of categories that are



used for the classification of a word or of a whole question within one specific domain (Lapshin, 2012). Such a system will be analyzed in this subsection.

It should be noted that although the ontology-based system that will be presented here is not using the common techniques that other ontology-based systems are using, it is still domain-specific in its application. The ontology used is inevitably defining what the domain of application of the system will be. Therefore, such systems are not to be compared with the other modern QA systems presented earlier but can prove useful for certain areas where domain-specificity is desired or even required.

A system based on an ontology is the one introduced by Athira, Sreeja and Reghuraj (2013). They propose a system which decomposes a complex query into a set of less complex queries using an ontology and a morphological expansion. They use ontology-based semantic labeling and domain knowledge in order to analyze the questions and to find the answers. Although most of the above presented modern systems use either syntactic and semantic analysis or ontology processing for the answer retrieval, their system implements all three modules based on a hybrid approach, which processes the question and the answer modules using semantic and syntactic approaches but a domain ontology is used for relation extraction and identification (Figure 7). As described in Athira, et al. (2013), the pipeline consists of three stages; the question processing, the document retrieval and the answer extraction and formulation. During the question processing stage, a query analyzer performs the syntactic and semantic analysis of the question. For the syntactic one they are using the Stanford CoreNLP tool. In the semantic analysis part, semantic role labeling is applied so that dependencies or other restrictions are already applying when the document retrieval step will be executed. Moreover, the question needs to be classified. For this its focus needs to be identified, meaning the type of answer that the user is anticipating with his answer. This can be determined by the question word itself or a combination of the question word and its succeeding words. To the classification of the question belongs as well the detection of phrases or clauses that contain information relevant to the expected answer. Moreover, the semantic roles have to be incorporated into a semantic frame for better retrieval. At the last stage of the question analysis, the query has to be reformulated by adding domain knowledge and ontological information. The ontology used contains the vocabulary of a specific area as well as rules for combining the relations between the original word and its reformulation. During the document retrieval stage, the system selects relevant documents from a domain

specific repository. These documents are further being processed to extract the candidate answer set. They undergo syntactic analysis using the usual NLP techniques. After that a semantic analysis tool performs shallow parsing to find the semantic phrases or clauses and map them to semantic frames, like it happened in the question analysis stage. The last stage of answer extraction uses filtering. The filtering is based on the combination of the question classification and document processing modules, which then produces the candidate answer set. The answer set produced is ranked based on the semantic similarity between the question and the answer frame. In case the answer set does not contain the answer directly, a specific answer has to be generated.

For their evaluation they are using a random set of documents collected over a specific domain. The questions are evaluated for their correctness, relevance and completeness. The system is also evaluated for efficiency using the recall measure. In their evaluation they observe that the system can filter semantically matching sentences and their relations and therefore rank the correct answers higher than the incorrect with an over 80% accuracy.

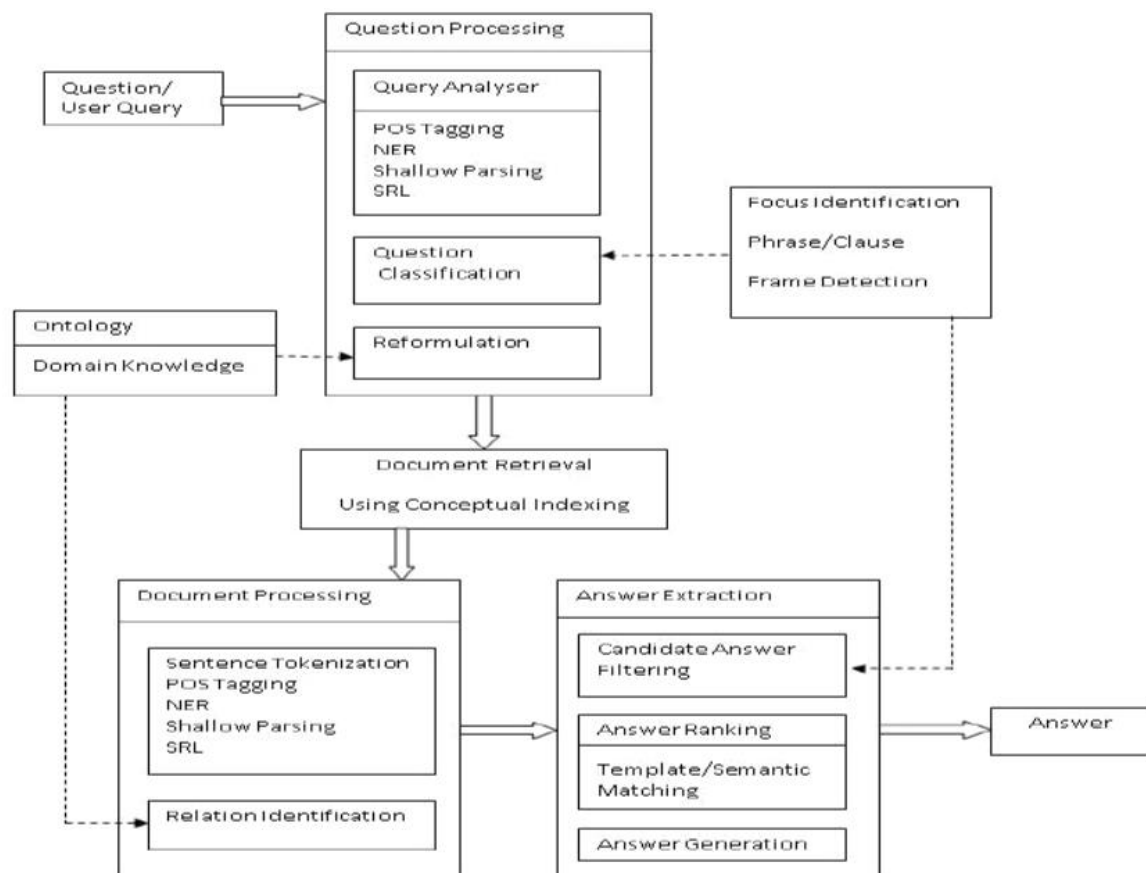


Figure 7: Overview of the ontology-based system as presented by Athira, Sreeja and Reghuraj (2013).

### 3.7.2. QA systems using information from previous questions

Another new proposal concerning the implementation of the QA systems is the one offered by Mansoori and Hassanpour (2012). Their proposal is to boost up the QA system by searching in the archive of the previous questions for keywords that are semantically related to the answer type of the current question of the user (Figure 8). Generally, question similarity was first introduced with the FAQ data and was later integrated into the QA data. The proposal can be easily illustrated by example (4) (Mansoori & Hassanpour, 2012:188):

- (4) Q1: Which sweeteners are used in ice cream?  
Q2: What ingredients are used in an ice cream?

This dialogue could either represent a multi-user system, where different users ask questions belonging semantically to the same domain, or also a single user asking a series of questions about the same domain. In this case, the more specific question Q1 could provide some useful and helpful facts for the more general question Q2, since Q1 is a part of Q2's answer.

In their approach they propose two ways how the reuse of a fact could boost up the QA performance. Firstly, the fact could be used in the query expansion, in other words during the question analysis. There are usually two ways how a query can be expanded in order to include a broader semantic spectrum. The first one is based on semantic knowledge resources such as WordNet which offer alternatives semantically belonging to the domain of each of the initial words of the query. The second one called *blind relevance feedback* (Mansoori & Hassanpour, 2012) “analyzes the top  $N$  retrieved documents from the initial query run and adds new terms to the query based on the standard *Rocchio term weighting method*”. However, they use a third technique where the archive of the previous stated questions is searched and within this the words that are semantically related to the expected answer type of the current question are being extracted. The extracted words are then used in the query formulation as additional query constraints. By using previous facts in their query expansion they are expecting improvements in the relevance, correctness and conciseness measures.

Their second proposal in order to boost up the QA performance is to present the (previously used) fact as a candidate answer to the user question, when it is semantically matching and appropriate. By this they expect improvements in the speed of the system response.

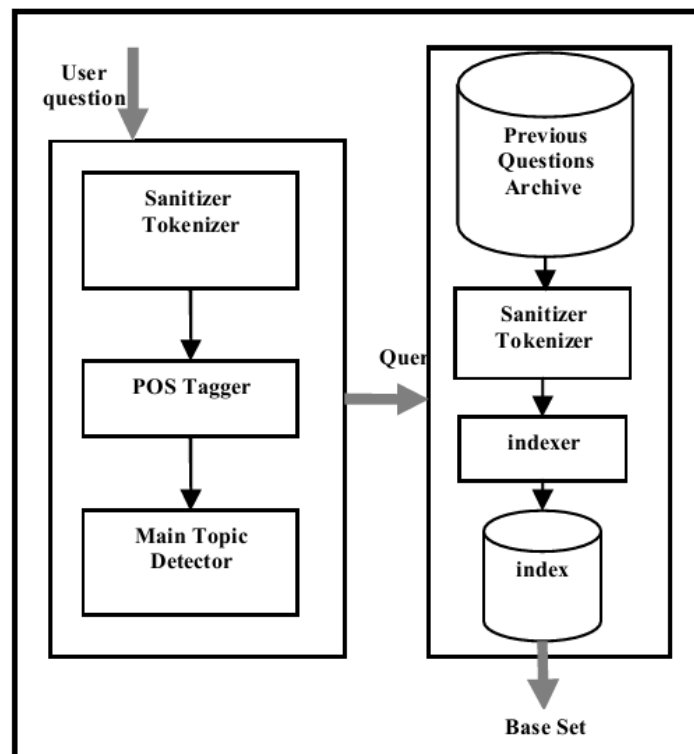


Figure 8: Architecture combining the current question and the questions in the archive as presented by Mansoori & Hassanpour (2012).

## **4. The Nuance Communications Deutschland GmbH QA system (the ‘thesis version’)**

### **4.1. Introduction to the ‘thesis version’ of the Nuance QA system**

As already mentioned in the introduction, the main focus of this thesis is the implementation of a state-of-the-art German QA system. The German QA system is based on a version of the English QA system developed within the QA Core project of Nuance. Everything mentioned from this point on refers to this ‘thesis version’. This Nuance QA system is an open-domain system which can be applied in various fields such as the automotive, the healthcare and the professional services domain. The automotive application refers to a system that is integrated in the car to help the driver when information of the user manual is needed, as it will be analyzed more detailed below. The healthcare domain involves cases where for example a doctor or another medical assistant wants to know in which cases a specific symptom appears or what the active substance of a medicine is: instead of a long “manual” search in books, articles and the web, a QA system can deliver an exact answer, relying on the very documents that the “manual” search would also use. The professional services domain, e.g. an insurance company or a bank, can profit from such a system if the employees do not need to search within different documents for specific client questions or even if the clients themselves can pose questions and get the information needed.

Within the wide spectrum of the QA systems presented above, the Nuance QA system uses a hybrid approach for its implementation, making use not only of linguistic techniques (e.g. parsing) but also of statistical-machine learning ones (e.g. *n*-gram mining). For the other dimensions (applications, users, question types) of the QA task it is worth noting the following. As far as its application is concerned, the sources within which the search for the answer takes place are documents or websites, thus unstructured data. This means that the range of the search documents is corresponding to those predefined documents or websites and does not include a broadly open collection of documents, like a search over the entire web would do. The system is domain-independent since there is no domain-specific annotation or training and can be therefore easily applied on sources of any desired content. The fact that the system focuses each time on predefined documents or websites (of a particular domain) makes it more efficient in that the passage retrieval can be faster since irrelevant sources do not have to be searched. Moreover, such an approach improves accuracy since there is less “irrelevant” information involved and therefore the chances for

“mishits” are reduced. The QA system is meant for various kinds of users, from everyday users wishing to look up information to “professional users”, who are using the system for their work. The questions that can be handled are mainly factual but include various linguistic phenomena, such as *wh*-questions, *yes/no* questions, embedded and passive voice questions.

One of the main advantages of the Nuance QA pipeline is its language configurability. The system has been built with an architecture (see section 4.2.) which facilitates its adaptation to other languages. This adaptation could be achieved by the use of resources of the other language, which would then be integrated into the language dependent parts of the system, while the language independent parts would not need any additional adjustment. Whether this is indeed the case and to what extent that is, is one of the main questions of the thesis, as presented in the introduction.

Having seen the general characteristics of the “thesis version’ of the Nuance QA system, the domain of application relevant to the thesis may be presented; the automotive domain. The existing QA application can be used in the car. We can imagine the following scenario: a driver gets some kind of a warning or information message and wants to look up what it means or how serious it is. The driver would have to pull off and look it up in the user manual. However, with an integrated QA system, the driver would just have to speak the question out loud and the system would look up the message in the user manual for him. Then, the answer would be read out loud so that the driver keeps his attention throughout this process.

This application of the system is integrated in an online Audi A4 user manual found at <http://www.audihelp.com/>. In the Nuance modified version of this website we can find the *Nina* interface within which we can ask questions by typing them in the question bar provided. *Nina* will return the best answer she could find but also other relevant answers with lower scores. Let us see an example question of how the system works (Figure 9):



Figure 9: Asking Nuance Nina for an answer to the question 'Where can I find the vehicle identification number?'.

Nina will then give back the following answer as the best match. It should be noted that we get back not only the exact answer to our question but also the passage of the manual where the answer was found and which is therefore the evidence for the correctness of the answer (Figure 10).

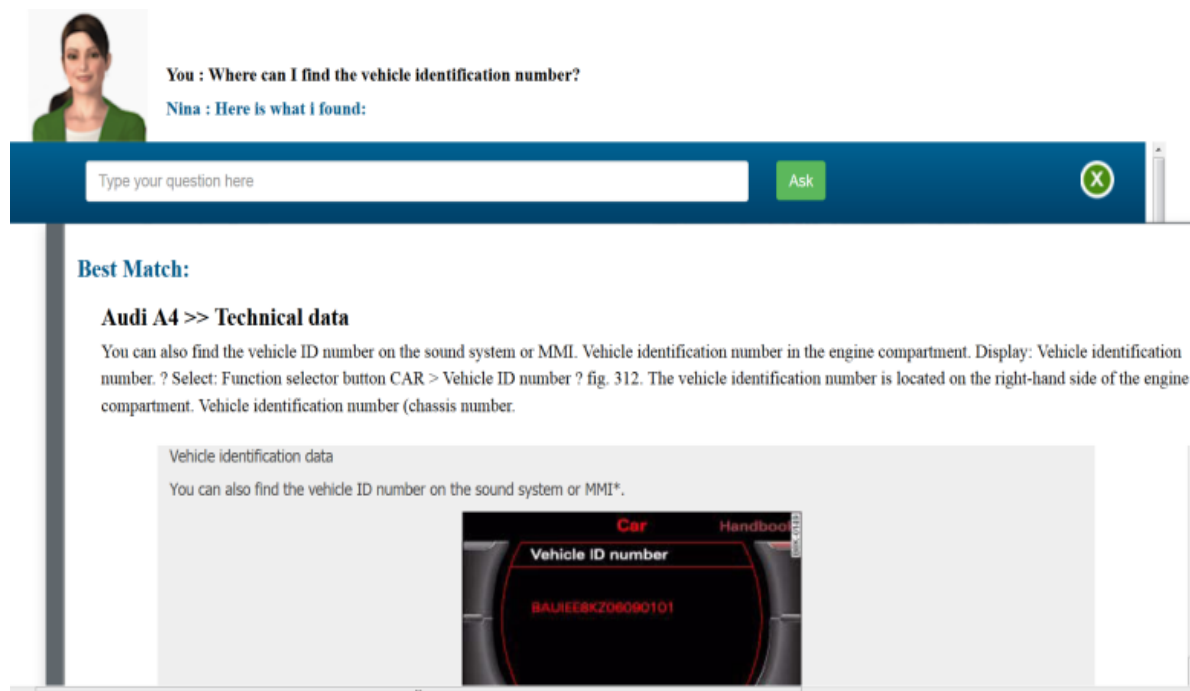


Figure 10: Nina giving back the best answer found.

#### 4.2.Components and architecture of the ‘thesis version’ Nuance QA system

As mentioned before, the main components of all QA systems are more or less the same; their exact implementation, the order of action and the concrete pipeline of the systems differ however. Therefore, it is necessary to understand how the existing English Nuance QA system works. Its main components are document preprocessing (leading to the *Ingested Document* of Figure 11), question analysis, passage search and retrieval, answer extraction and ranking, and answer presentation. The high level pipeline of the ‘thesis version’ of the system can be seen in Figure 11:

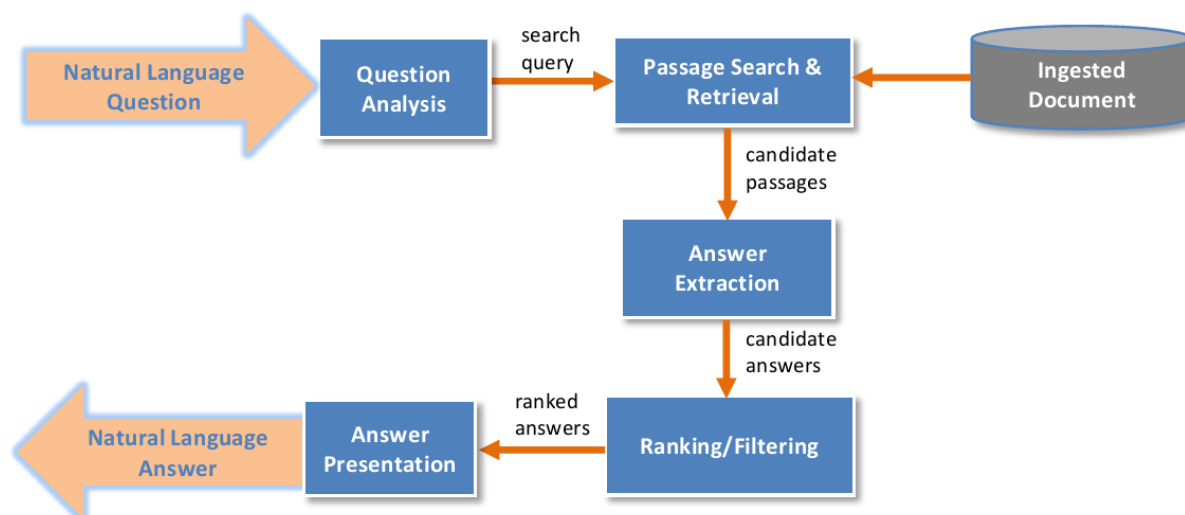


Figure 11: Overview of the ‘thesis version’ Nuance QA pipeline.

#### 4.2.1. Document preprocessing

The document preprocessing is the component leading to the ingestion of the document (top right corner of Figure 11). This component refers to the “preparation” of the source of the answers so that this step can be skipped at runtime and therefore the response time can be shorter. The current system is based, as already mentioned, on documents or websites, which means that the source can be in various formats. The application on the Audi A4 user manual involves a website and is therefore in an HTML format. This HTML document is being preprocessed in three stages (Figure 12). At the first stage, the scraping of the document takes place. During the scraping or crawling procedure an HTML translator tool runs recursively on the HTML website and converts the HTML document into XMLs of a Nuance internal specified format. In this way, the system acquires not only the given (initial) website but also all others linked to it. All HTML documents originating from the initial one are now many different XMLs. When this is completed, the annotation of those XMLs begins. The annotation is the procedure in which additional information or metadata is added to the documents. By adding this metadata, the search can be improved because it can take into account not only the initial elements found within the documents but also the additional information provided by the annotation. An annotator is creating an annotated file set including the separate XMLs. There are various annotators available such as shallow parser annotators, dictionary annotators and semantic ones. These can be used alone or sequentially. The English QA system is mainly using a semantic concept annotator (the annotator is



available within the company) through which the concept/word sense of each word of the document is extracted as additional information. The annotator applies a certain word sense disambiguation technique in order to tag each word with its right sense. In order, however, to apply the annotation, each word has to be processed separately, which means that a tokenization of the document is necessary. The tokenization is done by a parser; in this case, the Clear Parser (Clear Parser Documentation, 2012; the use of the parser will be analyzed below). Once the annotated file set is ready, the process of indexing starts. Indexing is a (IR-) process during which an *index*, meaning a position of a document containing a specific element, is stored as such within the system. The purpose of storing this position is to optimize speed and performance in the later stages of the retrieval. Without an index, the search component would have to scan every document of the set which would require much more time; having an index can directly lead to the relevant document. To make this comparison even clearer: an index of 10.000 documents can be searched within milliseconds, a scan of every word in 10.000 documents could take hours (Wikipedia: Search engine indexing, 2016). So, the annotated file set is fed as input into the *Apache Lucene Indexer* which in turn creates indexes for the documents/pages and the sentences within these documents. The *Apache Lucene Indexer* (Wikipedia: Lucene, 2016) is an open-source information retrieval software library and its main architecture relies on the idea that a document contains fields of text. This allows *Lucene* to be independent of the input file format, something being considered one of its most significant advantages.

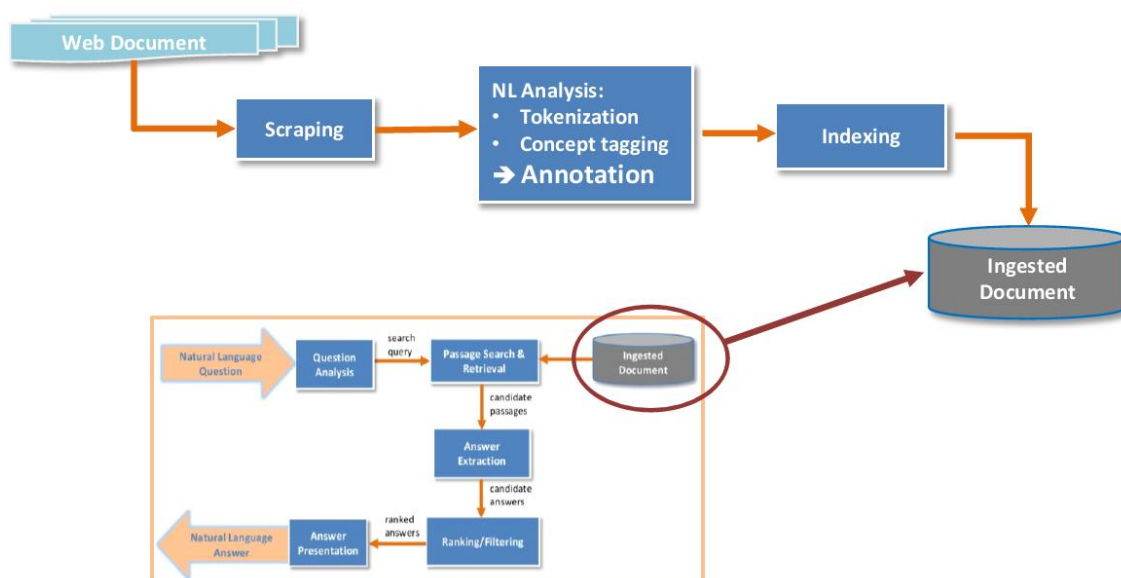


Figure 12: Structure of the document preprocessing component of the Nuance pipeline (‘thesis version’).

#### 4.2.2. Question analysis

The question analysis component is the first one in the actual pipeline (see Figure 11). It is the component where the natural language (NL) question is turned into a search query, suitable for the rest of the system to process. In order for the NL question to be turned into a search query, it is necessary to analyze the question based on different criteria (Figure 13).

First of all, a parser is utilized to extract the syntactic relations of the question. This is done by a dependency transition-based parser, the Clear Parser (Clear Parser Documentation, 2012). In order to get a better picture about the kind of the parser, its main characteristics should be presented. A dependency parser is based on the dependency grammar (DG), which assumes that the words in a sentence are connected with each other with directed links called dependencies (Van Valin, 2001). The finite verb of the sentence is taken to be the structural center of the clause and all other words depend (are linked) on it directly or indirectly. Structure is determined by the relation between a word (the head) and its dependents. DGs are to be distinguished from phrase structure grammars (constituent grammars) since the latter assume phrasal nodes to reflect the relations between the words of a sentence (for more details see Van Valin, 2001). This means that, roughly, a dependency parser gets a sentence as input and outputs the dependencies found between the words of the sentence. There are mainly two dependency parsing approaches. The first one is the transition-based approach in which the parsing result is based on the “local optima” (on the local transitions from one word to the next with the highest scores) and the parse history is used to predict the next transition (Choi & Palmer, 2011a). The other approach is the graph-based one in which the parsing result is based on the “global optimum” (on the version of the structure tree which yields the highest score among all other trees) from a complete graph (Choi & Palmer, 2011a). Recently, it was observed that the transition-based approach is preferred as it is noticeably faster than the graph-based one (Choi & Nicolov, 2009). On the basis of all that, the Clear Parser used in the current system is using the shift-eager algorithm (for more details on the algorithm see Choi & Nicolov, 2009) and the shift-pop algorithm (for more details on the algorithm see Choi & Palmer, 2011a), which show “near state-of-the-art performance in both speed and accuracy” as suggested within the Clear Parser Documentation (2012).

The parser applies tokenization, entity-tagging and POS-tagging and finally provides the dependency relations formed between the words of the question. To express the dependency relations each word is marked with its head. Now the job of the parser is done. These

relations are transformed to a dependency graph based on a certain transforming mechanism. Then a predefined language model defines, given the dependency graph, with what criteria and linguistic assumptions the graph will be turned into the linguistic structure *head(s)-arguments-modifiers*. The language model is the component where elements like *stopwords* are defined and semantic role labels are banned or boosted. Based on the graph and the language model, a structure of the form *head(s)-arguments-modifiers* can be extracted to represent the question. Given the *head* the intent of the question can be recognized. The intent expresses the kind of information that the question triggers. If the intent is not within the predefined intents, then a general *search* intent is being defined. Also, the mood is identified, meaning whether it is a *wh*-question or a *yes-no* question. Depending on that the answer type is predicted, e.g. a *when*-question requires a time/date as answer type.

The question which has been transformed into a query consisting of the three above mentioned units is not ready for the search yet. The query has to be further expanded so that the search is not restricted to this exact query and to those exact words. Therefore, each word of the query gets some additional information, following the logic of annotation described before in the document preprocessing. This kind of annotation can either rely on semantics or on morphological analysis or both. In the first case of the semantics, the analysis can involve concepts or a deeper semantic representation. Within the concepts/lexical semantics, each word is mapped to a concept based on some kind of a semantic word net. The English QA system was first using WordNet for this purpose. WordNet (WordNet: A lexical Database for English, 2015) is a large lexical database of the English language. The different parts of speech are grouped together into sets of cognitive synonyms, each expressing a distinct concept. These groups are interlinked by means of conceptual-semantic and lexical relations. Now, however, the English system relies for the concept/lexical semantics expansion on the Nuance-internal semantic concept annotator mentioned before (see section 4.2.1). This annotator applies a word sense disambiguation technique and thus the words are expanded with their disambiguated word sense. As far as the creation of a deeper semantic representation is concerned, the syntactic representation coming from the parser is mapped to a semantic one, using rewrite rules (for more details see Crouch & King, 2006) of a similar approach as the XLE’s transfer system (for more details see Crouch, 2005). The semantic representation produced is further mapped to a semantic graph. Although this semantic tool exists, it is not being used in the ‘thesis version’ and in the system under investigation. It is interesting to see how much can be achieved with how little. Therefore, it will not be

analyzed with more details. Apart from semantic, the annotation can also be morphological. One variant of the morphological annotation is lemmatization. Each word is expanded by its lemma. In this way, the search can also find hits that are different parts of speech from the current word found in the question.

After the query expansion has been completed in the above mentioned ways, the question analysis component constructs the final search query based on the schema *head(s)-arguments-modifiers* and including the expanded words. The schema is necessary so that the words that are marked as heads and arguments get a higher weight in the search query (‘+’ annotation in Figure 13) and the words that are marked as modifiers get a lesser weight within the search query. This means that the search will definitely retrieve the documents if the words marked with higher weights are found in them. On the other hand, the absence of the words with a lesser weight from these documents will not exclude those from the retrieval but the documents containing those “optional” words will be boosted in the score. The search query has the boolean form discussed in section 2.

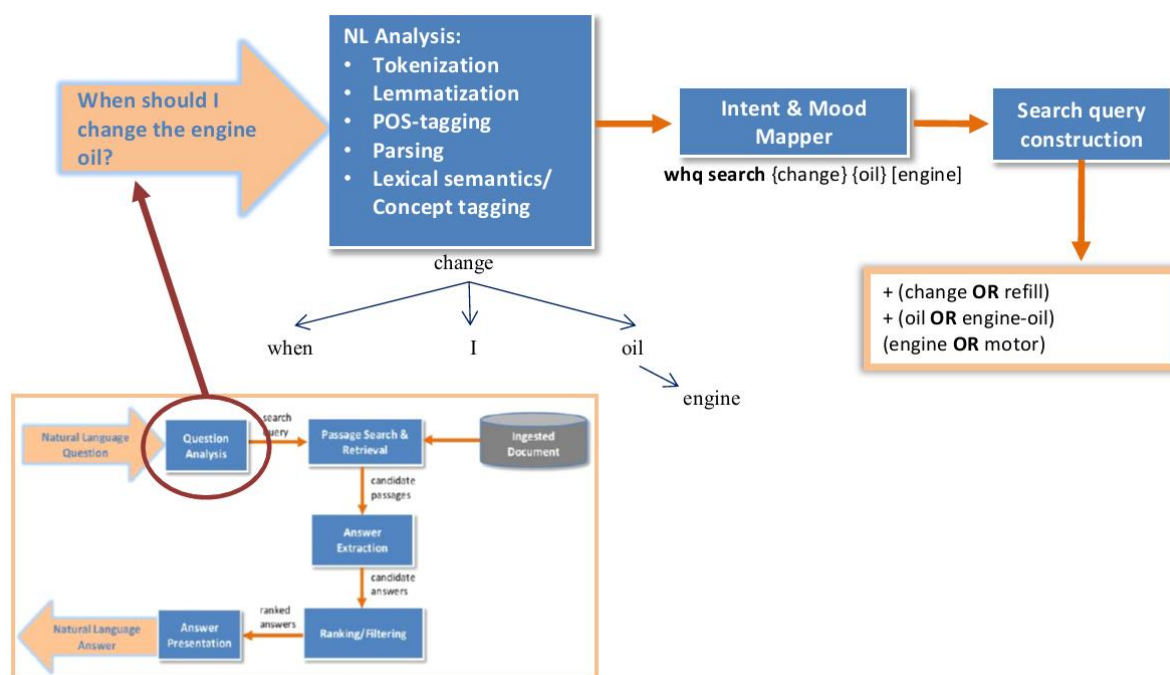


Figure 13: Structure of the question analysis component of the Nuance pipeline (‘thesis version’).

#### 4.2.3. Passage Search and Retrieval

At this point the search query is already formulated and the passage retrieval component has to look for it by using the *Lucene Search*, a component of the *Apache Lucene Indexer*

mentioned before, which based on the index searches for documents. There are two passes necessary. In the first pass, *Lucene* finds relevant pages and in the second pass, it finds relevant sentences (passages) within those pages. For this, the document IDs retrieved from the first pass are added to the search query so that the second pass can look within those exact documents for the search query.

Each candidate passage gets a score depending on the coverage of *head(s)-arguments-modifiers* discussed before; if all of them are found within the passage (or at least the two “high-weight” ones), then the passage gets a higher score than when only the head or some argument is found. Additionally, each passage gets a score depending on whether the expected answer type is found in it (Figure 14).

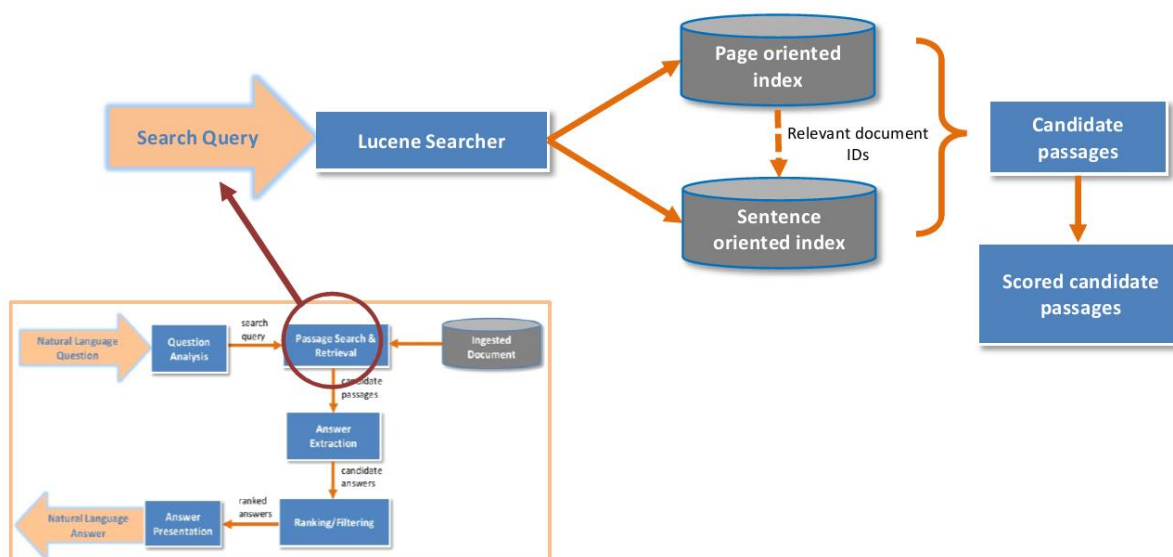


Figure 14: Structure of the passage retrieval component of the Nuance pipeline (‘thesis version’).

#### 4.2.4. Answer Extraction

The scored candidate passages are filtered so that those with scores below a predefined threshold can be removed. The extraction component identifies further the relevant sentences/sub-sections within those passages and creates an answer item. The answer item is presented along with the passage it was taken from as a pair: *pair {answer item, passage}*. In this way, the answer extraction component outputs a set of pairs of answer items and passages. In case there are multiple answer item/passage pairs that are related with each other, then the component merges them into one pair.

#### 4.2.5. Answer Ranking/Filtering

At this point of the pipeline, the final scoring of the answer items defines the final answers to be chosen. Each candidate answer item gets a score which is calculated by a specific algorithm that for confidentiality reasons will not be presented here. Generally, the algorithm is based on the score of the passage the answer item belongs to and the section inside the initial document from which the passage was extracted. In this way, the candidate answers are ranked based on their scores. After scoring each answer item, the component filters out candidate answer items which get a score under a predefined threshold. With the remaining candidate answers a list of  $N$ -best answers is created, where the answer with the highest score in the ranking is first in the list. The structure of the answer ranking/filtering component is shown in Figure 15.

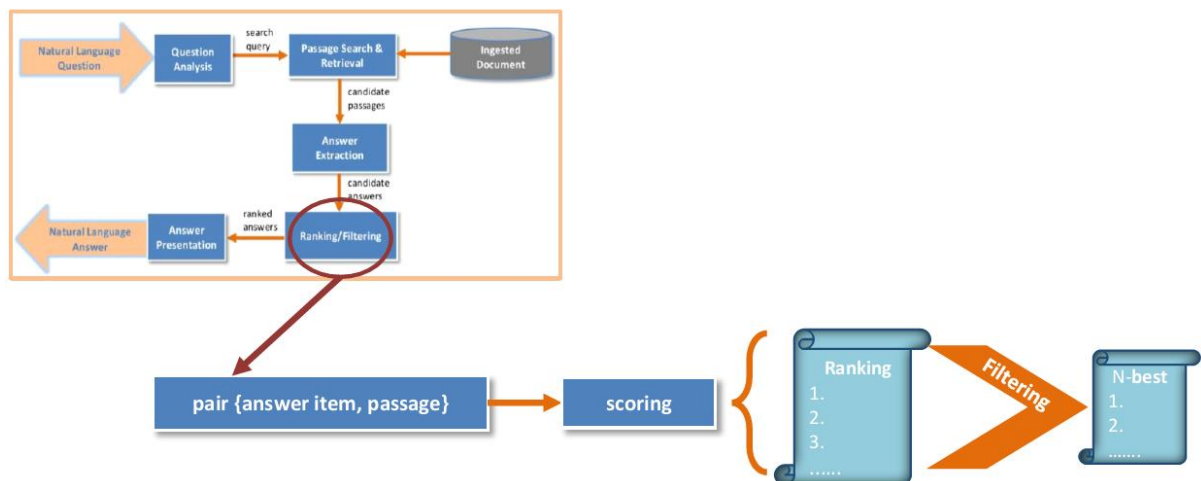


Figure 15: Structure of the Answer Ranking/Filtering component of the Nuance pipeline (‘thesis version’).

#### 4.2.6. Answer presentation

This last stage of the pipeline delivers the output to the user. The answers that are inside the  $N$ -best list created from the previous component are presented to the user in an HTML form. The answer at the top of the list is presented as “Best match”.

## 5. Implementation of the German QA system

The implementation of the German QA system was based on the porting of the ‘thesis version’ English Nuance QA system to German. The successful adaptation of the QA system will prove that the hypothesis that a pipeline designed to be easily language configurable and adjustable to different languages is correct. If not only the implementation succeeds but also the test results show a satisfactory performance, then the second question will also be confirmed: the algorithms used within the system are not language-restrictive and can achieve similar results for more languages. In this case, the last and main goal of the thesis will have been met: a state-of-the-art German QA system.

For the implementation of the German QA system it was necessary to find a source very similar to the English one, i.e. the online Audi A4 user manual, so that the evaluation and comparison of the German system with the English one is reliable. Also, since the English system was applied on a HTML source, the German system had to be in HTML as well. Since there was no Audi A4 user manual available online in an HTML form, an online user manual of Audi A3 found at <http://a3bb.s4net.de/> was selected. Nevertheless, if ones look closer, the text of this manual matches almost one-to-one to the text of the English manual, which makes it ideal for the purpose of this thesis.

### 5.1. Methods and implementation tools

For implementing the German QA system some of the above mentioned components had to be used as tools and some others as basis for the actual further implementation. The components having a language independent character, meaning the ones that do not involve any deep natural language processing, could be used as given tools for the system. However, it was still necessary to make sure that they are compatible with the rest modified components. Thus, the passage search and retrieval, answer extraction, answer ranking and filtering, and answer presentation were used unchanged. Whether the algorithms within those tools are adequate for the German system and indeed do not need any further implementation or adaptation, remains to be seen after the evaluation of the results. For now, those components were tools and no actual subject of the implementation. On the other hand, the components involving NLP techniques and thus having a language dependent character, i.e. the document preprocessing and the question analysis, were the ones needing a further implementation for German, always having as basis the English corresponding components.



Although the general architecture of the system was also taken from the existing system, it was necessary to make some refactoring on it in order to account for an architecture which implements both the English and the German system at the same time.

For the implementation which will follow it is important to keep in mind that the pipeline of the system presented above remained unchanged.

## 5.2. Implementation

The main implementation part refers to the document preprocessing component and the question analysis. These components are the ones that depend strongly on the language used, since they involve linguistic analysis. As Figure 16 shows, the natural language analysis involved in those two components consists of consecutive levels with the lower levels being absolutely necessary for the processing and quite simpler in their implementation and the higher levels being more complex. As discussed in section 4.2.2., the ‘thesis version’ English QA system is not using the semantics level, as the overall goal is to see how much can be achieved with how little. This means that the German QA system implemented also does not include a semantics analysis tool so that the systems are more easily comparable.

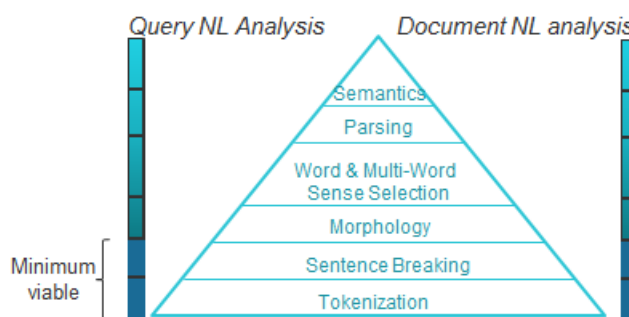


Figure 16: Pyramid illustrating the natural language analysis stages involved in the language dependent components of the pipeline (special thanks to Dick Crouch for the design of the pyramid).

### 5.2.1. German document preprocessing

As explained above there are three stages in the document preprocessing (Figure 17). Since the German system also uses a website as source of the answers, the first stage of the document scraping had to be applied on the HTML just like for English. However, the English HTML translation tool had to be extended in order to be able to crawl this new website as not all websites follow the same principles and structure. The German website is using CSS labels in addition to the standard HTML tags and therefore the English converter had to be extended in order to be able to convert those as well. The new converter can



efficiently go through the website and create the Nuance internal XML format. For this modification of the converter it was necessary to “decode” the tree-like structure of the CSS-labeled HTML and build up a new tree-like structure which includes the new tags of the Nuance internal XML format.

The second stage of the preprocessing relies on the annotation of the document. Since the semantic concept annotator used for English cannot be used for German and since there was no other similar annotator for German, there was the need to come up with another kind of annotation which would still extract enough metadata from the documents so that the search later succeeds. Given the restricted time frame of this thesis it was also important to adopt a solution which would be implementable in a timely manner. For these reasons the morphological procedure of lemmatization was chosen. Lemmatization is the process of extracting the lemma of a word, in other words the canonical form or the dictionary form of a word (Wikipedia: Lemma (morphology), 2016). For example, the lemma of the word ‘meeting’ is ‘meet’ and the lemma of the word ‘mice’ is ‘mouse’. Lemmatization is different than stemming in that the former extracts the canonical form of the word while the latter cuts off the inflectional ending and just returns the word root. Choosing to annotate each word in the documents with its own lemma has certain advantages. It makes sure that the passage retrieval component will still match the current word (and passage) even if: a) the part of speech of a word used in the question is not the same with the part of the speech of the word that is found inside the document and b) a phrase used in the question is broken down in its elements while the same phrase within the document is a compound word. Both cases are very common in German, a language with rich morphology. Of course, this requires that the question also has a similar analysis, to which point I will come back in the next subsection. To make this argument clearer, let us see example (5):

(5)

Question:     wie kann ich die Batterie des Funkschlüssels ersetzen?  
                  how can I the battery of-the remote-control-key replace  
                  *How can I replace the battery of the remote control key?*

Document excerpt:

Case a):       Der Ersatz der Batterie des Funkschlüssels...  
                  the replacement-of-the battery of-the remote-control-key....  
                  *The replacement of the remote control key battery...*

Case b): Die **Funkschlüsselbatterie** können Sie ersetzen, indem...  
The remote-control-key-battery can you replace, by...  
*You can replace the battery of the remote key by...*

Although the question contains the verb *ersetzen* ‘replace’ (marked with orange), the document *case a*) would still be retrieved –which contains the noun *Ersatz* ‘replacement’ (marked with orange) – because the lemma and therefore the annotation of *Ersatz* will be *ersetzen* ‘replace’. On the other hand, the document *case b* would also be retrieved. It will contain the annotation *Funkschlüssel+Batterie* (marked with blue) which will match to the words of the question *Batterie Funkschlüssels* (marked with blue).

Last but not least, the implementation of a lemmatizing tool could be within the desired time frame.

For the lemmatization it was decided to use the Finite State Morphology (FSM) (Beesley & Karttunen, 2003) and its compiling tool, Xerox Finite State Tool (XFST), whose licenses have been obtained from Nuance. Nuance has created its own versions of the tools (FST from now on) but the logic behind it is the same.

Briefly, FSM makes use of finite-state transducers in order to provide the morphological analysis of a given word in a language. A finite-state transducer is like a finite-state machine (network) only that it consists of two tapes; the input tape and the output tape. This means that it does not simple return an ACCEPT or a REJECT depending on whether the machine has accepted the given string like a finite-state machine does. A finite-state transducer compares a given input with its input tape and if the input is accepted, it returns the output of the output tape (Beesley & Karttunen, 2003). In this way, we can imagine of a transducer which has a word of the lexicon on its input tape (lower side of the transducer) and the morphological analysis of the word on its output tape (the upper side of the transducer). If we want to *look up* this word and see if it is part of the language and what is its morphological analysis, the input word will be matched to the lower side of the transducer and the word will be accepted, which will return the upper side of the transducer containing the morphological analysis of the word. This can also be applied vice versa: if we input the lemma of a word and some morphological features, the transducer will *look down* whether this input is accepted by its upper side and if so, it will return the inflected word of its lower side as output (generation). So, having the necessary lexicons and rules for a language, one can build such

finite-state transducers that do morphological analysis and generation for the language. This has been done by the Xerox Corporation for several languages, including German.

The decision to use the FST technology was based on the fact that it is a state-of-the-art system, offering a very good morphological analysis for many languages. Also by using this technology to develop the lemmatizer and not a language specific parsing tool it could be shown that there is not the necessity for a language specific parser at the document preprocessing stage and therefore the component can be applied to other languages just by changing the language that is being fed into FST and therefore to the lemmatizer.

In order to apply the lemmatization of each word, there is more than just the morphological analysis needed. Firstly, each document has to be tokenized, so that each word can be fed separately into the FST and output an analysis. The tokenization is done with the Xerox tokenization tool which is again based on the finite-state technology. After the document is split into tokens, each token is read and is applied on the FST (*look up*) to get back its morphological analysis. The point to be made here is that very often there is more than one analysis possible; a verb form can be indicative and imperative at the same time for example. To illustrate that, let us see the output of the FST for the word form *make* ‘do’ in (6):

- (6) machen+Verb+Subj+3P+Sg+Pres  
 machen+Verb+IndcSubj+1P+Sg+Pres  
 machen+Verb+Imp+2P+Sg  
 ^=machen+Verb+Subj+3P+Sg+Pres+V1  
 ^=machen+Verb+IndcSubj+1P+Sg+Pres+V1  
 ^=machen+Verb+Imp+2P+Sg

The symbols ^ mean that this verb could also be found as a separable verb with a prefix (*trennbare Verb*).

Since we are only interested in the lemma of the word, we do not have to keep apart such variants because all we need to extract is the first element of the above analysis. This is done with the use of regular expressions and the resulting lemma of each word is listed as the annotation of the word.

However, things get more complicated in some cases; two of them are well-known characteristics of the German language. The first case deals with compounds. German, a rich morphological language with a lot of compounding, offers a challenge as to how to deal with this phenomenon. The problem begins from the fact that a single compound can in most cases be analyzed in different ways depending on what one takes its elements to be. For example, in (7) the word form *Gaspedal* ‘acceleration pedal’ can have the two following analyses:

- (7) Gas+Noun+Neut+Sg^#Pedal+Noun+Neut+Sg+NomAccDat  
gasen+Verb^#Pedal+Noun+Neut+Sg+NomAccDat

This means that candidate lemmata and thus candidate annotations are *Gas+Pedal* or/and *gasen+Pedal* and that both need to be extracted. A pseudocode which captures this implementation with compounds is found in Pseudocode 1.

**Pseudocode 1:** Getting the lemmata out of a compound token

---

**Input:** token, lemmaList

**Output:** lemma(ta) of the token

morpho\_analysis = FST morphological analysis of the token

**FOR EACH** morpho analysis:

**IF** morpho\_analysis does not contain the '#' sign **THEN** //this means it is not a compound

        lemma = get first word before '+' from morpho\_analysis

        add lemma to lemmaList

**ELSE IF** morpho\_analysis contains the '#' sign **THEN** //this means it is a compound

        numberOfCompounds = count the subwords within the token

**FOR EACH** subword:

            stringOfSubword = get the string to which this subword belongs

            lemmaOfSubword = get first word before '+' from stringOfSubword

            add lemmaOfSubword to lemmaList

**ENDIF**

return lemmaList

---

The second challenge to face is the separable verbs with prefixes (*trennbare Verbe*), also very common in the German language. The lemmata extracted should account for the lemmata of the verb and its prefix and should combine the two of them in a single lemma. For illustration let us see how the FST output of the word *einschaltest* ‘(you) turn on’ looks like in (8):

- (8) ein^=schalten+Verb+Indc+2P+Sg+Pres+V2  
ein^=schalten+Verb+Subj+2P+Sg+Pres+V2

In this case the desired annotation of the word is the one which will extract the prefix *ein* and combine it with the lemma *schalten* to form the lemma *einschalten*.

Now, in cases where the two phenomena are combined, i.e. a compound noun consists of a separable verb with prefix and other words, all extraction mechanisms have to apply at once. An example of such a case is the word *Berganfaharrassistent* ‘slopes hold assistant’ in (9):

- (9) Berg+Noun+Masc+Sg^#an^=fahren+Verb^#Assistent+Noun+Masc+Sg+Nom  
bergen+Verb^#an^=fahren+Verb^#Assistent+Noun+Masc+Sg+Nom

In this case the compound noun consists of three elements: the noun *Berg* ‘mountain’, the separable verb *anfahren* ‘drive off’ and the noun *Assistent* ‘assistant’. This means that the three elements have to be recognized as such and their lemmata should be extracted into a unified representation.

Pseudocode 2 illustrates how the code from Pseudocode 1 was further modified in order to account for separable verbs within compounds:

**Pseudocode 2:** Getting the lemmata out of compounds which contain a separable verb (part of)

---

**Input:** token, lemmaList

**Output:** lemma(ta) of the token

morpho\_analysis = FST morphological analysis of the token

**FOR EACH** morpho analysis:

**IF** morpho\_analysis does not contain the '#' sign **THEN** //this means it is not a compound  
        lemma = get first word before '+' from morpho\_analysis  
        add lemma to lemmaList

**ELSE IF** morpho\_analysis contains the # sign **THEN** //this means it is a compound  
        numberOfCompounds = count the subwords within the token

**FOR EACH** subword:

            stringOfSubword = get the string to which this subword belongs

**IF** stringOfSubword contains a verb prefix **THEN**

                lemmaOfPrefix = get prefix

                lemmaOfSubword = get first word before '+' from stringOfSubword

                add lemmaOfPrefix + lemmaOfSubword to lemmaList

**ELSE**

                lemmaOfSubword = get first word before '+' from stringOfSubword

                add lemmaOfSubword to lemmaList

**ENDIF**

**ENDIF**

return lemmaList

---

After the lemmatization is completed, the extracted lemmata are added as annotations to the metadata of the corresponding word (Figure 17).

All in all, for the implementation of the lemmatizer it was important to successfully integrate the FST tool presented above and correctly extract the lemmata, even in cases that are more complicated than the common ones. The extraction was based among others on regular expressions. Lemmatization is a kind of annotation through which – as presented above – more complicated matches between the questions and the documents can succeed and its implementation can be achieved in a timely manner. However, it lacks a semantic background. Such a semantic background, e.g. semantic concepts used in the English system or word sense disambiguation, could succeed in even more complicated cases, where the different senses of a word play a decisive role for the matching of the question.

The third stage of the document preprocessing is the indexing of the documents. The *Apache Lucene Indexer* is also used in the German system and creates the indexes of the documents as already explained for English. This means that this indexing stage is language independent and could simply be used in the document preprocessing component as it is (Figure 17).

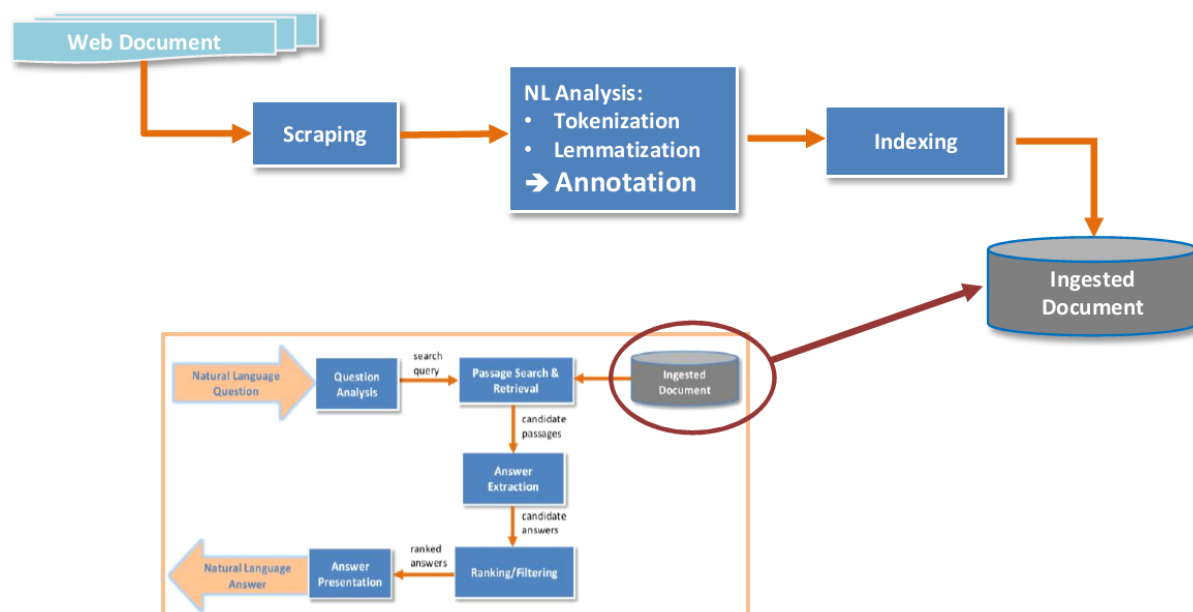


Figure 17: Structure of the document preprocessing component within the German QA system.

### 5.2.2. German question analysis

As presented before, this first stage of the actual pipeline is responsible for transforming the question to a query suitable for the passage search. In order to do so, the question has to be firstly analyzed by the parser.

For the German QA system the parser chosen to do the analysis is the Mate Parser (Bohnet & Nivre, 2012 and Bohnet & Kuhn, 2012). The Mate Parser is a transition-based dependency parser and is therefore very close to the Clear Parser used in the English system. This makes the performance of the two parsers and thus of the two systems comparable since one of the basic mechanisms used relies on similar algorithms. As mentioned before the parser requires some preliminary steps before doing the actual parsing. Some of those steps are lemmatization, POS-tagging and morphological tagging, which of course require that tokenization has also been applied on the input sentence. As stated in Bohnet and Nivre (2012:1455) “almost all dependency parsers presuppose that the words of the input sentence have been morphologically disambiguated using (at least) a part-of-speech tagger”. As supported in the same paper, parsers which include the POS-tagging and the actual parsing as a joint model, are faster, have a linear complexity and a higher accuracy. The Mate Parser is such a parser with the POS-tagging, the morphological tagging and the parsing happening jointly.

The steps before the POS-tagging, however, are separate components. Although the Mate Parser offers its own tool for lemmatization and suggests the OpenNLP Tools tokenizer (openNLP, 2016) for tokenization, it was considered better to use the same tools which were also used for the annotation of the documents within the document preprocessing stage since the query formed at this stage of the pipeline will have to be matched against those exact documents. In this way, inconsistencies or differences in the way special cases are being treated from two different tokenization and lemmatization tools can be avoided. For this purpose, first of all, the tokenizer of the FST presented before was implemented to run on the input of the parser and in that way, it returns the question tokenized. After that, the lemmatizer which was developed within the document preprocessing (see section 5.2.1) is applied on the output of the tokenizer, so that the output of this lemmatizer can serve as input for the tagging and finally the parsing.

The parsed sentence is returned as a group of dependencies with each word being identified by its head. Based on that, the further development includes the conversion of those dependencies into a dependency graph. In the graph each word is described further by its lemma, its POS-tag and its semantic role label; all information coming from the parser. As far as the lemma is concerned, it is worth noting that again the separable verbs (*trennbare Verbe*) have to be handled in a special way. This is because the question might contain the verb in its “separated” form which would mean that the created lemmatizer would find the lemma of the prefix and the lemma of the main verb separately, while the same verb might be found in its “unseparated” form within the document and therefore its lemma will correspond to the whole verb. In this case the two will not be able to match. For this reason, after converting the dependencies to a graph, it is necessary to check whether the question contains a separable verb with its prefix standing alone. If so, the prefix and its lemma have to be found and the lemma of the prefix has to be added to the lemma of the main verb. Example 10 following should make this clear:

- (10) Question: Wie **schalte** ich die Sitzheizung **an**?  
          how turn I the seat-heating on  
          *How do I turn on the seat heating?*

In this question the separable verb is found “separated” and the prefix is at the end of the sentence (separable verb marked with orange). Therefore, when converting the dependencies of this question to a graph, the lemma of the verb *schalte* will be identified as *schalten* and the lemma of the prefix *an* as *an*. However, that is not the desired result since this will not

match the lemma of the “unseparated” verb *anschalten* which might be found in the document. To avoid that, the lemma of the prefix should be added to the lemma of the verb stored in the graph. This mechanism will also prove useful for the further expansion of the query, which will be presented shortly. Pseudocode 3 shows how the implementation of this restriction looks like.

---

**Pseudocode 3:** Adjust verb node lemma if the verb is separable (part of the implementation)

---

**Input:** nodes of graph**Output:** void**FOR EACH** node:    **IF** the role label of the node is the one standing for verb prefixes **THEN**

parentNode = get the head of this node

lemmaOfThisNode = get the lemma of this node

set the lemmaOfParentNode to the lemmaOfThisNode+lemmaOfParentNode

**ENDIF**

---

In order to be able to extract the structure of the form *head(s)-arguments-modifiers* explained in section 4.2.2., an appropriate language model has to be built too. A language model consisting of German focused rules (e.g. *stopwords*, necessary role labels, etc.) and linguistic features was implemented for that purpose. Based on the graph and this language model the conversion to *head(s)-arguments-modifiers* can take place like in the English system. Again, the head defines the intent of the question and the mood is identified depending on the question word, if any. With all this, the search query for the German question analysis is formed but is not completed yet (Figure 18). What is still missing is its expansion so that it matches not only against documents containing these exact words of the query but also to other relevant ones.

The expansion of the query of the German QA system was implemented to include two kinds of information (annotations). The first kind of information is the one coming from the lemma. This means that each word of the query is further expanded with its lemma. This is very similar to the annotation of the documents. For this reason the right lemmatization of the separable verbs explained above is of great significance since the retrieved lemmata are the ones that are going to match the annotations of the documents (Figure 18).

The other kind of metadata used is of a semantic nature. Since a serious implementation of a deeper concept semantics approach – an approach involving word sense disambiguation as the English system has – was not possible due to the restricted time frame, it was decided to integrate at least some simple lexical semantics to account for a variation in the vocabulary used within the questions. The lexical semantics integrated was restricted to the synonyms of



each word of the search query. Among the available approaches for that kind of semantics, it was decided to stay in line with the initial tool used within the English system and use the German word net, the GermaNet (it should be stressed that GermaNet was obtained within the Academic Research License Agreement of the University of Konstanz and is integrated into this experimental German QA system only for the academic and research purposes of this thesis and will not be available to Nuance Communications Deutschland GmbH after the completion of this thesis). GermaNet (Hamp & Feldweg, 1997 and Henrich & Hinrichs, 2010) is a lexical-semantic net that relates parts of speech semantically by grouping lexical units that express the same concept into semantic nets called synsets. Thus, for each word of the search query the synsets are being constructed. However, in order to account for the lack of any deeper word sense disambiguation approach, there was a selection as to which of all synsets of the word are going to be further processed; only synsets belonging to certain ‘word classes’ are being forwarded.

GermaNet has some predefined word classes to which a synset can belong based on its ‘meaning’. For example, for nouns there are the classes of *Gefühl* ‘emotion’ or *Kommunikation* ‘communication’, to which the synsets of *Liebe* ‘love’ or *Blinklicht* ‘flashing light’ belong, respectively. For the purposes of the current application of the Audi A3 user manual the following generic classes were selected to be allowed to extract synonyms from:

- For nouns: *Artefakt* ‘artifact’, *Form* ‘form’, *Kommunikation* ‘communication’, *Mensch* ‘human’, *Motiv* ‘motivation’, *natPhaenomen* ‘natural phenomenon’, *Relation* ‘relation’ and *Substanz* ‘substance’.
- For verbs: *Allgemein* ‘general’, *Kommunikation* ‘communication’, *Lokation* ‘location’, *natPhaenomen* ‘natural phenomenon’, *Schöpfung* ‘creation’ and *Veränderung* ‘change’.
- For adjectives: *Allgemein* ‘general’, *Bewegung* ‘motion’, *natPhaenomen* ‘natural phenomenon’, *Relation* ‘relation’ and *Substanz* ‘substance’.

Such a restriction does not make the German QA system domain-specific, since the word classes selected are anyway generic, are not somehow related to car-domain concepts and contain far more notions than the Audi application would actually demand. On the other hand, the decision to allow only these classes offers a good background for some concept semantics to make up for the missing deeper concept tagging that the English QA system has and therefore does not need such restrictions.

Thus, to achieve this selection, the word class of each synset of a given word is being identified at the beginning of the analysis and only if it is one of the predefined ones, the lexical units of the synset are being extracted. The lexical units of all synsets are put into the list of synonyms of the word. Lexical units that are the same across synsets are added only once to the synonyms list. In the case that the word class of a synset is not one of the predefined ones, no lexical units of this synset are added to the synonyms list.

At this point it is worth mentioning another challenge that the implementation of this part had to deal with: the compounds. The compounds are already split into their elements from the previous stage of the implementation, during the lemmatization. Therefore, compounds enter the GermaNet analysis in their ‘split’ form, e.g. *Parkbremse* ‘parking brake’ is of the form *parken+Bremse* ‘park+brake’. This means that it is necessary to make sure that each element is looked up separately in GermaNet but that the synonyms of both are inserted together in the synonyms list of the word.

Pseudocode 4 aims at giving an overview of this part of the development.

---

**Pseudocode 4:** Getting the GermaNet synonyms out of a token (part of)

**Input:** GermaNet data, token (lemma of the token)

**Output:** synonymsList

```
IF token is not a compound THEN
  synsets = get the synsets of the token
  FOR EACH synset:
    IF synset belongs to the predefined classes THEN
      synonyms = get the lexical units of this synset
    IF synonymsList does not contain synonyms yet THEN
      add the synonyms to the synonymsList
ELSE IF token is compound THEN
  numberOfCompounds = count the subwords within the token
  FOR EACH subword:
    synsets = get the synsets of the subword
    FOR EACH synset:
      IF synset belongs to the predefined classes THEN
        synonyms = get the lexical units of this synset
      ENDIF
    IF synonymsList does not contain the synonyms yet THEN
      add the synonyms to the synonymsList
    ENDIF
ENDIF
```

---

After the GermaNet analysis has taken place, the synonyms list of each word of the search query is added to the metadata of the search query. Now, the expanded query contains not only the words of the initial query, but also their lemmata and their synonyms. The search query is thus complete and can be forwarded to the passage retrieval component (Figure 18).

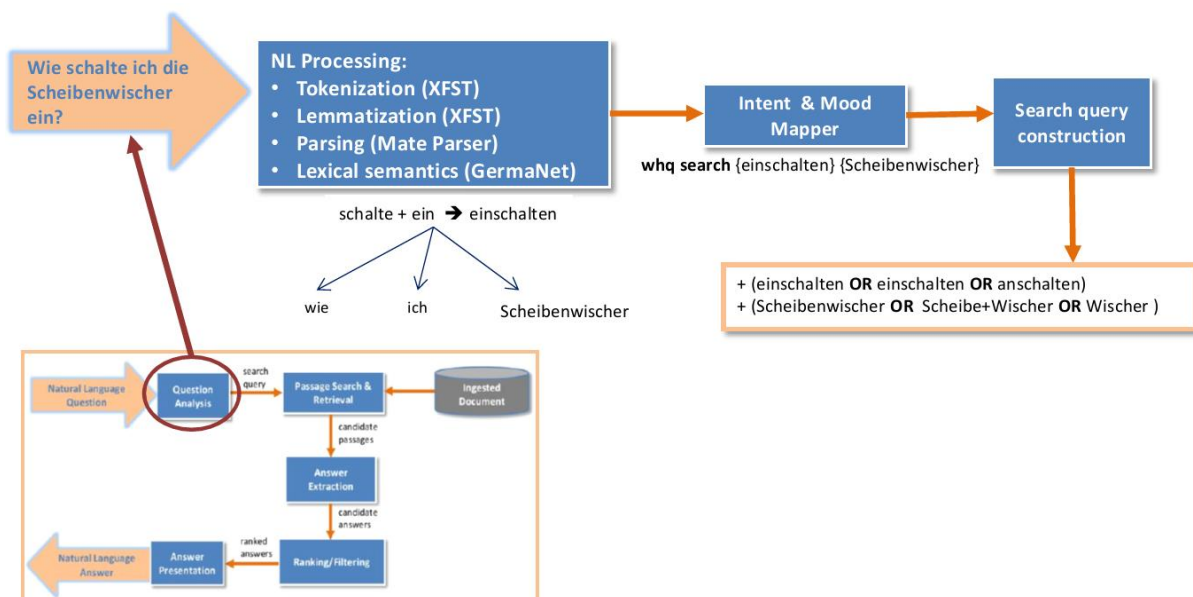


Figure 18: Structure of the question analysis component within the German QA system.

### 5.2.3. Other components of the pipeline

With the search query being complete, the implementation of the strongly language dependent parts is completed. In order to have a better overview of the differences in the implementation of the language specific parts of the two systems, table 2 shows what are the different resources used for each language.

Table 2: Differences between the implementations of the language dependent components of the English ‘thesis version’ and the German QA systems.

Level	English	German
Tokenization	Clear Parser	FST
Morphology	Clear Parser	FST
Word senses/ Lexical semantics	Nuance internal concept annotator	GermaNet
Parsing	Clear Parser	Mate Parser

Again, at this point it should be noted that none of the systems is using a higher level semantics approach (cf. Figure 16), even if there is such an approach implemented for English.

After those two first stages, the components following in the pipeline can execute their parts as in the English system. For those subsequent parts of the system (passage retrieval, answer extraction and answer ranking/filtering), as is was mentioned before, the algorithms involved

were not somehow adapted to the German QA system because one of the goals of this project is to see whether the algorithms used for the English system are flexible enough to be used for other languages as well. The performance that those algorithms ensure will be evaluated during the testing process.

However, it should be noted that certain smaller modifications were necessary in the subsequent components so that they become compatible with the new German implementation. Also the implementation included some refactoring of the current QA system in order to account for an architecture which allows the simultaneous implementation for the two languages. Neither the modifications nor the refactoring of the overall architecture are going to be presented since they refer to the exact architecture of the company's software and exceed the purpose of this thesis.

## 6. Testing and evaluating the German QA system

### 6.1. Testing the system

After implementing the language dependent parts for the German system and making the necessary adaptations in the other parts of the original system, the development of the German QA system was completed. At this point it was important to see whether the system is working altogether by being able to answer very simple questions and if so, test it with different kinds of questions to see how good it is, i.e. what its performance is. In the first case, a working system means that the first question addressed within this thesis, whether a partially language independent QA pipeline can be successfully ported to other languages than English, can be confirmed. On the other hand, a good performance with different kinds of questions means that the algorithms used for the English system are also language-flexible and can return satisfactory results for other languages. Thus, the testing was carried out in two steps as it will be presented in the following subsections.

#### 6.1.1. Testing the functionality of the system

Firstly, in order to check the overall functionality of the system the two simple questions in (11) and (12) were tried out.

(11) Wie starte ich den Motor?  
how start I the engine  
*How do I start the engine?*

(12) Wie kann ich die Sitzheizung einschalten?  
how can I the seat-heating turn-on  
*How can I turn on the seat heating?*

Those questions were considered “simple” because their answers are straightforward; the relevant passages can be found with the exact words within the Audi A3 user manual. For (10) the relevant passage is *Mit dem Schlüssel im Zündschloss wird die Zündung eingeschaltet und der Motor gestartet*. ‘With the key in the ignition, the ignition is switched on and the engine starts’ and for (11) the relevant passage is *Um die Sitzheizung einzuschalten drücken Sie die Taste <Bild> einmal*. ‘In order to turn on the seat heating, press once the button <Symbol>’. This means that for those questions the action of the lemmatizer is not essential anyway since the words of the question are those found in the document as well (e.g. no written out words vs. compounds, no different POS-tags, etc.) and even the conversion of the parser output to the dependency graph has some “room for errors”

since there are no complex dependencies involved. Additionally, the quality of the output of the lexical semantics does not play such an important role either because the words do not need any kind of synonyms to be matched to the documents. On the other hand, both sentences are relatively simple as far as their linguistic features are concerned. Question (10) is a how (*wie*)-question with a transitive verb being used as main verb at the second position. *Motor* and *starte* are the only words that need to be taken into account for the dependency graph and therefore for the search. Question (11) is similar; it's again a how (*wie*)-question but with the syntax of a modal verb; the modal is at the second position and is being inflected while its complement is a transitive verb at the last position and in infinitive. *Sitzheizung* and *einschalten* are the only words that need to be searched for. The questions do not include embedded clauses, collocations, metaphors or anaphora and co-reference phenomena. Although the questions are simple, some characteristics make them ideal to test whether the separate parts of the system work in the expected way. For example, the compound *Sitzheizung* can show whether the lemmatization works correctly – even if in this case it is not necessary to work correctly as explained above. Moreover, the two questions cover two of the most common syntax structures in German: questions with a “full” verb (*Vollverb*) and questions with a modal verb. This helps to find out whether the parser can produce the right output and if this output can indeed be converted into a correct graph for both cases. The expected synonyms of the main verbs *starten* ‘start’ and *einschalten* ‘turn on’ –even if not necessary– are quite straightforward, e.g. *anschalten* and *anmachen* ‘turn on/switch on’.

Both questions were able to retrieve relevant passages and within those the correct answers; in both of them the answer with the highest score was indeed the one presented before as the correct one. This result confirmed that the system is functional, at least with those two simple questions. The next investigation to be made is what the performance of the system is.

### 6.1.2. Testing the overall performance

The performance of the German QA system was tested in comparison with the performance of the ‘thesis version’ English QA system in order to test the hypotheses presented in 1.2. (absolute values are not going to be given for confidentiality reasons). Although it was attempted to implement the German system in a way that makes it comparable to the English one (e.g. by applying it on an Audi user manual, by using a parser of a similar technology and similar lexical resources), it should be noted that the two systems are not equal. The main differences are the differences in the annotation of the initial documents (concept vs. lemma

annotation), the different parsers used as well as the lack of a word sense disambiguation technique for the German search query. This means that the systems are compared to each other, but they are not equal since the time frame of this thesis did not allow for a full implementation of an equal German system; the English system is expected to have a better performance and the German system is expected to show a tendency to reach the performance of the English QA.

The testing of the two systems was based on two test sets of 200 questions each. In order to create these test sets, both user manuals, the English and the German, were laid side by side and only questions whose answers could be found in both manuals were created and included in the test sets. In other words, the German test suite with 200 questions (see Appendix B) contains the same questions as the English test suite (see Appendix A); it should be noted that the questions were not translated from one language to another but they were originally written in each language keeping their meaning the same. In this way, word-to-word-translation and transfer errors were avoided and the results that were produced are comparable.

Before going on with the exact results, it is worth describing how the test suites were built and what kinds of questions are found within them. The analysis focuses on the German test suite; however, the comments to be made for the German questions and their characteristics were taken into account for the creation of the English questions to ensure that the cases to be observed are similar. First of all, it should be made clear that the test suite contains questions referring to all contents under the link *Audi A3 Betriebsanleitung* ‘operational manual’ (see <http://a3bb.s4net.de/inhalt.htm>). For the creation of the questions there are phenomena of three levels of the language taken into account: morphology, syntax and semantics. In this way, it is ensured that there is much variation among the questions on many different levels and that no particular structures are favored against others. Additionally, those three disciplines of linguistics are important fields that are worth observing in such a natural language understanding task. With such a variation on different levels it can also be tested whether the system is really “clever” and does not simply match the keywords to words of the document; in other words, it can be shown whether the natural language processing involved (e.g. tokenization, lemmatization, syntactic and (partial) semantic parsing) and thus the components implemented are really efficient.

As far as the morphology is concerned, when possible, it was attempted to state the question in a way that the keywords are of a different part-of-speech than the one being used in the original manual or generally different morphological variants. An example of such a case is shown in (13). Although the document is using an NP *Ausfall der Zentralverriegelung* ‘failure of the central locking system’ to express the condition in which we should act in a special way, the question created contains an embedded clause *wenn die Zentralverriegelung ausfällt* ‘when the central locking system fails to work’ with the intransitive verb and its subject conveying the main concept of the condition.

- (13) Document excerpt: „Bei einem Ausfall der Zentralverriegelung...  
 at a failure of-the central-locking-system..  
*In case of failure of the central locking system...*

Question: „Was sollte ich tun wenn die Zentralverriegelung ausfällt?“  
 What should I do when the central-locking-system fails-to- work?  
*What should I do if the central locking system fails to work?*

Furthermore, it is important that the system can account for the common German phenomenon of compounds. Therefore, in the cases where the original document contained a compound, the question was stated using the subwords of the compound written out and vice versa; when the document contained a written out expression that could be turned into a compound, then the question was created to contain the compound. Example (14) shows that the document is using the compound *Funkschlüsselbatterie* ‘remote-control-key-battery’ but the question was created to contain the written out expression *Batterie des Funkschlüssels* ‘battery of the remote-control-key’.

- (14) Document excerpt: „Funkschlüsselbatterie ersetzen: ...“  
 remote-control-key-battery replace: ...  
*Replacing the battery of the remote control key:.*

Question: “Wie kann ich die Batterie des Funkschlüssels ersetzen?”  
 How can I the battery of-the remote-control-key replace  
*How can I replace the battery of the remote control key?*

Moving up a level, to syntax, there are three verb categories used in the role of the main verb. There are questions containing a modal verb such as *können* ‘can’, *dürfen* ‘may’, *sollen* ‘should’ or *müssen* ‘must’, which are being inflected and take another verb as their complement. There are also questions using transitive or intransitive verbs (*Vollverben*), also being inflected and belonging either to the separable ones (*trennbare Verbe*) or to the unseparable ones (*untrennbare Verbe*). The third verb category includes questions with copula



verbs such as *sein* ‘be’. By expressing the questions in all those different ways, it can be tested whether the main notion of the question can still be extracted no matter the way of expressing it. Apart from that, the questions include active and passive voice structures which again ensure the flexibility of the parsing (and of the system). Particularly, it was attempted to formulate questions using passing voice when the relevant excerpt of the document contained active voice and vice versa. Example (15) is a representative example where the active voice of the document *erneuern* ‘replace’ is found as a passive voice structure *gewechselt werden* ‘to be changed’ in the question (for now let us ignore the fact that there is a synonym used; this will be a case for the semantics presented later).

- (15) Document excerpt: „Aus Sicherheitsgründen sollten Sie die Scheibenwischerblätter  
Of security-reasons should you the windscreen-wiper-blades  
jährlich ein-bis zweimal erneuern.“  
every-year one-to-two-times replace.  
*For security reasons you should replace the wiper blades once  
or twice a year.*

Question: „Wie oft müssen die Scheibenwischerblätter gewechselt werden?“  
how often must the windscreen-wiper-blades changed AUX  
*How often should the wiper blades be replaced?*

Moreover, the questions of the test set include embedded clauses such as conditional ones or temporal. This means that the parsing needed is more complex since the content of the main clause has to be recognized as the central one and the content of the embedded clause has to support it. Example (16) is a question with a conditional clause:

- (16) “Darf ich rauchen wenn ich tanke?“  
May I smoke when I fill-the-tank  
*May I smoke when filling the tank?*

Another syntactic phenomenon – it could also be partially semantic – that is covered within the question set is collocations. Due to their special nature, it is important that collocations are recognized as such by the parser so that their parts are not treated separately but as a whole. Thus, questions like (17) are found within the test set in order to test whether this is indeed the case.

- (17) “Wie sollte ich die Start Stop Taste benutzen?“  
how should I the start stop button use  
*How should I use the start stop button?*

The last level of variation is the semantic one. In order to test the performance in this aspect, there were different kinds of strategies included in the questions. First of all, the questions

are both interrogative questions and *yes/no* questions. Interrogative questions i.e. the English *wh*-questions, are the ones that begin with a question word such as *wann* ‘when’, *wo* ‘where’, *warum* ‘why’, *was* ‘what’ etc. and expect therefore a variety of longer answers. *Yes/no* questions are the ones that can be answered with a single *yes* or *no*. Therefore, the kind of question defines the kind and context of answer we should get back in each case, as already explained in 2.5.1. Additionally, it is necessary that a good QA system not only answers questions containing the exact words found in the documents but also other words semantically related to those. The test questions were therefore created in many cases with synonyms, hypernyms, hyponyms etc. of the original words. Example (18) illustrates a very simple case of synonymy where the question uses the verb *anschalten* ‘turn on’ instead of the verb *einschalten* ‘switch on’ being used within the document:

- (18) Document excerpt: „Unter folgenden Bedingungen die Sitzheizung nicht  
 under following circumstances the seat-heating not  
 einschalten...”  
 switch-on:....  
*You should not turn on the seat heating under the following  
 circumstances:...*

Question: “Wann darf ich die Sitzheizung nicht anschalten?”  
 when may I the seat-heating not turn-on  
*When may I not turn on the seat heating?*

Another structure that falls within the semantic context and needs to be recognized correctly are foreign words. Since the user manual contains English terms, the test questions should also include those and test in this way if the foreign terms are recognized as such and it is not attempted to parse them word by word. An example of such a question is found in (19):

- (19) Was ist das Audi drive select?  
 what is the Audi drive select  
*What is the Audi drive select?*

Last but not least, the test set includes some more complex questions where semantic inference or the deep semantic representation of the question is required in order to be answered. This means either that the answer cannot be directly found within the document but needs to be inferred from other contents of the document or that the question can only be answered if the semantic representations of the document and the question can be matched. An example of such a case is given in (20) where *der Effect* ‘the impact’ should be matching a certain logic representation in order to match with the overall meaning of the relevant sentence found in the document.

(20) Document excerpt: „Im Umluftbetrieb wird die Luft im Fahrzeuginnenraum umgewälzt und gefiltert. Dadurch wird weitgehend verhindert, dass verunreinigte Außenluft in den Fahrzeuginnenraum gelangt.“

*Through the automatic air recirculation the air in the vehicle circulates and is being filtered. Through this it is prevented that dirty outside air comes into the inner of the vehicle.*

Question: “Was ist der Effekt des Umluftbetriebs?”

what is the effect of the air-recirculation

*What is the effect of the automatic air recirculation?*

All in all, it was attempted to include as much variation and as many phenomena possible in the question sets; however, it is not expected that all of those questions are going to be handled in the right way, e.g. the inference or deeper semantic analysis phenomena due to the lack of a deeper semantic analysis tool. That is exactly what we need to test.

## 6.2. Evaluating the system

For the actual evaluation of those test sets, each test set went through the QA pipeline, which returned a list of best scoring answers for each question. Those results were then run through the Nuance evaluation platform, which sets up a rating framework for every question along with all its answers. Then, every answer of every question was rated by being assigned one of the three following criteria:

- GOOD: the answer is a perfect answer to the given question.
- RELEVANT: the answer is relevant to the question; it could be a potential answer or part of the perfect answer.
- IRRELEVANT: the answer has nothing to do with the given question.

Every answer provided also the passage from where the answer was extracted, thus the justification of why the answer is relevant.

The relevance criterion (and the accompanying justification) is the one also used in the TREC evaluations as discussed in 2.6. The “goodness” criterion is an extra one which poses an additional challenge for our system; a given answer should not only be considered “relevant” to the question as the TREC evaluations require it but specifically answer it – be the “perfect” answer. With such a restriction, all criteria presented in section 2.6., such as correctness, conciseness, completeness etc. are taken into account, making the evaluation metrics produced even more representative and even more demanding of course. Example (21) illustrates a question and what would be a RELEVANT answer to it and what a GOOD

answer (answers are taken from the Audi A4 user manual used: *www.audihelp.com*). The RELEVANT one almost answers to the question but it is not a one-to-one answer like the GOOD answer is.

(21) Question: “What is the best position for the head restraints?”

Relevant Answer: “The front seats and head restraints must always be positioned correctly for the height of the occupant ...”

Good Answer: “For best protection, the top of the head restraint should be at least at eye level, or higher ...”

Based on the ratings of all answers of all questions, the following metrics can be produced for the whole set of the questions (see section 2.6 for more details on those metrics):

- Precision@k: how many results are GOOD in the top k returned. This precision metric does not exactly correspond to the precision metric commonly used – and analyzed in 2.6. – since for the latter the RELEVANT and not the GOOD answers are taken into account. There were three metrics produced within the precision: Precision@1, Precision@3 and Precision@10, corresponding to whether a perfect answer could be found among the first, three and ten, respectively, top results obtained.
- Relevance@k: how many results are RELEVANT (or GOOD since GOOD is always RELEVANT) in the top k returned. This metric is the one that actually corresponds to the precision metric commonly used since the RELEVANT answers are taken into account. Again, three metrics were produced: Relevance@1, Relevance@3 and Relevance@10, corresponding to whether a relevant answer could be found among the first, three and ten, respectively, top results obtained.
- nDCG: the ideal ranking (if all best results were ranked in descending order) compared to the actual ranking.

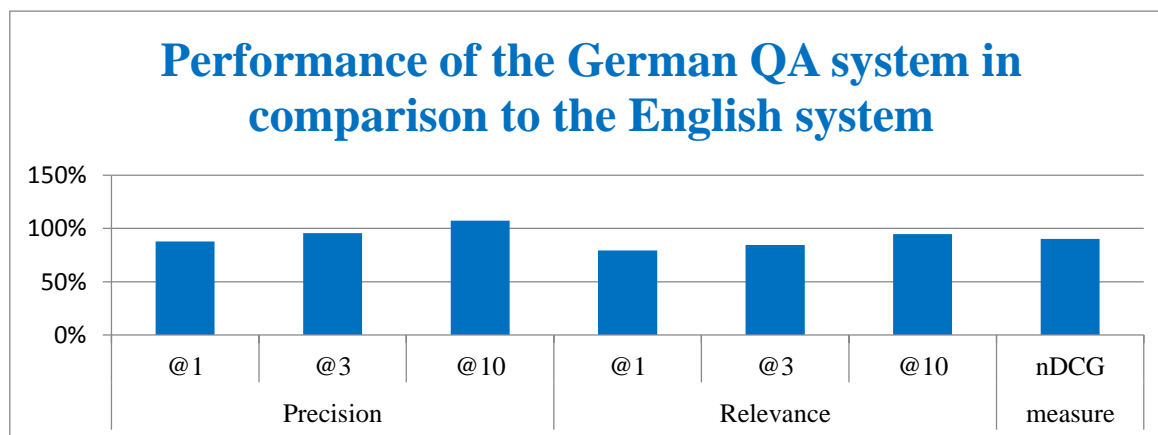
Table 3 shows the percentages that each metric of the German system achieved relative to the ‘thesis version’ English system. Absolute numbers are omitted for confidentiality reasons.

**Table 3: Relative performance of the German QA system in comparison to the ‘thesis version’ English QA system.**

	Precision			Relevance			nDCG
	@1	@3	@10	@1	@3	@10	
Relative performance of German vs. English QA system	88%	96%	107%	79%	84%	95%	90%

For the better visualization of the results Table 4 illustrates them in a graphic way:

Table 4: Visualization of the performance results.



The results show that the German system is able to correctly answer a high percentage of the questions correctly answered by the English system (all following comments concern the comparison of the two systems and do not refer to absolute values). Firstly, we should look at the relevance metrics since relevance is the measure most commonly used (*cf.* TREC evaluations, Hirschman & Gaizauskas, 2001). The German system can find a relevant answer within the top 10 results in 95% of the cases, when compared to English. For the relevance metrics @1 and @3, as it was expected, the percentages of relevant answers are a bit lower since the top results under investigation are less. Still, in both cases the German system achieves an around 80% performance in comparison to English. Moving on to precision, which as explained above, is an additional measure that captures whether an answer is the perfect answer to the question and not just relevant, there are some interesting comments to be made. Precision @1 and @3 prove a relative performance of around 90%. Interestingly, for precision @10 the German system outperforms the English one, since it can find a correct answer within the top 10 results in more cases than the English system does. As far as the nDCG is concerned, it can be observed that the quality of the current ranking of the answers within the German system in comparison to their ideal ranking corresponds to 90% of the quality of the ranking of the English system.

### 6.3. Discussing the performance of the German system

The overall performance of the German QA system is very satisfactory. The three most characteristic metrics, the Precision@3, the Relevance@10 and the nDCG prove that the English system is able to answer only 10% more questions than the German system. At this point, it should once again be stressed that the two systems being compared are far from

equal. Despite this fact, the results that the German system is returning would not necessarily indicate that the systems are not equal. Firstly, it could be argued that the slight differences in the results are due to the differences in the relative qualities of the sub components of the two systems. For example, the “younger” GermaNet contains 101.371 synsets (GermaNet. A German Wordnet, 2009), while the Nuance internal concept annotator for English contains even more than the 155.287 synsets contained in WordNet (WordNet: A lexical Database for English, 2015). Furthermore, the fact that German as an agglutinative language is a more complex task compared to the moderately inflected English could also explain those slight differences. That agglutinative languages, like German and Japanese, pose more problems and have a different behavior in such natural language processing tasks was also noted by Gomez-Soriano et al. (2005) in their effort to develop a language independent PR system (2005:817). Leaving such reasons aside, the performance of the German system is a clear indication that the system will resemble the performance of the English one, if the current differences of the two systems would be reconciled for, meaning if the NLP tools used would become equal for the two languages (*cf.* Table 2).

Starting from the bottom of the NLP hierarchy, the tokenization, which at the moment is done by the Clear Parser for English and with the FST tools for German, could be unified to a common technique using the FST tools for both languages. The same could be the case for the morphology being extracted from the documents and the questions; FST could be applied on both languages. As far as the word senses is concerned, the English system uses the Nuance internal concept annotator which not only maps words to concepts but is also developed further to do word senses disambiguation, while the German system relies on the GermaNet resources exclusively for synonym extraction without going deeper into some kind of disambiguation. It should also be mentioned that the German system makes use of a morphological technique, lemmatization, for the document annotation instead of the concept annotation used in the English system. Although lemmatization has certain advantages and allows for limited variation as presented in 5.2.1., it is not comparable to the variation that the concept annotation can achieve. Moving on to the parsing part, the fact that there are two different parsers used plays a big role in the difference of the performance even if the parsers are of a similar nature and use similar algorithms. The Mate Parser which is also able to process English input could take up the parsing for both languages and therefore account for cases which at the moment are dealt differently by each parser. Such steps could make the two systems even more comparable and reduce the differences in their performance.

In fact, another side-experiment that could be completed showed that working in this direction is the right way to get the expected results and close the gap in the performance. For this experiment the ‘thesis version’ of the English system was made “dumber”. From the above mentioned linguistic aspects it was decided to “switch off” the English word sense annotation (and disambiguation), since this feature is completely absent from the German system. In this way it could be tested how the results change. With this feature “switched off”, the system has no lexical/conceptual semantics and is thus even “dumber” than the German system, which involves the search query expansion with synonyms. The relative performance of the German system in this case (summarized in table 5) shows interesting results.

**Table 5: Relative performance of the German QA system in comparison to the "dumb" English QA system.**

	Precision			Relevance			nDCG
	@1	@3	@10	@1	@3	@10	
Relative performance of German vs. “dumb” English QA system	138%	134%	122%	87%	89%	96%	93%

The precision results show that now the German system has a better performance than the “dumb” English one, something that was expected due to the presence of lexical semantics in the German system. Such a finding implies that making the German system as “clever” as the normal English system could actually close the gap in the performance. Nevertheless, the rest of the results raise interesting questions firstly as far as the relation of precision and relevance is concerned. While the precision of the German system is higher than of the “dumb” English system, its relevance is lower, something counter-intuitive at a first glance. Questions also arise concerning the aspects that could account for the relevance and the nDCG being higher in the “dumber” English system. Such interesting questions remain to be investigated in the future.

Going back to the overall performance and to the main experiment of the thesis, it can be argued that the effort to port the partial language independent Nuance QA pipeline to German was successful in three aspects. Firstly, the German system is functional, which means that the adaptation of the language specific components of the system is enough to make the Nuance system operative for another language. With that, the first question addressed within this thesis may be confirmed. Furthermore, the German QA system does not simply “work”,

it also has a relative high performance when compared to the English system. Such a performance confirms that the algorithms used within the Nuance QA system are also efficient for other languages than English, especially if we take into account that the performance would probably be even better if the systems were exactly equal and therefore the algorithms used more representative for the result. Also the fact that the metrics used are relative strict in comparison to the common evaluation metrics used in TREC (Hirschman & Gaizauskas, 2001) is another aspect supporting the quality of the Nuance system. With all this the second question addressed within this thesis that the algorithms used are appropriate and “flexible” for other languages is also answered. Moreover, the fact that the German results are not equal to the English ones is a good start, giving the system the opportunity to learn and improve, e.g. by improving its semantic processing. Last but not least, it should be noted that both systems could improve their absolute performance if the higher level semantics tool would be integrated – as already mentioned, for English this tool already exists. If it is assumed that the current systems and their performance are “starting points”, it can also be argued that the integration of deeper semantics would boost their performance even more.



## 7. Summary and Future work

The goal of the current work was to present the technology behind the modern state-of-the-art QA systems, describe the most popular ones and contribute to the development of the field by porting the ‘thesis version’ English Nuance QA system to German.

The implemented German system showed that the English system can successfully be ported to other languages as the adaptation of the language dependent parts is enough to make the overall performance of the system similar to the performance of the ‘thesis version’ English system. Thus, the questions stated within this thesis can be confirmed and the overall aim to build a state-of-the-art German QA system has also been met.

Despite the effort to keep the differences in the implementation of the two systems minimal, so that they are easily and correctly comparable, differences do exist and they can account for the different results in the overall performance of the two systems. Future work could try to eliminate those differences even more, e.g. by using the same parser for both languages, applying the same kind of annotations on the documents, adding a word sense disambiguation technique for German, etc. As shown with the side-experiment, making the two systems equally “clever” could close the gap in their performance. With such new experiments, it is expected that new questions and challenges will arise. Moreover, a further goal should be to try out different approaches for the various stages of the process, e.g. use word embedding models instead of the GermaNet resources for the lexical semantics part. Additional work could also include the adaptation of the system to a third language, preferably a non-Germanic one, e.g. a Romance language, since such a language could show other interesting performance results.

## List of Abbreviations

ALGOL: Algorithmic Language

CSS: Cascading Style Sheets

DG: Dependency Grammar

FAQ: Frequently Asked Questions

FSM: Finite State Morphology

FST: Finite State Tool

HTML: HyperText Markup Language

IPL: Information Processing Language

IPL-V: Information Processing Language, version 5, 1958

IR: Information Retrieval

LISP: Locator/Identifier Separation Protocol

NL: Natural Language

NLP: Natural Language Processing

POS-tagging: Part-Of-Speech tagging

PR: Passage Retrieval

QA: Question-Answering

QLL: Query Logic Language

RDF: Resource Description Framework

TREC: Text Retrieval Conference

XFST: Xerox Finite State Tool

## List of Figures

Figure 1: Generic Architecture of a QA system as presented in Hirschman & Gaizauskas (2001) – slightly modified. ....	11
Figure 2: Example of a <i>START</i> answer to the question "When did Beethoven die?" .....	20
Figure 3: Overview of the <i>Ephyra</i> system as presented in Schlaefter, et al. (2006).....	21
Figure 4: Overview of the <i>AskMSR</i> system as presented in Brill, et al. (2002).....	23
Figure 5: Overview of the <i>AnswerBus</i> system as presented in Zheng (2002). ....	25
Figure 6: Overview of the <i>Watson</i> system as presented in Lapshin (2012).....	27
Figure 7: Overview of the ontology-based system as presented by Athira, Sreeja and Reghuraj (2013). ....	29
Figure 8: Architecture combining the current question and the questions in the archive as presented by Mansoori & Hassanpour (2012). ....	31
Figure 9: Asking Nuance Nina for an answer to the question 'Where can I find the vehicle identification number?'. ....	33
Figure 10: <i>Nina</i> giving back the best answer found. ....	34
Figure 11: Overview of the ‘thesis version’ Nuance QA pipeline.....	35
Figure 12: Structure of the document preprocessing component of the Nuance pipeline (‘thesis version’). ....	36
Figure 13: Structure of the question analysis component of the Nuance pipeline (‘thesis version’). ....	39
Figure 14: Structure of the passage retrieval component of the Nuance pipeline (‘thesis version’). ....	40
Figure 15: Structure of the Answer Ranking/Filtering component of the Nuance pipeline (‘thesis version’). ....	41
Figure 16: Pyramid illustrating the natural language analysis stages involved in the language dependent components of the pipeline (special thanks to Dick Crouch for the design of the pyramid).....	43

Figure 17: Structure of the document preprocessing component within the German QA system. ....49

Figure 18: Structure of the question analysis component within the German QA system. ....54

## List of Tables

Table 1: Comparison between the three main QA approaches (Dwivedi & Singh, 2013:422). .....	10
Table 2: Differences between the implementations of the language dependent components of the English ‘thesis version’ and the German QA systems. ....	54
Table 3: Relative performance of the German QA system in comparison to the ‘thesis version’ English QA system. ....	63
Table 4: Visualization of the performance results. ....	64
Table 5: Relative performance of the German QA system in comparison to the "dumb" English QA system. ....	66

## References

- [1] Athira, P. M., M. Sreeja, & P.C. Reghuraj. 2013. Architecture of an Ontology-Based Domain-Specific Natural Language Question Answering System. *International Journal of Web & Semantic Technology (IJWesT)*, 31-39.
- [2] Audi A3. Retrieved 01.11.2015 from <http://a3bb.s4net.de/inhalt.htm>.
- [3] Audi A4. Retrieved 01.11.2015 from <http://www.audihelp.com/>.
- [4] Beesley, K., & L. Karttunen. 2003. *Finite State Morphology*. US: CSLI Publication.
- [5] Björkelund, A., B. Bohnet, L. Hafdell, & P. Nugues. 2010. A high-performance syntactic and semantic dependency parser. *Coling: Demonstration Volume*, 33-36.
- [6] Bohnet, B., & J. Kuhn. 2012. The Best of Both Worlds -- A Graph-based Completion Model for Transition-based Parsers. *European Chapter of the Association for Computational Linguistics (EACL) 13*, 77-87.
- [7] Bohnet, B., & J. Nivre. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. *Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 1455-1465.
- [8] Brill, E., S. Dumais, & M. Banko. 2002. An Analysis of the AskMSR Question-Answering System. *Empirical Methods in Natural Language Processing (EMNLP)*, 257-264.
- [9] Choi, J., & N. Nicolov. 2009. K-best, locally pruned, transition-based dependency parsing using robust risk minimization. *Recent Advances in Natural Language Processing V*, 205-216.
- [10] Choi, J., & M. Palmer. 2011a. Getting the Most out of Transition-based Dependency Parsing. *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies 49*, 687-692.
- [11] Clear Parser Documentation. 2012. Retrieved 15.01.2016 from <https://code.google.com/archive/p/clearparser/>
- [12] Crouch, D. 2005a. Packed rewriting for mapping semantics to KR. *International Workshop on Computational Semantics*, 6.
- [13] Crouch, D. & T. H. King. 2006. Semantics via f-structure rewriting. *Lexical Functional Grammar*, 6.
- [14] Dwivedi, S. K., & V. Singh. 2013. Research and reviews in question answering system. *Procedia Technology 10*, 417-424.
- [15] Eliza. Retrieved 20.12.2015 from <http://www.masswerk.at/elizabot>
- [16] GermaNet. A German Wordnet. 2009. Retrieved 05.01.2016 from <http://www.sfs.uni-tuebingen.de/GermaNet/>
- [17] Gómez-Soriano, J. M., M. Montes-y-Gómez, E. Sanchis-Arnal, L. Villaseñor-Pineda, & P. Rosso. 2005. Language Independent Passage Retrieval for Question Answering. *Mexican International Conference on Artificial Intelligence (MICAI)*, 816-823.
- [18] Green, F. B., A. K. Wolf, C. Chomsky, & K. Laughery. 1961. BASEBALL: An Automatic Question Answerer. *Institute of Radio Engineers. American Institute of Electrical Engineers. Association for Computing Machinery (IRE-AIEE-ACM)*, 219-224.
- [19] Hamp, B., & H. Feldweg. 1997. GermaNet - a Lexical-Semantic Net for German. *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.
- [20] Henrich, V., & E. Hinrichs. 2010. GernEdiT - The GermaNet Editing Tool. *International Language Resources and Evaluation (LREC) 7*, 2228-2235.

- [21] Hirschman, L., & R. Gaizauskas. 2001. Natural language question answering: the view from here. *Natural Language Engineering* 7, 275-300.
- [22] Katz, B. 1997. Annotating the World Wide Web using Natural Language. *Computer Assisted Information Searching on the Internet (RIAO)* 5.
- [23] Lapshin, A. V. 2012. Question-Answering Systems: Development and Prospects. *Automatic Documentation and Mathematical Linguistics*, 138-145.
- [24] Llopis F, J. L. Vicedo, & A. Ferrández. 2002. Using a Passage Retrieval System to support Question Answering Process. *International Conference on Computational Science (ICCS)*, 61-69.
- [25] Mansoori, M., & H. Hassanpour. 2012. Boosting Passage Retrieval through Reuse in Question Answering. *International Journal of Engineering*, 187-196.
- [26] *OpenEphyra*. Retrieved 20.12.2015 from <https://sourceforge.net/projects/openephyra/>
- [27] *openNLP*. 2016. Retrieved 10.02.2015 from <http://opennlp.apache.org/>
- [28] Prager, J. 2001. One search engine or two for question answering. *Text Retrieval Conference (TREC)* 9.
- [29] Roberts, I., & R. Gaizauskas. 2004. Evaluating Passage Retrieval Approaches for Question Answering. *European Conference on IR Research*. 26, 72-84.
- [30] Schlaefter, N., P. Giesemann, T. Schaaf, & A. Waibel. 2006. A Pattern Learning Approach to Question Answering Within the Ephyra Framework. *Text, Speech and Dialogue Conference*, 687-694.
- [31] Seeker, W., & J. Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. *International Conference on Language Resources and Evaluation*. 8, 3132-3139.
- [32] Simmons, R. 1965. Answering English questions by computer: A survey. *Communications of the ACM* 8, 53-70.
- [33] Simmons, R. F. 1970. Natural Language Question-Answering Systems: 1969. *Communications of the ACM*, 15-30.
- [34] START. Natural Language Question Answering System. Retrieved 15.12.2015 from <http://start.csail.mit.edu/index.php>
- [35] Sindhu, L., & U. Sasikumar. 2014. A Survey of Natural Language Question Answering. *International Journal of Computer Applications*, 42-46.
- [36] Van Valin, R. D. 2001. *An introduction to Syntax*. Cambridge: Cambridge University press.
- [37] Weizenbaum, J. 1966. ELIZA – A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM* 9, 36-45.
- [38] Wikipedia: Information Retrieval. 2016. Retrieved 10.02.2016 from [https://en.wikipedia.org/wiki/Information\\_retrieval](https://en.wikipedia.org/wiki/Information_retrieval)
- [39] Wikipedia: Lemma (morphology). 2016. Retrieved 02.02.2016 from [https://en.wikipedia.org/wiki/Lemma\\_%28morphology%29](https://en.wikipedia.org/wiki/Lemma_%28morphology%29)
- [40] Wikipedia: Lucene. 2016. Retrieved 05.01.2016 from <https://en.wikipedia.org/wiki/Lucene>
- [41] Wikipedia: Search engine indexing. 2016. Retrieved 12.01.2016 from [https://en.wikipedia.org/wiki/Search\\_engine\\_indexing](https://en.wikipedia.org/wiki/Search_engine_indexing)
- [42] Woods, W. 1973. Progress in natural language understanding: An application to LUNAR geology. *National Computer Conference*, 441-450.
- [43] Woods, W. A. 1978 Semantics and Quantification in Natural Language Question Answering. *Advances in Computers*.

- [44] WordNet: A lexical Database for English. 2015. Retrieved 20.01.2016 from <https://wordnet.princeton.edu/>
- [45] Yining, W., W. Liwei, L. Yuanzhi, H. Di, C. Wei, & L. Tie-Yan. 2013. A Theoretical Analysis of NDCG Ranking Measures. *Conference on Learning Theory (COLT)* 26.
- [46] Zheng, Z. 2002. AnswerBus question answering system. *Human Language Technology Conference*.
- [47] 12 Star Trek Gadgets That Now Exist. Retrieved 15.02.2016 from <http://mentalfloss.com/article/31876/12-star-trek-gadgets-now-exist>



## Appendices

### Appendix A: English Test set

1. What are the warning and indicator lamps for?
2. Is it good to run the tank dry?
3. What is the auto lock function?
4. When does the locking system prevent me from being locked out?
5. What should I do if the central locking system fails to work?
6. What should I do if there is a failure of the central locking system?
7. How can I replace the battery of the remote control key?
8. How can I manually open the boot lid?
9. What are the child-proof catches?
10. For how long can the windows be operated after the ignition has been switched off?
11. How can I open the sun roof?
12. What is the warning for a malfunction of the light sensor?
13. When should I use the fog lights?
14. What is the adaptive light function?
15. What is the tilting function for the exterior mirror on the passengers' side?
16. How do I change the windscreen wiper blades?
17. How often should the windscreen wipers be changed?
18. Can I switch on the windscreen wipers even when the wiper blades are frozen?
19. Does the interior light go off if the door is left open?
20. From where can I adjust the driver's seat?
21. Does the electrical seat adjustment work when the ignition is off?
22. What is the best position for the head restraints?
23. What is the purpose of the load through hatch?
24. What is the maximum weight that the retaining net can hold?
25. May I use cloth hangers to hang up my clothing on the coat hooks?
26. Can the luggage compartment be extended?
27. What should I pay attention to when I use the roof carrier?
28. How heavy may the roof load be?
29. How is the cigarette lighter used?
30. How many volts is the socket of the car?
31. Does the cigarette lighter work when the ignition is off?
32. What are the storage compartments of the vehicle?
33. What is the use of the pollution filter?
34. When does the rear window heating switch off?
35. How can I use the air conditioner environmentally friendly?
36. What is the supplementary heater?
37. How do I turn on the air conditioner?
38. What is the automatic mode of the air conditioner?
39. How can I defrost the windows?
40. What is the effect of the automatic air recirculation?
41. What could be the reason for the rear window heater not working?
42. Is there a seat heating?
43. When shouldn't I turn on the seat heating?
44. Where is the lever to adjust the position of the steering wheel?
45. Why is the engine noisy when starting it?

46. May I remove the key form the ignition while driving?
47. How do I start the engine?
48. How should I use the start stop button?
49. When is the steering lock engaged?
50. How long does the fan work for after the ignition has been turned off?
51. When does the speed limit control warning appear?
52. What is the electro mechanical parking brake good for?
53. How does the Audi hold assist support me?
54. What should I do if the main brake system fails?
55. Can I set a speed limit?
56. Can I set my distance to other vehicles?
57. What is the adaptive cruise control system?
58. When shouldn't I use the adaptive cruise control system?
59. How does the distance control work?
60. Can I use the acceleration pedal when the adaptive cruise control function is on?
61. How do I set a new speed for the cruise control system?
62. Do I have to change gears when the cruise control is on?
63. How can I set a driving program?
64. What could impair the function of the radar sensors?
65. What is the Audi braking guard?
66. How does the lane assist work?
67. How do I get the display of the lane assist?
68. What does the side assist do?
69. How can I switch the side assist on?
70. What are the limitations of the side assist?
71. How does the side assist work?
72. Can I adjust the brightness of the warning lamp on the exterior mirror?
73. What could impair the function of the side assist?
74. Is the side assist function available when towing?
75. What is the Audi drive select?
76. On what does the vehicle set-up in each mode depend?
77. What are the different driving modes?
78. What are the characteristics of each driving mode?
79. How do I select a driving mode?
80. Can I configure my own personal driving mode?
81. What are the characteristics of the comfort driving mode?
82. What is multi-tronic?
83. When is the hill hold assist function activated?
84. What are the selector lever positions?
85. What is the kick down feature?
86. How can I shift between gears in the tip-tronic system?
87. How can I release the selector lever manually?
88. What should I do if I accidentally switch to N while driving?
89. How does the Audi parking system work?
90. What is the Audi parking system plus?
91. How can I switch on the display of the Audi parking system?
92. Can the Audi parking system replace the driver?
93. Where is the reversing camera located?
94. Which parking mode should I use when parking in a garage?

95. What kind of obstacles cannot be detected by the parking system?
96. What is the long beep I am hearing?
97. Are the rear parking sensors activated when the vehicle is in towing mode?
98. What should I do if I want to park next to obstacles?
99. When does the parking aid switch on automatically?
100. What is the correct seating position for the driver?
101. What should the distance between my breast and the steering wheel be?
102. How should the front seat passenger adjust his seat?
103. What are some examples of incorrect sitting positions?
104. Which floor mats are the most suitable?
105. Are there any tips when putting things in the luggage compartment?
106. Why is it good to fasten the seat belt?
107. What could happen if I don't fasten my seat belt?
108. Is it ok to wear my seat belt loose?
109. When should the seat belts be replaced?
110. May two passengers share the same seat belt?
111. Is it good for the unborn child if I wear a seat belt?
112. How should I wear the seat belt if I am pregnant?
113. How can I adjust the height of the seat belt?
114. How do the belt tensioners work?
115. What is the use of the airbag system?
116. What are the main parts of the airbag system?
117. Can airbags substitute the seat belts?
118. How would I notice that there is a fault in the airbag system?
119. Where is the front airbag for the driver?
120. When are the airbags triggered?
121. Is there a warning lamp indicating the functionality of the airbag system?
122. Where are the side airbags located?
123. How do the head protection airbags work?
124. What is the best position for the child seats?
125. Do I need to deactivate the airbags when I put the child seat on the front seat?
126. Where should children under 12 years old sit?
127. What kind of child restraint systems should I best use?
128. What does ESP do?
129. What is the ABS system?
130. What does the brake assist system do?
131. How do I switch off the ESP?
132. What does the wear of the brake pads depend on?
133. When do new brake pads give full braking effect?
134. What does the brake servo do?
135. How does the power steering help?
136. Where is the engine power?
137. Should I use winter tires for the winter?
138. What does the power management system do?
139. What does the power management system do if the vehicle stays parked for many days?
140. How much distance should I cover in order for the vehicle to run in?
141. What kind of petrol should I use?
142. What are some driving tips to minimize pollution and fuel consumption?

143. When should I change to the next gear?
144. How does the trailer stabilization system work?
145. Should I turn the ignition off when washing the vehicle?
146. How often should I apply a wax coat?
147. When should I polish the vehicle?
148. Where can I find the number of the original paint finish of the car?
149. With what product should I best clean the display screen?
150. How do I remove stains from the interior of the vehicle?
151. What kind of leather is there in the interior of the car?
152. How can I clean the seat belts?
153. Is it allowed to have business equipment installed later on?
154. May I smoke when filling the tank?
155. From where do I release the bonnet?
156. Should I open the bonnet if I see steam coming out of it?
157. What should I note when working in the engine compartment of the vehicle?
158. Where is the radiator expansion tank located?
159. When must I refill the engine oil?
160. How do I check the level of the engine oil?
161. How much of the anti-freeze additive do I need?
162. How can I check the coolant level?
163. What is the right amount of brake fluid?
164. Can I open the battery?
165. Where is the battery?
166. How do I recharge the battery?
167. Where are the terminals for charging the battery?
168. May I charge a frozen battery?
169. Is plain water suitable for the windscreen washer system?
170. After how many kilometers do new tires give maximum grip?
171. Where do I find information about the tire inflation pressures?
172. What are the tread wear indicators?
173. How fast can I go if there are snow chains on the tires?
174. What is the benefit of run flat tires?
175. When shouldn't I continue driving with the help of the run flat tires?
176. What should I note when driving with run flat tires?
177. May the run flat tires be repaired with the TMS?
178. Where is the warning triangle located?
179. Where can I find the fire extinguisher?
180. Where can I find the tire mobility system?
181. How do I take out the spare wheel?
182. How should I use the spare wheel?
183. Could I use snow chains on the spare wheel?
184. What is the speed limit when driving with the spare wheel?
185. Can the tire mobility system be used at all temperatures?
186. What should I pay attention to after repairing a tire?
187. How should I inflate the tire?
188. What do I need to turn the anti-theft wheel bolts?
189. How should I raise the vehicle when changing the wheel?
190. How do I connect the jump leads?
191. What should the maximum speed be when being towed by another vehicle?

192. How can I tow start?
193. What should I pay attention to when using a towrope?
194. Can I change the xenon headlights myself?
195. How can I change the bulb for the front fog lights?
196. Where is the type plate located?
197. Where can I find the vehicle identification number?
198. What info does the vehicle data sticker contain?
199. How many liters fit in the windscreen washer?
200. What is the capacity of the windscreen washer?

## Appendix B: German Test set

1. Wozu gibt es die Kontrollleuchten?
2. Ist es gut den Tank leer zu fahren?
3. Was ist die Funktion Auto Lock?
4. Wann wird es verhindert mich auszusperren?
5. Was sollte ich tun wenn die Zentralverriegelung ausfällt?
6. Was sollte ich beim Ausfall der Zentralverriegelung tun?
7. Wie kann ich die Batterie des Funkschlüssels ersetzen?
8. Wie kann die Gepäckraumklappe per Hand notentriegelt werden?
9. Was ist die Kindersicherung?
10. Für wie lange kann ich noch die Fenster schließen wenn die Zündung ausgeschaltet ist?
11. Wie kann ich das Panorama-Glasdach öffnen?
12. Was ist die Meldung für eine Störung des Lichtsensors?
13. Wann darf ich die Nebelschlussleuchte benutzen?
14. Was ist das Audi adaptive light?
15. Was ist die Kippfunktion des Beifahreraußenspiegels?
16. Wie wechsele ich die Scheibenwischerblätter?
17. Wie oft müssen die Scheibenwischerblätter gewechselt werden?
18. Darf ich die Scheibenwischer anschalten wenn die Scheibenwischerblätter gefroren sind?
19. Wird das innere Licht automatisch ausgeschaltet wenn die Tür offen geblieben ist?
20. Wie kann ich den Fahrersitz manuell einstellen?
21. Funktioniert die automatische Einstellung der Vordersitze wenn die Zündung ausgeschaltet ist?
22. Was ist die beste Position für die Kopfstützen?
23. Wozu kann die Durchladeeinrichtung verwendet werden?
24. Was sollte die maximale Beladung der Gepäcktasche sein?
25. Darf ich Kleiderbügel zum Aufhängen der Kleidung auf den Kleiderhaken benutzen?
26. Kann der Gepäckraum vergrößert werden?
27. Was sollte ich beachten wenn ich den Dachgepäckträger benutze?
28. Wie schwer darf die Dachlast sein?
29. Wie ist der Zigarettenanzünder zu benutzen?
30. Wie viele Volt ist die Steckdose des Fahrzeugs?
31. Funktioniert der Zigarettenanzünder auch wenn die Zündung ausgeschaltet ist?
32. Was sind die verschiedenen Fächer des Fahrzeugs?
33. Wozu wird der Schadstofffilter verwendet?
34. Wann wird die Heckscheibenbeheizung ausgeschaltet?
35. Wie kann ich die Klimaanlage umweltbewusst benutzen?
36. Was ist der Zuheizung?
37. Wie schalte ich die Klimaanlage an?
38. Was ist der Automatikbetrieb der Klimaanlage?
39. Wie kann ich die Scheiben entfrosten?
40. Was ist der Effekt des Umluftbetriebs?
41. Warum könnte die Heckscheibenbeheizung nicht funktionieren?
42. Gibt es eine Sitzheizung?
43. Wann darf ich die Sitzheizung nicht anschalten?

44. Wo befindet sich der Hebel zum Einstellen des Lenkrads?
45. Warum gibt es Laufgeräuschen beim Starten des Motors?
46. Darf ich den Schlüssel aus dem Zündschluss ziehen während ich fahre?
47. Wie starte ich den Motor?
48. Wie sollte ich die start-stop Taste benutzen?
49. Wann wird die Lenkradsperre eingerastet?
50. Für wie lange läuft der Kühlerventilator noch wenn ich den Motor abgestellt habe?
51. Wann gibt die Geschwindigkeitswarnanlage eine Meldung?
52. Wozu dient die elektromechanische Parkbremse?
53. Was ist der Anfahrsistent?
54. Was sollte ich tun wenn die Fußbremse ausgefallen ist?
55. Kann ich ein Geschwindigkeitslimit einstellen?
56. Kann ich meinen Abstand zu anderen Fahrzeugen festlegen?
57. Was ist das adaptive cruise control System?
58. Wann sollte ich das adaptive cruise control System nicht benutzen?
59. Wie funktioniert die Abstand Funktion des cruise control Systems?
60. Kann ich den Gaspedal treten wenn der cruise control an ist?
61. Wie stelle ich eine neue Geschwindigkeit in dem cruise control System ein?
62. Muss ich schalten auch wenn der cruise control eingeschaltet ist?
63. Wie kann ich ein Fahrprogramm einstellen?
64. Was könnte die Funktion des Radarsensors beeinträchtigen?
65. Was ist das Audi pre sense front System?
66. Wie funktioniert der Spurhaltassistent?
67. Wie rufe ich den active lane assist auf?
68. Was macht der Spurwechselassistent?
69. Wie kann ich den Spurwechselassistent einschalten?
70. In welchen Fällen gibt es Funktionseinschränkungen im Spurwechselassistent?
71. Wie warnt mich der Spurwechselassistent?
72. Kann ich die Anzeigehelligkeit am Außenspiegel ändern?
73. Was könnte die Funktion der side assist Sensoren beeinträchtigen?
74. Ist die side assist Funktion im Anhängerbetrieb verfügbar?
75. Was ist das Audi drive select?
76. Wovon ist die Fahrzeugabstimmung in jedem Modus abhängig?
77. Was sind die verschiedenen Fahrmodus?
78. Was sind die Eigenschaften jedes Fahrmodus?
79. Wie kann ich einen bestimmten Fahrmodus einstellen?
80. Kann ich selber den Fahrmodus festlegen?
81. Was sind die Charakteristika des Fahrmodus comfort?
82. Was ist s tronic?
83. Wann wird die Bergabunterstützung aktiviert?
84. Was sind die Wählhebelstellungen?
85. Was ist die kick down Funktion?
86. Wie kann ich umschalten im tiptronic System?
87. Kann ich den Wählhebel notentriegeln?
88. Was sollte ich tun falls ich während der Fahrt versehentlich auf N schalte?
89. Wie funktioniert die Einparkhilfe?
90. Was ist die Einparkhilfe plus?
91. Wie stelle ich die Anzeige der Einparkhilfe ein?
92. Kann die Einparkhilfe den Fahrer ersetzen?



93. Wo befindet sich die Rückfahrkamera?
94. Welche Ansicht sollte ich benutzen wenn ich in eine Garage parken will?
95. Welche Objekte können vom Einparkhilfesystem nicht erkannt werden?
96. Was ist der Dauerton den ich höre?
97. Sind die hinteren Sensoren der Einparkhilfe aktiviert wenn das Fahrzeug im Anhängermodus ist?
98. Was sollte ich tun wenn ich neben ein Hindernis parken will?
99. Wann wird die Einparkhilfe automatisch eingeschaltet?
100. Was ist die richtige Einstellung für den Fahrersitz?
101. Welcher sollte der Abstand zwischen Lenkrad und Brustbein sein?
102. Wie sollte der Beifahrer seinen Sitz einstellen?
103. Was sind einige Beispiele von falscher Sitzposition?
104. Welche Fußmatten sind die geeignetsten?
105. Gibt es einige Tipps für das Verstauen von Gepäckstücken im Gepäckraum?
106. Warum ist es empfehlenswert die Sicherheitsgurte zu benutzen?
107. Was könnte passieren wenn ich nicht angegurtet bin?
108. Ist es ok wenn ich meinen Sicherheitsgurt lose angelegt habe?
109. Wann sollten die Sicherheitsgurte ersetzt werden?
110. Dürfen zwei Passagiere die gleichen Sicherheitsgurte teilen?
111. Ist es gut für das ungeborene Kind wenn ich den Sicherheitsgurt anlege?
112. Wie sollte ich die Sicherheitsgurte anlegen wenn ich schwanger bin?
113. Wie kann ich die Gurthöhe einstellen?
114. Wie funktionieren die Gurtstraffer?
115. Was ist der Zweck des Airbag Systems?
116. Woraus besteht das Airbag System?
117. Kann das Airbag System den Sicherheitsgurt ersetzen?
118. Wie würde ich merken dass es eine Störung des Airbag Systems gibt?
119. Wo finde ich den vorderen Airbag für den Fahrer?
120. Wann werden die Airbags ausgelöst?
121. Gibt es eine Kontrollleuchte welche die Funktionsbereitschaft des Airbags Systems anzeigt?
122. Wo sind die Seiten Airbags zu finden?
123. Wie funktionieren die Kopf Airbags?
124. Was ist die beste Position für den Kindersitz?
125. Muss ich die Airbags deaktivieren wenn ich den Kindersitz auf dem Beifahrersitz verwende?
126. Wo sollten Kinder unter 12 Jahre sitzen?
127. Was für Kindersitze darf ich benutzen?
128. Was tut ESC?
129. Was ist das ABS System?
130. Was tut der Bremsassistent?
131. Wie kann ich das ESC System ausschalten?
132. Wovon ist die Abnutzung der Bremsbeläge abhängig?
133. Wann erhalten neue Bremsbeläge ihre volle Bremswirkung?
134. Was tut der Bremskraftverstärker?
135. Wie unterstützt mich die elektromechanische Lenkung?
136. Wo ist die Antriebskraft verteilt?
137. Sollte ich Winterreifen für den Winter benutzen?
138. Was tut das Energiemanagement System?



139. Was macht das Energiemanagement System wenn ich das Fahrzeug für mehrere Tage nicht fahre?
140. Wie viel muss ich fahren um das Fahrzeug einzufahren?
141. Was für Benzin soll ich tanken?
142. Was sind einige Tipps um den Kraftstoffverbrauch und die Umweltbelastung zu reduzieren?
143. Wann sollte ich schalten?
144. Wie funktioniert die Gespannstabilisierung?
145. Sollte ich die Zündung ausschalten wenn ich das Auto waschen will?
146. Wie oft muss ich das Fahrzeug mit Hartwachs schützen?
147. Wann sollte ich das Fahrzeug polieren?
148. Wo finde ich die Lacknummer des Originallacks des Autos?
149. Womit sollte ich am besten das Display sauber machen?
150. Wie kann ich Flecken im Innen des Fahrzeugs entfernen?
151. Was für eine Lederart gibt es im Innen des Autos?
152. Wie kann ich die Sicherheitsgurte reinigen?
153. Darf ich Geschäftsausrüstung nachträglich einbauen lassen?
154. Darf ich rauchen wenn ich tanke?
155. Wie kann ich die Motorraumklappe entriegeln?
156. Darf ich die Motorraumklappe aufmachen wenn Dampf daraus austritt?
157. Was sollte ich beachten wenn ich im Motorraum arbeite?
158. Wo befindet sich der Kühlmittelasgleichsbehälter?
159. Wann sollte ich das Motoröl nachfüllen?
160. Wie stelle ich den Ölstand fest?
161. Wie viel Kühlmittelzusatz brauche ich?
162. Wie kann ich den Kühlmittelstand prüfen?
163. Wo muss der Bremsflüssigkeitsstand liegen?
164. Darf ich die Batterie öffnen?
165. Wo befindet sich die Batterie?
166. Wie kann ich Batterie aufladen?
167. Wo finde ich die Anschlüsse zum Laden der Batterie?
168. Darf ich eine gefrorene Batterie laden?
169. Kann ich einfaches Wasser zum Nachfüllen des Scheibenwaschbehälters benutzen?
170. Nach wie vielen Kilometern habe neue Reifen die optimale Haftfähigkeit?
171. Wo finde ich Information über die Reifendruckwerte?
172. Was sind die Verschleißanzeiger?
173. Wie schnell darf ich fahren wenn Schneeketten am Auto gefestigt sind?
174. Was ist der Vorteil von Reifen mit Notlaufeigenschaften?
175. Wann darf ich auch mit den Reifen mit Notlaufeigenschaften nicht weiterfahren?
176. Was sollte ich beim Fahren mit Reifen mit Notlaufeigenschaften beachten?
177. Dürfen die Reifen mit Notlaufeigenschaften mit dem Reifenreparaturset repariert werden?
178. Wo ist das Warndreieck zu finden?
179. Wo befindet sich der Feuerlöscher?
180. Wie gelange ich zum Bordwerkzeug?
181. Wie kann ich das Notrad rausnehmen?
182. Wie sollte ich das Notrad benutzen?
183. Könnte ich Schneeketten auf dem Notrad benutzen?
184. Was ist das Geschwindigkeitslimit wenn ich mit dem Notrad fahre?

185. Darf das Reifenreparaturset in allen Temperaturen benutzt werden?
186. Was sollte ich berücksichtigen nachdem ich eine Reife repariert habe?
187. Wie sollte die Reife aufgepumpt werden?
188. Was brauche ich um die Diebstahlhemmenden rausschrauben zu lösen?
189. Wie soll ich das Fahrzeug anheben wenn ich die Reife wechseln will?
190. Wie sollte ich die Starthilfekabel anklemmen?
191. Was ist die maximale erlaubte Geschwindigkeit wenn mein Fahrzeug abgeschleppt wird?
192. Wie kann ich anschleppen?
193. Worauf sollte ich achten wenn ich ein Abschleppseil verwende?
194. Kann ich die Xenon Leuchtmittel selber wechseln?
195. Wie kann ich die Nebelscheinwerferlampe wechseln?
196. Wo ist das Typschild?
197. Wo finde ich die Fahrgestellnummer?
198. Welche Information steht auf dem Fahrzeugdatenträger?
199. Wie viele Liter passen in der Scheibenwaschanlage?
200. Was ist die Kapazität der Waschanlage der Scheiben?