A Cross–Language Question/Answering–System for German and English

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 2003 edition of the Cross-Language Evaluation Forum (CLEF). We have participated in the new track "Multiple Language Question Answering (QAatCLEF)" that offers tasks to test monolingual and cross-language QA–systems. In particular we developed an open–domain bilingual QA–System for German source language queries and English target document collections. Since it was our very first participation at such kind of competition, the focus was on system implementation rather than system tuning.

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), and eventually express the found answers in the query language (German).[1] 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 textual fragments representing the answer (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.

At the Language Technology Lab of DFKI we have begun the development of large–scale open–domain cross–language QA systems, currently with a focus on German and English. In [NX03] we have described a first prototype of a monolingual Web–based QA–system that processes German queries and Web pages (using Google for initial web page retrieval). On basis of this initial prototype we have implemented BiQue a German–English bilingual textual QA–system. BiQue receives a German language query, parses and translates it into English, and searches for answers in a large English text collection maintained by the full–text search engine MG [WMB99].

The main motivation for our participation at this year's CLEF was to foster development of an initial end–to–end cross–language QA–system enforced by external evaluation. Since, we also plan to extend the system for English query and German document analysis (and to support mixed language mode), we have focused on the development of common LT-core components for bilingual query and answering processing that enable us to easily improve the system in the future. Thus we also focused on the development of generic APIs (based on XML) and knowledge formalisms that helps us to systematically improve our system in next development cycles.

We start with an overview of the whole system, highlight some technical aspects, followed by a more detailed description of the methods we used for query translation and expansion. Finally, we present the results we have obtained for the task.

[1] The translation of answers into the query language is currently not part of the QAatCLEF track, hence we will say nothing about this problem here.

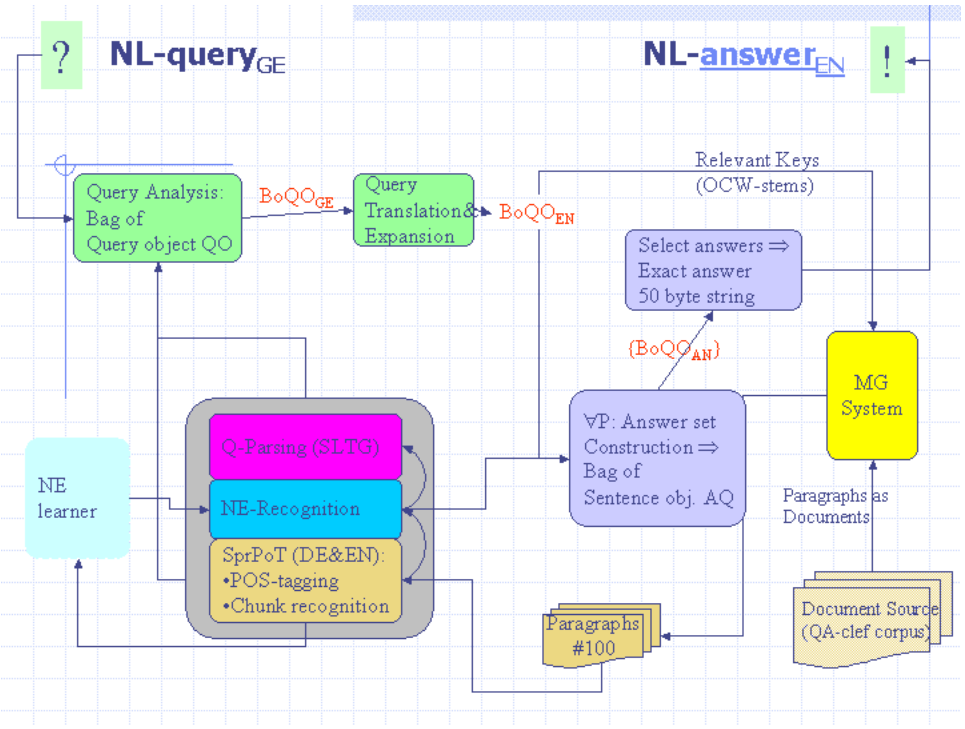

Figure 1: A blueprint of BIQUE's architecture.

2 System overview

The picture in Figure 1 displays the control flow between the major components of BIQUE. The major control flow is basically state–of–the–art and — from a coarse–grained point of view — not novel. However, we think that we have realized a number of interesting "sub–issues" and an interesting "translation approach" (with hopefully fruitful future impacts, at least for us ;-) which is motivation enough to give some more details here.

2.1 Document retrieval

We are using the MG system — a public–domain full–text retrieval engine, cf. [WMB99] — for the selection of relevant paragraphs. MG is an easy useable software package that can handle text corpora of several Gigabytes very efficiently. In order to make use of the MG system in the context of the QAatCLEF track, we actually had to solve two problems:

- How can we keep track of the document identifier?
- How can we use MG for the selection of relevant short text passages?

The first issue is important because for each answer candidate one has to indicate the document from which the answer was extracted. Secondly, only a small fragment of the documents need to be processed more deeply in order to identify possible answer candidates. In order to fulfill both requirements when using MG, we performed a simple preprocessing of the text corpus: we attached to the front of each paragraph (identified by means of the P SGML tag) of a document a status line which represents the document identifier and the number of the paragraph in the document. Each such extended paragraph is then treated as a single document by MG. Thus, given a set of keywords as input to MG it will return a set of paragraphs where each paragraph

encodes its location within the original text document.[2] Here is example of a paragraph returned by MG for the query "leader, india":

> ######## LA110594-0041 10 ######## .
>
> The official Indian position has changed a few times but basically has been that all the stones, or the most remarkable among them, should stay in India. India's leaders, however, haven't had the cash to purchase them.

MG supports document retrieval by either using a boolean query or a ranked query. We decided to use the ranked query because MG should only return paragraphs (see above), and hence a boolean query would be too restrictive already in an early processing phase. Currently, we use the first 100 paragraphs returned by MG, basically because of performance reasons.

2.2 Shallow syntactic processing

NL queries and documents are linguistically analyzed using ShProT, a shallow processing tool that consists of several integrated components: SPPC for tokenization and analysis of compound words (cf. [NP02]), TnT for part–of–speech tagging (cf. [Bra00]), Mmorph for morphological analysis (cf. [DG94]) and Chunkie for phrase recognition (cf. [SB98]). TnT and Chunkie are statistical based components which derive the linguistic entities, rules and generalizations from annotated corpora. The language models are based on the Penn treebank (for English) and the Negra treebank (for German). ShProT receives as input an ascii text and returns a stream of sentences each consisting of a sequence of tagged phrases and tagged wordforms. The tagged phrases actually define the type of the phrase (either NP or PP) and consists of a sequence of tagged wordforms. A tagged wordform contains the POS, and the lemma as determined by Mmorph. For unknown words (which also includes proper names) TnT tries to guess the POS. In case of a proper name, these are tagged with generic tags like NNP (for singular proper noun). Figure 2 shows the XML—representation of the shallow analysis of an example sentence.

2.3 Named Entity Recognition

Our Named Entity Recognition (NER) method is based on the unsupervised learning approach of [CS99]. A decision list of NER–rules (also represented in XML) is applied on the XML–output of ShProT and performs an additional annotation of relevant NPs with corresponding NE–type information (currently, we consider the NE–types PERSON, ORGANIZATION, LOCATION, TIME, and DATE). Currently, only NPs that contain at least one word recognized by ShProT as a proper noun or time/date expression will be considered as candidates for NE–typing.[3] All these NE candidate phrases are then further processed by the decision list matcher. Each element of the decision list is a simple IF–THEN rule. If a NP–candidate fulfills some spelling or contextual conditions (these are based on generic syntactic criteria of adjacent phrases like "capitol of XXX") then it receives the NE–type as indicated by the rule (e.g., in the example case XXX is typed as LOCATION).

Our current NER–learner is still under development. Usually the decision list is automatically learned. However, for the QAatCLEF track we have not been able to perform a complete training phase because preprocessing of the QA–corpus turned out to be too expensive (it will be one future research issue to explore more efficient learning methods). For that reason a number of rules of the decision list were specified manually. In order to compensate possible (and actual) recall problems we combined the decision list with external Gazetters.

[2] By way: since the status line is part of a paragraph it can also be specified as part of the query. Hence, we also can use the status line information for reconstructing the whole document as well as for performing corpus navigation using MG.

[3] Note that this means that we perform NER after shallow parsing. Hence the accuracy of the NER depends on the accuracy of shallow parsing. In some sense the approach can also be viewed as a top–down classification approach, since NER–module actually performs a sub–typing of those generic NE types already recognized by the shallow processor.

```
<SENTENCE id="S4">
   <CHUNK id="H26" cat="NP">
    <W id="W99" PoS="DT" tclass="25" mclass="24" stems="[a]">
     <WORDFORM string="An" />
     <READINGS>
      <R id="R0" subtype="art_indef" category="Det" />
     </READINGS>
    </W>
    <W id="W100" PoS="NNP" tclass="22" mclass="-1" stems="[]">
     <WORDFORM string="FBI" />
     <READINGS />
    </W>
    <W id="W101" PoS="NN" tclass="24" mclass="29" stems="[informant]">
     <WORDFORM string="informant" />
     <READINGS>
      <R id="R0" subtype="char" category="Abbr" />
     </READINGS>
    </W>
   </CHUNK>
   <W id="W102" PoS="VBN" tclass="24" mclass="205" stems="[claim]">
    <WORDFORM string="claimed" />
    <READINGS>
     <R id="R0" subtype="main" tense="past" verbclass="intrans" category="Verb" vform="psp" />
    </READINGS>
   </W>
   <W id="W103" PoS="IN" tclass="24" mclass="-1" stems="[that]">
    <WORDFORM string="that" />
    <READINGS />
   </W>
   <W id="W104" PoS="NNS" tclass="25" mclass="-1" stems="[]">
    <WORDFORM string="Wilkins" />
    <READINGS />
   </W>
   <W id="W105" PoS="VBD" tclass="24" mclass="-1" stems="[be]">
    <WORDFORM string="was" />
    <READINGS />
   </W>
   <CHUNK id="H27" cat="NP">
    <W id="W106" PoS="DT" tclass="24" mclass="399" stems="[the]">
     <WORDFORM string="the" />
     <READINGS>
      <R id="R0" wh="no" subtype="gen" number="plural" category="Det" />
     </READINGS>
    </W>
    <W id="W107" PoS="NN" tclass="24" mclass="1" stems="[trig, German]">
     <WORDFORM string="triggerman" />
     <READINGS>
      <R id="R0" gender="neutrum" number="singular" category="Noun" />
     </READINGS>
    </W>
   </CHUNK>
   <W id="W108" PoS="$." tclass="1" mclass="-1" stems="[$PUNCTUATION]">
    <WORDFORM string="." />
    <READINGS />
   </W>
  </SENTENCE>
```

Figure 2: The XML–representation of the shallow syntactic analysis for the sentence *An FBI informant claimed that Wilkins was the triggerman .*

2.4 Internal query and document representation

Internally, queries and documents are uniformly represented as weighted sets of structured (possibly linked) objects in order to facilitate a robust and efficient comparison between queries and answer candidates. More formally, we call the set $B := \{O_1, \ldots, O_n; \alpha\}$ a *Bag–of–Objects* or short BoO consisting of n objects O_i and weight α. Each object O_i is a tuple of the form $\langle WF, Stem, PoS, NE, \alpha_i \rangle$, i.e., a structured object consisting of a word form, a lemma, part–of–speech, named entity and weight α_i (note that for all elements but WF and α_i the actual value can be empty).

The weight of a BoO is determined during the matching phase of the query with a candidate answer sentence. The actual approach we are exploiting for comparing and merging two different BoOs is a variant of the *word overlap* method described in [LMRB01]. A word overlap (which is also a BoO in our case) is the subset of objects a query and an answer candidate have in common, i.e., the word overlap of two sentences s_1 and s_2 is $Ov_{s_1,s_2} := B_{s_1} \cap B_{s_2}$, where B_i is the BoO of s_i. The weight β of a word overlap Ov is determined as the sum over the weights α_i of the overlapping words.[4] After Ov_{s_1,s_2} has been computed, the B_i obtain β as their weight, i.e., BoO with same word overlap have equal weight (however, this weight will later be updated using the expected answer type, see below).

We also define the *overlap set* Os_q of a query q as the set of all BoOs of all candidate answer sentences which have the same word overlap with q, i.e., $Os_q := \{B_{s_1}, \ldots, B_{s_n}\}$, with: $Ov_{q,s_i} = Ov_{q,s_j}$ for $i \neq j$. This means that the overlap sets define equivalence classes over the set of possible answer candidates wrt. the set of objects each answer has in common with the query, i.e., query and sentences with same word overlap (and hence, with equal weight, but see 2.6).

2.5 Query processing

The main tasks of the query processor are the

1. parsing of a German (or English) NL query, and the
2. translation and expansion of the German query object to an English one.

A query object is a tuple $\langle EAT, BoO, Keys, L \rangle$ consisting of the expected answer type EAT, the BoO representation of the question, the set of relevant keywords used as query for the full–text retrieval engine MG (see 2.1), and the language identifier. We will now describe very briefly the different steps, and will only say a bit more on parsing in the subsection that follows.

The main goal of parsing a NL query in the context of open–domain QA is the identification of the question focus and the expected type (or concept) of the potential answer phrase (Expected Answer Type (EAT), cf. [HMP+00]). The question focus is a phrase or word in the question that can help to disambiguate it and — together with the question stem (e.g., *who*, *how much*, *where*) — can help to deduce the EAT.

After the parser has determined the EAT for the current question, a BoO representation is constructed on basis of all content words of the question. (In principle, it is also possible to link the elements of the BoO based on the derivation tree computed during parsing (which corresponds roughly to a dependency tree, see next paragraph). However, we have not been able to finish implementation of this further step in due time which would have helped us to define more clever strategies for the identification of exact answers, see below 2.6.) So we currently have to live with a quite flat internal representation of the query. The set of relevant keywords is determined very simply from the BoO by collecting all stems of the content words (or word forms if no such stem could have been computed).

Finally, query translation and expansion takes place in order to perform retrieval of English paragraphs and to allow for computation of overlap sets on basis of word overlap between the query and answer candidates. A description of details of this third step during the analysis of a question is postponed until section 3. The only thing worth to mention here, is that the German query

[4] The weight of an individual object is currently specified a priori and is based on the word's part–of–speech.

BoO is basically translated to an English one (by keeping the EAT determined for the German query). Translation is basically realized by means of "merging" results from EuroWordNet with the results of externally available translation services which we are using as a means for performing word sense disambiguation. Query expansion is performed simultaneously with query translation.

Query parsing Before going on in describing how answer processing is performed on basis of the translated query object, we describe some details of our parsing methods.

In our current system, we have specified manually a German and a English query grammar in form of *lexicalized tree substitution grammars* (LTSG). A query LTSG consists of set of syntax/semantics oriented tree patterns which express mutual constraints for the identification of a question focus and an EAT. Here is an example of such an elementary tree:

```
<tree id="6a" label="F-Wo" eat="LOCATION" freq="" prob="">
     <node label="PWAV">
         <node label="wo" type="TERM" anchor="YES"/>
     </node>
     <node label="VVFIN">
         <node label="schliessen" type="TERM" anchor="YES"/>
     </node>
     <node label="NE" nclass="PERSON" type="SUBST"/>
     <node label="NP" type="SUBST"/>
     <node label="PTKVZ">
         <node label="ab" type="TERM"/>
     </node>
</tree>
```

which would be applicable for a question like *Wo schloss Hillary Clinton das College ab?* (*Where did Hillary Clinton graduate college?*). A query grammar is applied on top of the shallow chunk analysis computed by first applying ShProT on the NL question. Note that nodes of type TERM are lexical anchors and nodes of type SUBST have to be expanded by substituting the node with a consistent (complete) phrase. Parsing of a query LTSG is performed along the line of the method described in [Neu03].[5]

In some sense, the elementary trees of a LTSG define clause–level patterns using lexical information about the question type and focus to constraint their applicability. Linguistically, an elementary tree of a LTSG also describes a head–modifier relationship between the lexical anchors and the modifiers (basically the substitution nodes). Hence a derivation of a query analysis can also be used to uncover the dependency structure.[6] The current grammars (together with the possible supported EAT) have been defined on the basis of a manually translation of the QA–Trec 8 and 9 question corpus. Actually, it turned out that the current grammars have been defined a bit too Trec–8/9 specific concerning supported subcategorization. Hence, future work will focus on improving generalization without loosing the benefits of lexicalized tree structures.

2.6 Answer processing

Paragraph selection The keywords of the translated query object are used to build a query expression for the MG system. Currently, we use the whole set of keywords (including the expanded terms) to form one MG–query. The ranked query mode of MG is then used to retrieve the N best paragraphs (see also 2.1). In the ranked query mode, MG actually ranks all documents according to some similarity measure applied on each document which specifies how close the document matches with the query. Thus seen, MG returns the N most relevant documents with respect to the query.

[5] One of the reasons why we have chosen an LTSG approach is our future goal, to automatically extract a linguistically expressive but specific query subgrammar form a large–scale general HPSG–source grammar following the approach described in that paper.

[6] Following the approach of [HMP+00], it would then be possible to construct a quasi logical form from the dependency relation in order to support theorem proving for answer validation.

Candidate answer selection All retrieved paragraphs are analysed by ShProT (see 2.2) which maps a paragraph into a sequence of sentence objects. A sentence object consists of the shallow syntactic XML–structure, the sentence BoO (constructed from it) and additional bookkeeping information (e.g., pointer to document identifier).

All sentence objects of every paragraph are collected into one container from which the overlap sets are constructed along the line described in 2.4. This means that a word overlap Ov_{q,s_i} is computed by merging the BoO of the query q with the BoO of every sentence object s_i, which is then used to construct and rank the overlap sets. In a next step, all sentence objects from the top five equivalence classes are collected into one list of answer candidate sentences. For each such sentence object it is then check, whether it contains one element which is type–compatible with the expected answer type EAT of the query object. If so, the weight of the sentence is increased. Note that this means that a sentence whose corresponding word overlap weight is smaller than that of another sentence (which means that it is less similar wrt. to the query) might now receive a higher rank.

In a final step, each sentence is searched for an NP phrase which can serve as the exact answer of the question. The method that we have exploited so far, actually constructs a ranked list of all NPs (extracted from every sentences) that do not contain any element from the sentence's word overlap. Ranking is performed by taking into account the type of the NP (e.g., EAT–compatible, containing other NEs), and the number and distance of elements from the sentence's word overlap wrt. the NP. By doing so we determine exact answer and 50bytes answer strings. Note that the underlying assumption made by our current method is that the strings of NPs serve as exact answers. Generally, this view is surely too restricted (and might only apply for certain kind of questions), and hence will be improved in the future.

3 Query translation and expansion

In this section we are going to describe in more detail how question translation and expansion is performed.

Background Traditional approaches for cross–language information management systems can be classified as follows:

1. systems that translate the queries into the target language, or
2. the document collection into the source language, or both,
3. queries and documents into an intermediate representation (inter–lingua).

Two types of translation services are well known within this context which are based on

- lexical resources (e.g., dictionaries, aligned wordnets), or
- machine translation (e.g., example–based translation).

Each translation method has to deal with the following issues: *word sense disambiguation* (WSD) and *coverage*. WSD accounts for translating the appropriate meaning of a word, as suggested by its context, while coverage guarantees that source language words have a chance to be translated, to the extent to which it is intended (e.g., not all named entities should be translated). In retrieving the documents related to a formulated query, it is often useful to take into consideration words related to the query words. This *query expansion* method can be achieved either through syntactic or semantic variations. A query of the form "presidential election" could be extended with "election of the president", "the president was elected", "presidential vote", "presidency vote"", etc. An issue in query expansion is the word sense disambiguation, too. As query words may be ambiguous, only the intended meanings of them should be targets of the expansion task.

Method The system BIQUE as used in this competition translates the German language question to the English language of the document collection by means of machine translation techniques. The system accounts for the above-mentioned coverage issue by using three different translation services: FreeTranslation, Altavista and Logos. The results of translating the original German question are used in generation of bag-of-object (BoO) collections of English open-class words, which are further on target of the query expansion module. Expansion is being achieved only through semantic variations using WordNet-like resources, whereby a pseudo word-sense-disambiguation task using the German original question and its English translations is being applied. Following we will describe the functioning of the question translation and expansion module by means of the example question:

Wo wurde das Militärflugzeug Strike Eagles 1990 eingesetzt?

Question Translation Three different translation services have been considered for this purpose:

- FreeTranslation (via http://www.freetranslation.com/) yields:
  "Where did the military airplane become would strike used Eagles 1990?"
- Altavista (via http://babel.altavista.com/) yields:
  "Where was the military aircraft Strike Eagle used 1990?"
- Logos (off–line) yields:
  "Where was the soldier airplane Strike Eagles installed in 1990?"

Initial experiments using only one translation service unveiled the limitation imposed by the coverage problem: inadequate or no translations (e.g., some name of countries that were different in German and English). Extending the translation module with two further services, the results improved and pointed out the advantage of indirectly using it for question expansion as well, as different translations can generate synonym words. Moreover, the original German question and its English translations were used for question expansion too, as building blocks for our pseudo-WSD module.

Given the above-listed translations, a BoO collection of open-class normalized words has been created, with the following content (for convenience we abbreviate the object elements by means of their lexemes):

{SOLDIER, AIRPLANE, STRIKE, EAGLE, INSTALL, 1990, MILITARY, BECOME, STRIKE, USE, AIRCRAFT, EAGLE}

This BoO is obtained as follows: from each English version of the question a corresponding BoO is constructed by applying ShProT and the English question grammar. The resulting different BoO's are then merged into one BoO which represents the translated query object (re–using the expected answer type EAT as computed for the German question analysis).

Question Expansion For the expansion task we have used the German and English wordnets aligned within the EuroWordNet lexical resource. Our goal was to extend the English BoO collection with synonyms for the words that are present in the wordnet.

Considering the ambiguity of words, a WSD module was required as part of the expansion task. For this purpose we have used both the original question and its translations, leveraging the reduction in ambiguity gained through translation. Our devised *pseudo-WSD algorithm* works as following:

1. look up every word from the translated BoO collection (see example above) in the lexical resource;

2. if the word is not ambiguous (which is, for example, the case for AIRPLANE, AIRCRAFT) then extend the BoO collection with its synonyms, e.g.,
   AIRPLANE $\Longrightarrow$ ⟨AEROPLANE, PLANE⟩
   AIRCRAFT $\Longrightarrow$ ⟨ ⟩ (i.e., in case of AIRCRAFT there are no synonyms);

3. if the word is ambiguous (e.g., USE) then

   (a) for every possible reading of it, get its aligned German correspondent reading (if it exists) and look up that reading in the German original question (i.e., in the BoO representation of the original German question "Wo wurde das Militärflugzeug Strike Eagles 1990 **eingesetzt**?"), e.g.,
   Reading-697925:
   EN: ⟨HANDLE, USE, WIELD⟩, DE: ⟨HANDHABEN, HANTIEREN⟩
   Reading-1453934:
   EN: ⟨BEHAVE TOWARD, USE⟩, DE: not aligned
   Reading-661760:
   EN: ⟨BE A USER OF, USE, USE REGULARLY⟩, DE: not aligned
   Reading-658041:
   EN: ⟨EXPEND, USE⟩, DE: ⟨AUFWENDEN⟩
   Reading-658243:
   EN: ⟨APPLY, EMPLOY, MAKE USE OF, PUT TO USE, USE, UTILISE, UTILIZE⟩,
   DE: ⟨ANBRINGEN, ANWENDEN, BEDIENEN, BENUTZEN, **einsetzen**, . . . ⟩

   (b) if an aligned reading is found (e.g., Reading-658243) retain it and add the English synonyms of it to the BoO collection, i.e., expand it with:
   ⟨APPLY, EMPLOY, MAKE USE OF, PUT TO USE, USE, UTILISE, UTILIZE⟩

Following the question expansion task, the BoO collection has been enriched with new words that are synonyms of the un–ambiguous English words and by synonyms of those ambiguous words, whose meaning(s) have been found in the original German question. Thus our expanded example looks as follows:

{SOLDIER, AIRPLANE, STRIKE, EAGLE, INSTALL, 1990, MILITARY, BECOME, STRIKE, USE, AIRCRAFT, EAGLE, AEROPLANE, PLANE, APPLY, EMPLOY, MAKE USE OF, PUT TO USE, USE, UTILISE, UTILIZE}

4 Results and conclusion

We have participated for the first time in a QA track, and hence had to build BIQUE from scratch, so the focus was on system implementation, rather than on system tuning (and actually we had no time to test different settings of critical system parameters, like the weighting values). We submitted only one run for the 50byte run, and obtained as result for the strict statistics 14.5% correct answers, and 15% for the lenient statistics.[7] This is surely a result that should and can be improved.[8] Besides evaluation of the performance of the system wrt. different parametrization, important next steps for system improvement are, among others, the unsupervised online learning of more fine–grained NE rules, Machine Learning of query grammars, methods for determining the utility of answer candidates, development of ontology based answer validation methods, and more controlled query expansion by using fine–grained ontologies (following [HGH+00]).

[7] We had also planned to submit a second run by first preprocessing the whole corpus with ShProT, but it turned out that this was too time consuming. The major motivation was, that we wanted to perform a stemming of the complete corpus by using ShProT instead of the build in stemmer of MG which turned out to cause too much trouble in some cases.

[8] For example we were not able to process questions containing quoted terms, because we simply have not foreseen such questions. However, 20% of the test set contained such kind of questions.

References

[Bra00] Thorsten Brants. Tnt – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, Seattle, WA., 2000.

[CS99] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.*, Association for Computational Linguistics, 1999.

[DG94] Petitpierre D. and Russell G. Mmorph - the multext morphology program. Technical report, ISSCO, University of Geneva, 1994.

[HGH+00] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question answering in Webclopedia. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.

[HMP+00] Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. FALCON: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.

[LMRB01] Marc Light, Gideon S. Mann, Ellen Rilo, and Eric Breck. Analysis for elucidating current question answering technology. *Natural Language Engineering*, 7(4), 2001.

[Neu03] Günter Neumann. Data-driven approaches to head-driven phrase structure grammar. In Rens Bod, Remko Scha, and Khalil Sima'an, editors, *DATA-ORIENTED PARSING*. CSLI Publications, University of Chicago Press, 2003.

[NP02] Günter Neumann and Jakub Piskorski. Shallow text processing core engine. *Computational Intelligence*, 18(3):451–476, 2002.

[NX03] Günter Neumann and Feiyu Xu. Mining Answers in German Web Pages. In *Proceedings of The International Conference on Web Intelligence (WI 2003)*, Halifax, Canada, October 2003.

[SB98] Wojciech Skut and Thorsten Brants. A maximum entropy partial parser for unrestricted text. In *6th Workshop on Very Large Corpora*, Montreal, Canada, August 1998.

[WMB99] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.