Experiments on Robust NL Question Interpretation and Multi-layered Document Annotation for a Cross–Language Question/Answering System

Günter Neumann and Bogdan Sacaleanu
LT–Lab, DFKI, Saarbrücken, Germany
{*neumann,bogdan*}*@dfki.de*

Abstract

This report describes the work done by the QA group of the Language Technology Lab at DFKI, for the 2004 edition of the Cross-Language Evaluation Forum (CLEF). Based on the experience we obtained through our participation at QA@Clef-2003 with our initial cross-lingual QA prototype system BIQUE (cf. [NS03]), the focus of the system extension for this year's task was a) on robust NL question interpretation using advanced linguistic-based components, b) flexible interface strategies to IR-search engines, and c) on strategies for off-line annotation of the data collection, which support query-specific indexing and answer selection.

The overall architecture of the extended system, as well as the results obtained in the CLEF–2004 Monolingual German and Bilingual German/English QA tracks will be presented and discussed throughout the paper.

1 Introduction

The basic functionality of an open–domain cross–language question/answering (QA) system is simple: given a Natural Language query in one language (say German) find answers for that query in textual documents written in another language (say English). In contrast to a standard cross-language IR system, the NL queries are usually well-formed NL–query clauses (instead of a set of keywords), and the identified answers should be the *exact* answer string (instead of complete documents containing the answer). Thus, for a question like "Welches Pseudonym nahm Norma Jean Baker an?" (*Which pseudonym did Norma Jean Baker use?*) the answer should be "Marilyn Monroe" rather than an English document containing this name. In contrast to QA@Clef-2003, this year the task was made further difficult by demanding that only *one* exact answer should be returned instead of a ranked list of (say three) answer candidates.

Last year our group participated for the very first time in a QA competition. Since the focus was more on system implementation than on system tuning, the main motto was "participation is everything". However, we learned a lot and found several sources of potential improvements for our initial system. Especially two aspects have drawn our attention.

Firstly, the use of a statistical based chunk-parser turned out to be a major bottleneck for the complete NL question processor. In our Clef–2003 system, we implemented a two-stage question process: first we performed a shallow chunk analysis using a statistical based chunker (trained for German as well as English) on which output we applied a manually written specialized question grammar. The rules of this grammar represented direct relationships between relevant chunks and their interpretation wrt. question and expected answer type. However it turned out that the error rate of the first stage actually caused to much noisy input for the second stage, so that in many

cases we were not able to determine the expected answer type correctly. This was further effected by the very low coverage of the manually specified question grammars, so that the whole question processor actually performed quite poor. However, it is known that a high number of errors in question answering can be attributed to errors in question analysis (cf. [MPHS02]). Furthermore, since the Clef-2004 QA task required that only one exact answer should be returned, we were convinced that it would be at least a good strategy to prefer a more deeper linguistic–based question analysis strategy.

Secondly, in the Clef-2003 system we applied a very simple strategy for determining relevant paragraphs which are then used as starting points for determining possible answer candidates, simply by directly using the SGML paragraph tags from the original corpus. Furthermore, the IR-query language of the MG system actually turned out to be too inflexible so that we could not take advantage of a preprocessing of the corpus wrt. different dimensions. Hence, we could only perform a very basic word/stem–level oriented paragraph indexing.

Based on these experiments, we decided to extent the Clef–2003 system to the following directions:

- development of a robust NL question interpretation using sophisticated deeper linguistic-based strategies,
- development of flexible "programmable" interface strategies to IR-search engines, and
- development of strategies for off-line annotation of the data collection, which support query-specific indexing and answer selection.

We now start with an overview of the whole Clef–2004 system, and highlight some technical aspects. Finally, we present and discuss the results we have obtained for the task.

2 System overview

Figure 1 displays the architecture of our Clef–2004 QA–system. Basically, the same system is used for the monolingual as well as the bilingual QA task with only very few additional task–specific parameterizations. The core architecture consists of five major components:

1. the linguistic core engine
2. the multi-dimensional index of the answer source corpus
3. the robust NL query processor
4. the information search component
5. the answer processor

The linguistic core engine consists of two major sub-components: a) LingPipe, which performs NE and sentence boundary recognition, as well as NE co-reference resolution, and b) a sentence-based syntax parser. This parser is currently only used for German NL question and document analysis.

For each corpus of the individual task (German and English), a multi-dimensional index structure is computed off–line. This is done by first preprocessing the whole corpus with the LingPipe component of the linguistic core engine, which basically adds named entities, sentence boundary, NE–co-references and abbreviations to each document in form of XML–tags. For each specific dimension a separate index structure is computed which can be accessed via the IR server (we are using the Jakarta Lucene full–text search engine, see also sec. 4.1).[1]

The major control flow for both QA tasks can now briefly be described as follows:

[1] cf. http://jakarta.apache.org/lucene/docs/index.html

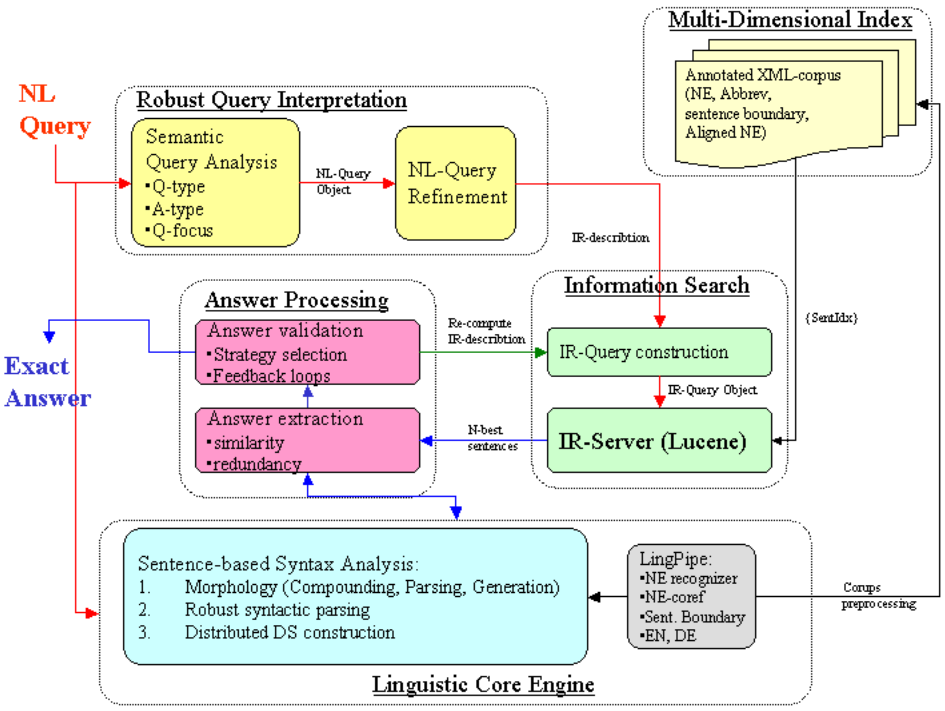

Figure 1: The architecture of DFKI's Clef–2004 QA-system.

Robust NL Question Analysis The main purpose of the NL question analysis in the context of a QA-system is to determine the expected answer type, the set of relevant keywords, and the set of recognized NE–instances in order to guide information search and answer processing. Consider, for example, the NL–query result presented in figure 4, where the value of tag A-TYPE represents the expected answer type, S-CTR's value represents the answer control strategy, and the value of SCOPE represents additional constraints for the search space (for more details, see sec. 4.2).

NL Question Refinement Refinement of the result of the NL–query covers the translation of the NL–query and its expansion. The cross-language aspect of the system has been approached by using machine translation engines for query translation (along the line of the approach described in [NS03]). We have selected a number of 8 translation services (7 online + 1 offline) in order to account for a better lexical coverage for the translated queries. The results of translation have been linguistically processed, annotated with named entities and merged into a translation object consisting of named entity instances and keywords (open class words which were not parts of named entities). The question analysis has yielded a similar structure for the original question plus additional information about expected answer type and scope. By using a dictionary-based alignment technique this additional information has been transferred to the translation object. The same happened with the named entities of types PERSON, DATE (year instances) and NUMBER, which should remain unchanged through translation. As for the remaining types of named entities (LOCATION, ORGANIZATION and DATE without year instances), which might have different lexical representations in source and target language, the following heuristic applied: if the original string and its translation were different (e.g., "Europäische Gemeinschaft" vs. "European Union") they were regarded as unreliable, added as keywords and discarded from named entities. The distinction made between named entities and keywords along the question analysis process will be used later on in constructing the IR-query and defining search strategies.

In contrast to our previous system, we wanted to implement and test question expansion

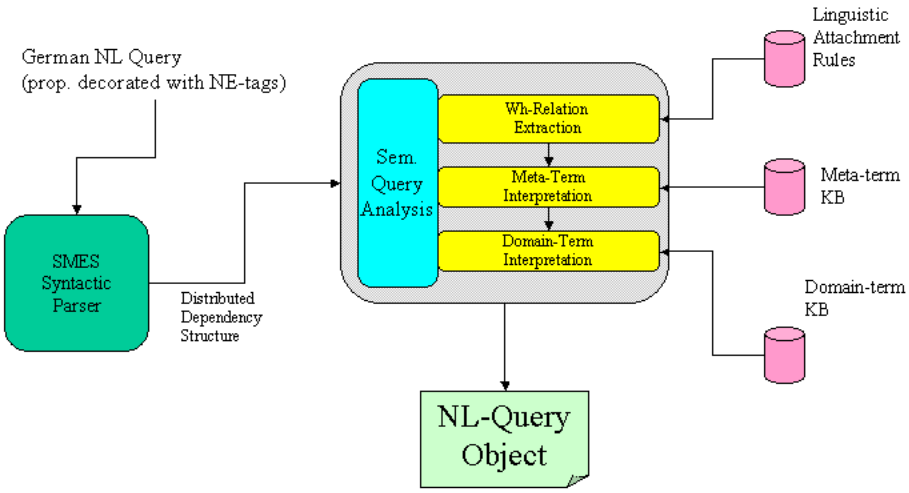

Figure 2: The architecture of the query analysis component.

methods based on *natural language generation* (NLG), instead of using WordNet (in a latter system, we will combine these methods, of course). The main reason for doing this, is the fact, that the NL–query analysis actually normalizes all words to their corresponding lemmas. On the other side, the morphological component of our German parsing system SMES (cf. section 3.2) is *reversible*, i.e., can also be used for the generation of word forms. Of course, one could directly use the word forms of the input query (accessible via indices).[2] However, generating all plausible word forms directly from the input query actually would perform a controlled morpho-syntactic query expansion. Thus, in the case of the monolingual German task, for all relevant lemmas of the NL–query analysis (these are basically belonging to the open-class words), we generate all word forms which are *consistent* with the feature description of the syntax analysis of the parsing result. In other words, this means that the parsing output directly controls the generation input. For example for the lemma *geben* (`to give`) we are generating the word forms *gaben, gab, gegeben, gibt.*[3]

Information Search In order to perform the information search, the (possibly refined) NL–query has to be mapped to a concrete IR-query. Most today's information search engines come with expressible IR-query interfaces, which support a flexible user-driven filtering of the index space. In order to take advantage of this rich parameterization and to support the use of multiple IR-engines in the future, we actually perform the mapping from a NL-query to a IR-query in two steps (cf. also figure 3):

1. construction of a IR-query schema

[2]We have to submit word forms instead of lemmas to Lucene, because it applies its own stemmer on each IR–query term.

[3]Note that our method also allows to specify additional constraints for the generation process, e.g., that only word forms of a certain tempus should (not) be generated. In this way, a more sensitive morpho-syntactic-based control of query expansion is possible.

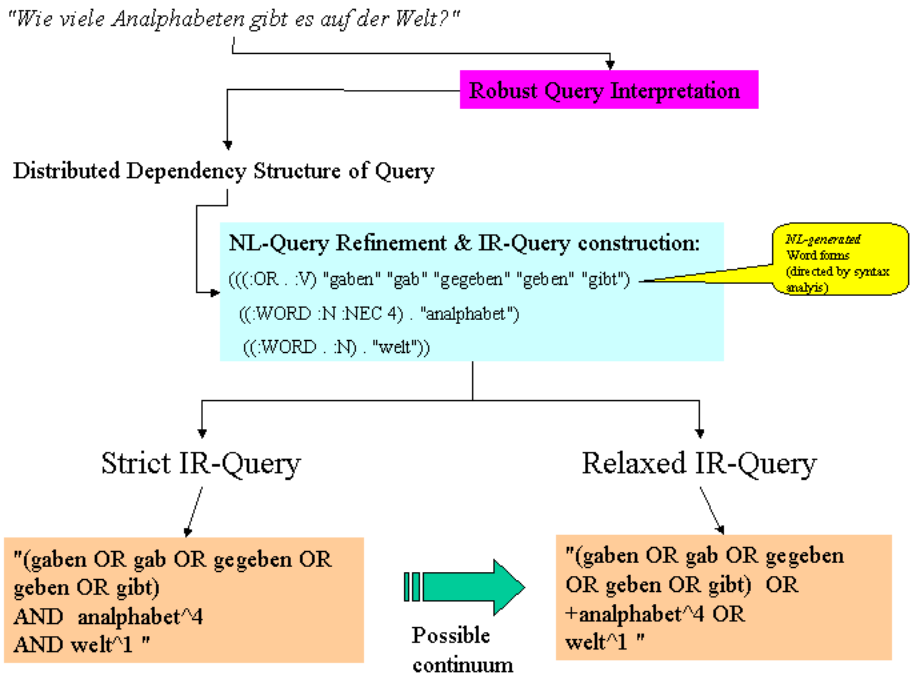

Figure 3: Example of query refinement and IR-Query construction.

2. construction of a IR-query

An IR-query schema is actually an under-specified representation of an IR-query. It is constructed directly from the NL–query result. Although it contains all relevant information from the NL–query, this information is under-specified, because it still lacks the use of IR-specific syntax (e.g., the '+'–prefix for necessary terms) and a specification of logical connectives. The main task of the IR–query construction component is to create a concrete IR–query from this schematic description, which directly can be feed into a IR–engine (in our current QA system, we are using the Jakarta Lucene full text engine. However, it would also be possible to create, say a Google-specific expression from it using a different script). Which mapping to perform is expressed in form of search strategies which are activated on basis of concrete values of the question tags of NL–query. Furthermore, based on input from the answer validation component (through feedback loops), the component might use different sorts of logical connectives resulting in different IR–queries, e.g., in figure 3 two alternatives are displayed: a strict IR–query (using only logical AND), and a relaxed IR–query (using only OR).

Performing the IR–query construction process in the way just described allows us to selectively make use of different indexing structures.

Answer Processing The result of the information search is a set of N indexes (currently N=10), where each index is a pointer to a single sentence of an annotated document. Each sentence is tagged with all NE–instances recognized by LingPipe during the document preprocessing phase. Note that, because we do indexing on a sentence level (and not on a paragraph level as done in our Clef-2003 system), we actually can take advantage of cross–document sentence–level redundancies.

Next, all NE–instances which are type compatible with the expected answer type of the question are selected as possible answer candidate. All identified exact answer candidates are stored in a global list together with its frequency counts (determined on basis of the selected N–sentences). During that step a similarity function is applied on the NE–instances in order to identify variants

of the same name. Note that this means that the quality of the answer extraction step currently depends directly on the quality of the used NE–recognition system.

By default, we do the information search with a strict IR–query. If in this case, no sentence can be retrieved or no answer can be extracted, we perform a new information search using a relaxed IR–query and re-call the answer processing component.

3 Linguistic Core Engine

Our linguistic core engines consists of two major components which we describe briefly in the next two subsections.

3.1 LingPipe

LingPipe, which is a software package from Alias–i, consists of several language processing modules: a statistical named entity recognizer, a heuristic sentence splitter, and a heuristic within-document co-reference resolution system.[4]

LingPipe comes with a English language model. The types of NE covered by LingPipe are locations, persons and organizations. We have retrained LingPipe, both for English and German, so as to cover two more named entities types: DATE and NUMBER (just for German available) and extended the co-reference resolution algorithm to count for German pronouns as well. A large Gazetteer of named entity instances has been used for both languages and for English a PERSON Gazetteer with gender attributes has been integrated for a better co-reference resolution.

3.2 SMES

SMES is a robust wide-coverage unification-based system for the parsing of German texts (cf. [NBB+97, NP02]).[5] It produces a partial analysis of natural language texts by combining shallow processing techniques (i.e., finite state regular expression recognizers) with generic linguistic resources (e.g., subcategorisation, morphology, online compound analysis). In contrast to the common approach of deep grammatical processing, where the goal is to find all possible readings of a syntactic expression, we provide a complete but underspecified representation, by only computing a general coarse-grained syntactic structure which can be thought of as domain independent. This rough syntactic analysis can be then made more precise by taking into account domain-specific knowledge. Our parser recognizes basic syntactic units and grammatical relations (e.g., subject/object) robustly by using relatively underspecified feature structures, by postponing attachment decisions and by introducing a small number of heuristics.

Originally, SMES was developed as an Information Extraction core system, however we now have SMES extended substantially for its use as a core-engine in textual question answering systems. Major extensions of SMES concern the development of the robust interpretation of NL questions (see sec. 4) and the development of a distributed representation for the dependency structures, which we will describe now in more detail.

Distributed representation In the original SMES system, the analysis of a sentence is represented in form of a possibly recursive dependency tree where each node and edge is decorated with rich feature information. During the development of our Clef–2004 system it turned out that the nested parse trees (which can be very huge for very long sentences) are unsuited as a generic interface, because they do not support a flexible and efficient access to relevant linguistic information. Furthermore, a nested representation cannot easily be enriched with additional linguistic structure, e.g., additional grammatical functions or deeper attachment, scopus etc. The same is true for a selective, local integration of domain–specific information (e.g., to perform a

[4] LingPipe is available at http://www.alias-i.com/lingpipe/

[5] SMES is available at http://www.dfki.de/~neumann/pd-smes/pd-smes.html. SMES has been used in a number of third–party projects, and has extensively been evaluated.

sort of concept–spotting on basis of domain-independent syntactic normalization of relevant text fragments).

For that reason, we re–represent dependency trees in form of a distributed representation, adapting the approach of [Mil00]. A distributed representation provides the robustness of a bag–of–object approach with the ability to use higher level relational information where this can provide a more accurate analysis. Thus distributed representations are more flexible wrt. the integration of shallow and deep linguistic analysis, and the integrating domain knowledge. In our distributed representation, we explicitly separate the representation of linguistic entities like words/chunks/named entities (the "bag–of–objects" or *BaseObjects*) from their structural relationship like head/modifier/topology/grammatical functions (the "bag–of–links" or *LinkObjects*). Both layers are connected through indices which allow a simple bidirectional traversal between the different object types. Linguistic and application specific extension can then be described as operations (typing, re-organization of attachment) applied on LinkObjects. Actually, this is how the strategies of the semantic NL–query interpretation are implemented for determining the expected answer type and question scope (cf. sec. 4.2). It is also basis for the specification of a flexible similarity function applied on two distributed dependency trees.

4 Some more details

4.1 Multi-layered Document Annotation

The current annotation analysis performed on the document collection consisted in sentence boundary identification, named entities annotation and co-reference resolution of named entities and personal pronouns. A heuristic syntax–based algorithm for identifying abbreviations and their possible extensions was run over the collection as well, resulting in an annotation similar to that of named entities.

Throughout the document processing part of the system we have insisted on a systematic analysis of named entities and a reduction of the amount of information necessary to answer a question. Based on experiments and results with the question set of the previous competition, we have confined the information amount to sentence level and added named entity and abbreviation types, along words, as basic units of information in the indexing process. By doing this, we could query the IR component not only by keywords extracted from the questions, but also by NE types corresponding to their expected answer types. An example would make this clear: for the question *Where did John Lennon die?* beside creating an IR–query containing the keywords: {`+''John Lennon'', +die`}, we could supply also the expected answer type `LOCATION` querying an additional field neTypes: {`+text:''John Lennon'', +text:die +neTypes:LOCATION`}. This will not only narrow the amount of data being analyzed for answer extraction, but will also guarantee existence of an answer candidate.

4.2 Robust NL question analysis

In context of a QA system or information search in general, we interpret the result of a NL question analysis as *declarative description of search strategy and control information*. Consider, for example, the NL question result presented in figure 3, where the value of tag A-TYPE represents the expected answer type, S-CTR the answer control strategy, and SCOPE additional constraints for the search space. Parts of the information can already be determined on simple local lexico-syntactic criteria (e.g., for the Wh-phrase *where* we known that the expected answer type is `location`), however in most cases we have to consider larger syntactic units in combination with information extracted from external knowledge sources. For example for a definition question like *What is DFKI/a battery?*, we have to combine syntactic and type information from the verb and the relevant NP (e.g., consider definite/indefinite NPs together with certain auxiliary verb forms) in order to distinguish it from a description question like *What is the name of the German Chancellor?*.

```
<IOOBJ msg='quest' s-ctr='C-DESCRIPTION' q-weight='1.0'>
 <A-TYPE>NUMBER</A-TYPE>
 <SCOPE>analphabet</SCOPE>
 <BRELS>
  <BREL rel='GOV' level='0'>
   <ARG1 pos='V'>geb</ARG1>
   <ARG2 pos='N'>analphabet</ARG2>
  </BREL>
  <BREL rel='GOV' level='0'>
   <ARG1 pos='V'>geb</ARG1>
   <ARG2 pos='N'>es</ARG2>
  </BREL>
  <BREL rel='GOV' level='0'>
   <ARG1 pos='V'>geb</ARG1>
   <ARG2 pos='P'>auf</ARG2>
  </BREL>
  <BREL rel='GOV' level='1'>
   <ARG1 pos='P'>auf</ARG1>
   <ARG2 pos='N'>welt</ARG2>
  </BREL>
  <BREL rel='GOV' level='0'>
   <ARG1 pos='N'>analphabet</ARG1>
   <ARG2 pos='WP'>wieviel</ARG2>
  </BREL>
 </BRELS>
 <PRELS>
  <PREL rel='GOV' level='1'>
   <ARG1 pos='N'>welt</ARG1>
   <ARG2 pos='QUANT'>d-det</ARG2>
  </PREL>
 </PRELS>
 <KWS>
  <KW type='UNIQUE'>
   <TK pos='V'>geb</TK>
  </KW>
  <KW type='UNIQUE'>
   <TK pos='N'>analphabet</TK>
  </KW>
  <KW type='UNIQUE'>
   <TK pos='N'>welt</TK>
  </KW>
 </KWS>
 <NEL/>
 <NETS/>
</IOOBJ>
```

Figure 4: The result of the robust NL question interpretation for the example *Wie viele Analphabeten gibt es auf der Welt?* (`How many illiterates are there on the world?`)

In our system, we are doing this by following a two-step parsing schema, where in a first step a full syntactic analysis is performed (cf. sec. 3.2), and in a second step a question–specific semantic analysis. During the second step, the values for the question tags A-TYPE, SCOPE and S-CTR are determined on basis of syntactic constraints applied on relevant NP and VP phrases, and by taking into account information from two small knowledge bases, see also figure 2. They basically perform a mapping from linguistic entities to values of the questions tags, e.g., trigger phrases like *name_of*, *type_of*, *abbreviation_of* or lexical elements to expected answer types, like *town*, *person*, *president*. Note that in case of the German language, we perform a sort of fuzzy match to the knowledge bases taking into account on–line compound analysis and string–similarity tests. For example, assuming the lexical mapping *Stadt*$\Longrightarrow$*LOCATION* for the lexeme `town`, then automatically we will also map the nominal compounds *Hauptstadt* (`capital`), *Großstadt* (`large city`) to the A-TYPE *LOCATION*.

5 Results and Discussion

We have submitted two runs. One for the monolingual German task, and one for the bilingual German/English task. The results are as follows:

Track	#Answ	#T	#F	#Inexact	#Unsup.	Overall Acc	Fact. Acc.	Def. Acc.	NIL prec.
DE-DE	197	50	143	1	3	25.3%	28.25%	0	13.6%
DE-EN	200	47	151	0	2	23.5%	23.8%	20%	10.79%

Compared to our results obtained at QA@Clef2003 this is a good improvement because the tasks were more difficult and because we could use nearly the same system for both, the bilingual track as well as the monolingual track. We will now discuss the results for the two individual tasks, comparing them where possible.

In both cases, we only considered answers which directly where recognized as NE instances, i.e., for all questions which would refer to more general noun phrases or to NE types and instances LingPipe did not recognize, we did not identify any answer candidates. Note that although in both tasks we were able to properly analyze all definition questions as such, in our current system we only determine possible answer candidates for abbreviation based questions (by way: not such questions were found in the German test set, which explains, why we did not recognize any definition question). The fact, that we did not answer definition question (modulo abbreviation) correlates with our restrictions to only consider NE instances as answer candidates.

As previously mentioned, both the monolingual and bilingual task have shared the same QA-framework, which was presented above. Nevertheless, there were task specific system configurations, resulting in different retrieval and answer extraction methods, which will shortly be mentioned in the following lines.

Monolingual Task For the German monolingual task we were able to have the system recognize named entity instances of type *NUMBER*, as result of training LingPipe on a German corpus with a larger coverage of named entity types than its English counterpart. Even though the indexing method was similar for both tasks, the monolingual task did not make any use of the named entity type field (neType) during information search.

Bilingual Task No questions with a *MEASURE* expected answer type were considered, because the bilingual settings were not able to identify named entity instances of type *NUMBER*. The system used a similarity function, which compared to the monolingual task, resembles a co–reference algorithm by identifying answers mentioning the same NE instance in the answer candidate set (e.g., “Bill Clinton” and “Clinton” will count as two references to the same person).

Acknowledgement

The work presented in this paper has been funded by the BMBF project Quetal, FKZ 01 IW C02. Many thanks to Jumamurat Bayjanov and Olga Goldmann for their implementation support.

References

[Mil00] D. Milward. Distributed representation for robust interpretation of dialogue utterances. In *Proceedings of the ACL-2000*, pages 133–141, Hong Kong, 2000.

[MPHS02] Dan Moldovan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the ACL-2002*, pages 33–40, Philadelphia, 2002.

[NBB+97] G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *ANLP 97*, pages 208–215, Washington, USA, March 1997.

[NP02] G. Neumann and J. Piskorksi. A shallow text processing core engine. *Journal of Computational Intelligence*, 18(3):451–476, 2002.

[NS03] Günter Neumann and Bogdan Sacaleanu. A cross-language question/answering-system for german and english. In *proceedings of the CLEF 2003 working notes of the QA@CLEF, Trondheim, August*, 2003.