The *Alyssa* System at TREC 2006: A Statistically-Inspired Question Answering System

Dan Shen   Jochen L. Leidner   Andreas Merkel   Dietrich Klakow
Spoken Language Systems
Saarland University
D-66125 Saarbrücken, Germany
`lsv_trec_qa@lsv.uni-saarland.de`

Abstract

We present our new statistically-inspired open-domain Q&A research system that allows to carry out a wide range of experiments easily and flexibly by modifying a central file containing an experimental "recipe" that controls the activation and parameter selection of a range of widely-used and custom-built components.

Based on this, we report our experiments for the TREC 2006 question answering track, where we used a cascade of LM-based document retrieval, LM-based sentence extraction, MaxEnt-based answer extraction over a dependency relation representation followed by a fusion process that uses linear interpolation to integrate evidence from various data streams to detect answers to factoid questions more accurately than the median of all participants.

1 Introduction

This paper describes *Alyssa*, a new open-domain question answering (Q&A) system developed at Saarland University.[1] It was built as a tool for the investigation of Q&A from a more principled, i.e. information theoretically well-founded approach. Another requirement was the construction of a software that can serve as an experimental platform in the longer term. Therefore, our development efforts were focused on flexibility and modularity rather than performance tuning in the first year of *Alyssa* participation at TREC.

[1]The system is named after Alyssa, the departmental secretary's terrier dog and system mascot.

We report our experiments for the TREC 2006 question answering track, where we used a cascade of LM-based document retrieval, LM-based sentence extraction, maximum entropy based answer extraction over a dependency relation representation followed by a fusion process that uses linear interpolation to integrate evidence from various data streams to detect answers to factoid questions more accurately than the median of all participants.

The remainder of this paper is structured as follows: Section 2 describes the architecture of our system. In Section 3, we describe the specific methods used for TREC 2006, and Section 4 presents our results. Finally Section 5 concludes the paper with a summary and possible future work.

2 Architecture

2.1 Design Choices

As pointed out in (Leidner, 2003), software engineering in speech & language technology is constrained by the development resources available. According to (Cunningham, 2000), the "creation of software infrastructure must be undertaken in conjunction with the development of systems on which the infrastructure is based and which will in turn use its services". In this case, our challenge was to construct a reusable and extensible software system while still being able to deliver first experiments for TREC 2006, to the same deadline. As a consequence, we had to find a feasible minimum-overhead architecture without producing yet another ad-hoc "solution".

To this end, a modular architecture was chosen that comprises an array of components that are remote-controlled by a central driver module. The

```
#--------------------------
# Define settings for doc. IR
#--------------------------
[document_retrieval]
# or: lucene, terrier
engine=lemur
# or: tdidf dfr lm_dir lm_abs lm_jm
method=lm_dir
top_n=60
index=default

# LSVLM options -------------------
# 0=TFIDF 1=OKAPI 2=Kullback-Leibler
lemur_model=2
dirichlet_prior=1000
# smoothing factor:
jelinek_mercer_lambda=0.5
discount_delta=0.5
```

Figure 1: A central experimental description facilitates experiments.

crucial insight is that the driver knows about the components, but the components need not know about the existence of the driver. Our design was informed by the design of the *QED* system ((Leidner et al., 2004)) and other systems, where the kind of processing undertaken is encoded as sequences of file name extensions.

We designed the core driver module so as to read a global configuration file (Figure 1) describing all component's activation status (on or off) and the settings of all their parameters in a central place. We use many external tools widely shared across NLP researchers, and they often have their own parameter files, which we generate automatically from the master "recipe", which serves a dual purpose: first, the recipe preserves the details of the experiment as it was run for later inspection (*provenance*). Second, even the master recipe can later be auto-generated (for example in the future by a graphical GUI) using varied parameter settings in order to explore potentially better component constellations than the ones we currently use.

The file layout and naming conventions together with the recipe form the backbone of the *Alyssa* system architecture. Next, we describe how this infrastructure is currently populated by components.

2.2 Result

Figure 2 shows the resulting architecture of *Alyssa* with its various component modules and data streams.

The question first undergoes question analysis, a phase in which several modules are involved independently. The type of the question is determined and a linguistic analysis is carried out, including full and shallow parses and named entity tagging. Then a query is constructed from the question based on this analysis. The query is run against document retrieval on the AQUAINT index and passage retrieval on a Wikipedia index. An optional co-reference step allows pronouns or NPs to be replaced by their antecedents. Alternative sentence extraction strategies have been implemented based on language modeling and windowing techniques, and extracted sets of sentences undergo further linguistic analysis before being fed into two answer extraction module, one based on patterns, another one based on a supervised machine learning method using dependencies (see below). Special modules take care of event questions, definition questions, and list questions. As a result, several candidate answer streams emerge that are integrated in a answer validation and fusion step.

This architecture and module inventory is a superset of what was actually used for TREC 2006, as will become evident in the subsequent experimental description, in part because some modules are still in development, in part because some modules were ready but we simply did not have enough time to carefully test their contribution to performance was beneficial. For example, the Web validation and anaphora resolution modules were not used for TREC 2006 yet.

3 Methods

3.1 Question Analysis

We process questions as follows:

Preprocessing: We apply a simple anaphora resolution strategy for a question: 1. We replace all pronouns in the question with main NP of its target (TNP). 2. For each NP in the question, if it has same head word as the target and shorter length than the target, we replace it with TNP. 3. If the question doesn't contain the target after the previous steps, we choose a noun phrase in the question based on rules and replace it with TNP.

Expected Answer Type (EAT) Extraction: We chunk questions using Abney's chunker(Abney, 1989). Based on chunk sequence of question, we identify a NP chunk which indicates EAT of the question using rules, such as, for Q: *Which terrorist organization claimed*

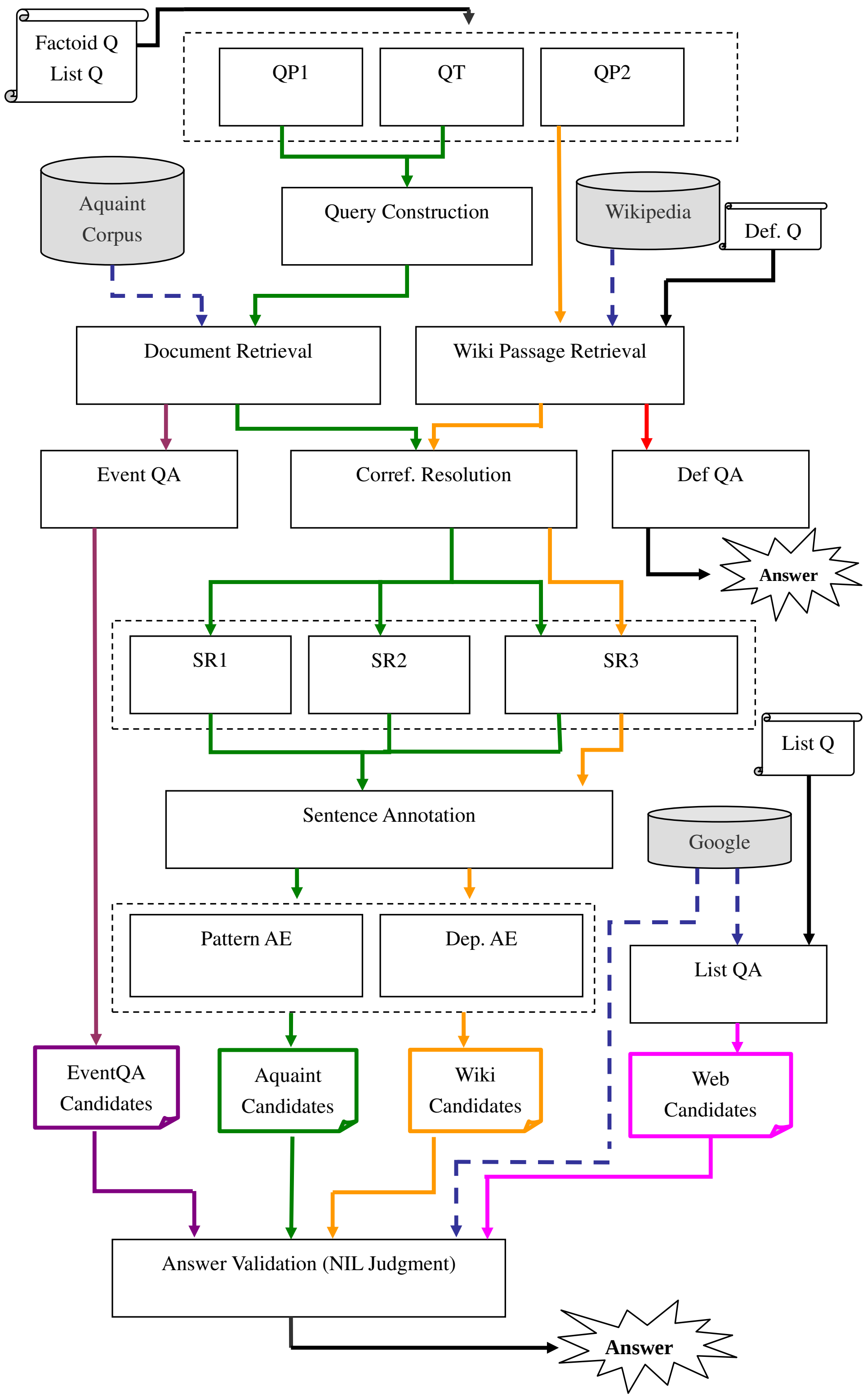

Figure 2: Architecture of *Alyssa*.

responsibility for the massacre?, which chunk sequence is *which np0 vb0 np1 for np2 ?*, the first chunk *np0*(*terrorist organization*) following *which*, is identified as EAT. Furthermore, the head of EAT is mapped to a hand-built ontology to obtain the corresponding named entity (NE) type (e.g. *organization* corresponds to NE type *ORGANIZATION*). In the current system, about 50 named entity types (listed in Figure 3) are considered.

```
PERSON, ORGANIZATION (PARTY / UNIVERSITY / BUILDING / OTHER),
LOCATION (CITY / COUNTRY / CONTINENT / MOUNTAIN / ISLAND /
LAKE / OCEAN / PENINSULA / RIVER / STRAIT / VOLCANO / OTHER),
NUMBER (COUNT / ORDER / MONEY / PERIOD / PERCENT / DISTANCE /
SPEED / TEMPERATURE / SIZE / AREA / VOLUME / WEIGHT / OTHER),
DATE, COLOR, CURRENCY, LANGUAGE, MUSIC, OCCUPATION, RELIGION,
SPORT, WAR, ANIMAL (BIRD / INSECT / MAMMAL / OTHER), PLANT
(FLOWER / TREE / OTHER), ABBREVIATION, SYMBOL, POSTCODE,URL
```

Figure 3: List of NE types.

Question Pattern Matching: Considering that there are questions with very high frequency to be asked in TREC, we build question patterns (QPTN) to map high frequent questions to classes. Different from (Kaisser and Becker, 2004) and (Wu et al., 2005), question classes are defined in terms of meaning but not syntactic structures of questions. For each question class, we also build answer patterns (APTN) to extract answers for questions in the class, as described in Section 3.5.1. Figure 4 shows QPTNs for question class *WHAT_CREATION*. Given a question *Q* and a question pattern *QP*, if the chunk sequence of *Q* is matched to *BFORM* of *QP*, and if the key chunks of *Q* satisfies all constraints *CON* listed in *CONS* of *QP*, then *Q* is matched to *QP* and belongs to then corresponding class. Furthermore, *SLOTS* of *QP* records which key chunks of *Q* will be used in answer pattern matching. 23 question classes, such as *WHO_CREATION*, *WHEN_CREATION*, *WHEN_BORN*, are considered in the system. In TREC 2006, 92 among total 403 factoid questions are classified to question classes.

Dependency Relation Path Extraction: For the remaining questions that are not classified, we use a statistical model to rank candidate answers. The core idea of the model (Shen and Klakow, 2006) is to compare dependency relations between questions and answer sentences. In question analysis, we use MINIPAR (Lin, 1994) to parse questions and extract dependency relation paths between question words, such as *what*, *who*, *when*, and all key chunks of questions.

```
<CLASS key="WHAT_CREATION">
<QPTN_SET>
   <QPTN>
      <BFORM>what do np0 vb0 ?</BFORM>
      <CONS>
         <CON key="vb0">var0</CON>
      </CONS>
      <SLOTS>
         <SLOT key="slot0">np0</SLOT>
      </SLOTS>
   </QPTN>
   <QPTN>
      <BFORM>what be np0 of np1 ?</BFORM>
      <CONS>
         <CON key="np0">var1</CON>
      </CONS>
      <SLOTS>
         <SLOT key="slot0">np1</SLOT>
      </SLOTS>
   </QPTN>
   <QPTN>
      <BFORM>what be np0 's np1 ?</BFORM>
      <CONS>
         <CON key="np1">var1</CON>
      </CONS>
      <SLOTS>
         <SLOT key="slot0">np0</SLOT>
      </SLOTS>
   </QPTN>
</QPTN_SET>
<VARS>
   <VAR key="var0">found|establish|form|organize|create|
                build|invent|discover|design</VAR>
   <VAR key="var1">founding|creation|invention|discovery</VAR>
</VARS>
</CLASS>
```

Figure 4: Example of question patterns for class *WHAT_CREATION*.

3.2 Question Classification and Typing

For question classification, we use the taxonomy as proposed in (Li and Roth, 2002). It uses 6 coarse and 50 fine grained classes. In our system we used the fine grained classification only because we use specific sub-classes (like DATE, which is a specific sub-class of NUMBER) later on in our sentence retrieval. We use the 5,500 questions provided by the Cognitive Computing Group at University of Illinois at Urbana-Champaign[2] for training and the TREC 10 data for evaluation.

In terms of classification paradigm we start with the Bayes classifier

$$\hat{c} = \mathrm{argmax}_c P(Q|c)P(c) \qquad (1)$$

which is known to produce the minimum number of misclassifications if the correct probabilities

[2]http://l2r.cs.uiuc.edu/∼cogcomp/Data/QA/QC/

are known (Q is the question and c the question type).

However, the probability $P(Q|c)$ can easily be calculated using a language model (LM) trained on all questions of class c. The major advantage of the language modeling approach is that we can draw on a vast amount of available techniques to estimate and smooth probabilities even if there is very little training data available. On average, there are only about 100 training questions per question type.

Specifically, we evaluated absolute discounting, linear interpolation and Dirichlet prior as smoothing techniques. It turns out that absolute discounting in a variant known as Kneser-Ney smoothing for bigram language models gives best possible results. On top of this, we employ a count specific discounting technique.

The prior $P(c)$ can be considered a unigram language model as well. However, as all classes are seen sufficiently often (that is at least 4 times) there is no smoothing issue at all and relative frequencies can be used.

Algorithm	Accuracy
Naive Bayes	67.8%
Neural Network	68.8%
SNoW	75.8%
Decision Tree	77.0%
SVM	80.2%
LM	80.8%

Table 1: Comparison of various algorithms (Naive Bayes, ... SVM) investigated in (Zhang and Lee, 2003) with the proposed language model based approach, denoted by LM in the table.

Tables 1 compares results from (Zhang and Lee, 2003) with the proposed LM approach for various machine learning algorithms. As (Zhang and Lee, 2003) use two different feature sets (bag-of-words and bigram features) we always picked the better result from (Zhang and Lee, 2003). The LM approach uses bigram features.

It is interesting to see that the approach is much better than the Naive Bayes approach even though it uses the same independence assumptions. This difference is probably due to the very much different smoothing technique. The SVM is the best algorithm from (Zhang and Lee, 2003) however it is outperformed by the LM approach by a small margin.

3.3 Document Retrieval

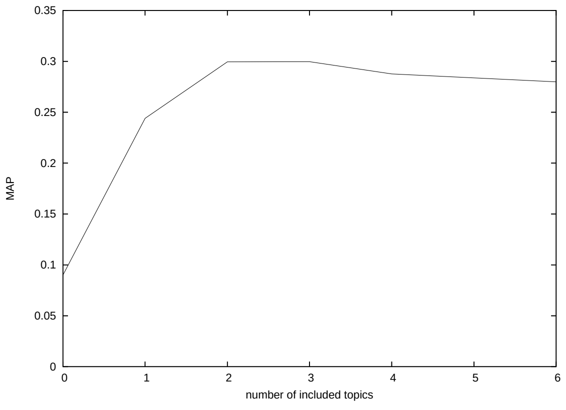

Figure 5: Number of included topics versus MAP for TREC 2004 data.

Document retrieval is a crucial part in Question Answering systems. In this part we tried to reduce the number of documents to a smaller, more manageable set of relevant documents. In our experiment we considered a two-step-approach necessary in order to meet the problem. First, a kind of query construction was made to optimize the search for documents. Then the document retrieval was done in a second step using standard language model based techniques.

3.3.1 Query construction

In this first step, we prepared the query to ideally fit the needs of our document retrieval algorithm. As explained in (Miller et al., 1999) each term in the query can be weighted with a score. This means that repeating a specific term multiple times would give the latter term more importance in form of a higher score. Because this holds also for our language model approach, we expanded the query in this step with the topic of the question. Figure 5 shows the effects on our document retrieval systems when we expand the query with the topic multiple times on TREC 2004 data. On the x-axis the number of included topics is shown whereas on the y-axis the *Mean Average Precision (MAP)* is plotted. The performance increases until including the topic three times. So, if the topic is added too often it gets too much weight and other possible important keywords are scored too lowly and, therefore, the retrieval system performs worse.

There is also just a marginal improvement between adding the topic twice or three times so we decided to include the topic only twice for our experimental setup.

3.3.2 Document retrieval and fetching

For document retrieval the *Lemur Toolkit for Language Modeling and Information Retrieval*[3] was used. Queries as well as the AQUAINT corpus were stemmed with the Porter stemmer and no stop-word removal was done. As mentioned above, we chose a language model based approach for the retrieval step using unigram distributions. The smoothing method we used was Bayesian smoothing with Dirichlet priors. As shown in (Hussain et al., 2006) smoothing with Dirichlet prior performs best in context of document retrieval experiments and even outperforms traditional information retrieval techniques like Okapi and TFIDF. (Hussain et al., 2006) also suggests an optimal smoothing parameter for TREC question sets which we used within our experiments as well.

After the retrieval step, we fetched the best 60 relevant documents because this number was sufficient in previous TREC runs to get about 90% of answers within those documents.

3.4 Sentence Retrieval

In this section, we describe the setup for retrieving sentences and the actual re-ranking step. Sentence retrieval, which is just a special case of passage retrieval, is a common step in question answering ((Clarke et al., 2000), (Tellex et al., 2003)). Normally it is necessary to further reduce the size of a document collection in order to improve finding answers. But there also might be very long documents within the collection or perhaps topic changes within a single document. So the document is further split up to smaller chunks to deal with such events. In our case we split up the documents to single sentences.

3.4.1 Experimental setup

Before we started with the sentence retrieval experiments some preparations had to be done. First of all we took the retrieved document collection from the previous section (3.3.2) and extracted those documents from a specially prepared AQUAINT corpus. In this prepared corpus, we used a sentence boundary detection algorithm[4] to identify possible ends of sentences.

[3]http://www.lemurproject.org/

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

After the extraction, the sentences as well as the queries were stemmed using the Porter stemmer. Parallel to this stemming process we did a kind of expansion of queries and sentences. If the expected answer type of a query[5] was *DATE* then the token *DATE* was added at the end. The sentences were prepared in almost the same manner. Here, patterns were used to identify possible occurrences of time and date information. Due to this expansion, possible answer candidates for the expected answer type *DATE* were ranked higher. To get an optimal score for those kind of queries we introduced a weighting scheme and experimentally determined the specific weight.

Again an unigram language model based technique was used to rank the sentences in our experiment. In detail we used Bayesian smoothing with Dirichlet prior which is given by the following formula:

$$p_\mu(w|d) = \frac{c(w;d) + \mu p(w|C)}{\sum_w c(w;d) + \mu} \qquad (2)$$

Whereas *c(w;d)* means the count of word *w* in sentence *d*, *C* is the collection of sentences and μ is the smoothing parameter.

We chose this kind of smoothing because it performed already promising for document retrieval. As smoothing parameter we chose the interpolation weight $\mu = 100$. We got this value by a search for the complete parameter space. Figure 6 shows that this method actually performed better than Jelinek-Mercer linear interpolation.[6]

For the re-ranking experiments of the sentences the `LSVLM` toolkit was used. It is the *Language Modeling toolkit from LSV*[7] and implements standard language modeling techniques. We decided to use this particular toolkit instead of *Lemur* because it is much more flexible. So it is easily possible to switch between different language models for interpolation or to manipulate the used vocabulary. In our case, we closed the vocabulary over the query to get a better performance as described in (Hussain et al., 2006).

Another preparation step was to include a dynamical list of stop-words. This list consists of the four most commonly used terms of the complete sentence collection. However, these words were not removed but just got a smaller score. Again a

[5]see Section 3.2

[6](Hussain et al., 2006) also showed that Jelinek-Mercer performed better than absolute discounting

[7]Lehrstuhl für Sprachsignalverarbeitung

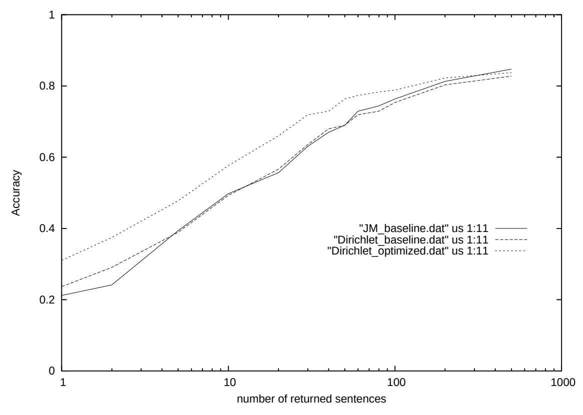

Figure 6: Number of returned sentences versus accuracy of baseline and optimized experiments for TREC 2004 data.

weighting scheme was used to optimal score the stop-words.

Finally, the queries were optimized. This was done by removing the query word. In most cases the query word has no meaning in terms of searching for relevant sentences so removing it gives a higher score for the rest of the possible relevant query words.

3.4.2 Sentence Re-ranking

The settings described in the previous section were used to do the sentence re-ranking.

Figure 6 shows first results when testing different language model based techniques with different optimization parameter explained in the previous section on TREC 2004 data. The plot shows on the x-axis the number of sentences returned on a logarithmic scale and on the y-axis the accuracy of the system.

We experimented with two baseline methods, the *Jelinek-Mercer* linear interpolation (`JM_baseline`) and a baseline of the Dirichlet prior smoothing (`Dirichlet_baseline`). The figure show that both distributions perform nearly equally but the Dirichlet baseline is slightly better when returning just a smaller set of sentences. The third smoothing method (`Dirichlet_optimized`) shows the language model with optimization changes we described in section 3.4.1. It outperforms the baseline distributions for nearly all cases. The interesting point in regard to our experiments is that we had to go back to just a smaller number of sentences to obtain the same accuracy as for a baseline method.

Table 2 also shows the significant improvement of our smoothing method regarding the *Mean Reciprocal Rank (MRR)* of the distributions for TREC 2004 dataset. A performance gain of 25% can be observed between the Dirichlet prior baseline and the optimized language model and a gain of actually more than 34% regarding the Jelinek-Mercer linear interpolation.

Distribution	MRR
JM_baseline	0.29
Dirichlet_baseline	0.31
Dirichlet_optimized	0.39

Table 2: MRR of baseline and optimized experiments.

3.5 Answer Extraction

For the sentences retrieved by SR Module, we first process them by using linguistic tools including LingPipe (http://www.alias-i.com/lingpipe/) for named entity recognition, Abney's chunker (Abney, 1989) for NP chunking and MINIPAR (Lin, 1994) for dependency parsing. Next, we apply two strategies, which are mainly based on surface text pattern matching and correlation of dependency relation path respectively, to extract exact answers from the processed sentences. We set them as pipeline structure. For each sentence, firstly, answer patterns (Section 3.5.1) are used to match candidate answers. If matched, the candidate answers will be returned. Otherwise, path correlation-based Maximum Entropy model (Section 3.5.2) is applied to rank the candidate answers.

3.5.1 Chunk-based Surface Text Pattern Matching

For the questions in question classes, answer patterns (APTN) indicate what are expected answer positions in surface sentences. APTNs are represented as regular expressions over tokens, using the variables ***slot***, ***var*** and ***ANSWER***. ***slot*** is bound to the key chunks of questions. A question chunk, expected by certain slots, is assigned in question pattern matching. ***var*** is a set of special alternative words, which are usually shared by various patterns and also assigned in question pattern matching. ***ANSWER*** indicates the expected answer. Figure 7 shows APTNs for question class *WHAT_CREATION*. Patterns are manually authored for the system. However, results show that the coverage is not satisfactory since only 12 questions are answered by pattern matching.

```
<CLASS key="WHAT_CREATION">
<APTN_SET>
   <APTN>slot0 (var0) ANSWER</APTN>
   <APTN>ANSWER be (var0) by slot0</APTN>
   <APTN>slot0 be (var2) of ANSWER</APTN>
   <APTN>slot0 's (var1)( of) ANSWER</APTN>
   <APTN>ANSWER be (var1) (of|by) slot0</APTN>
   <APTN>ANSWER be slot0 's (var1)</AP>
   <APTN>ANSWER (who|that|which) be (var0) by slot0</APTN>
   <APTN>ANSWER (who|that|which) be (var1) (of|by) slot0</APTN>
   <APTN>after (var0) ANSWER( \\w+){1,5}, slot0  </APTN>
   <APTN>(var1) of ANSWER by slot0</APTN>
   <APTN>slot0 's (var1) of ANSWER</APTN>
   <APTN>be (var2) of ANSWER(,)? slot0</APTN>
   <APTN>slot0( ,)? who (var0) ANSWER</APTN>
   <APTN>ANSWER 's( first)? (var0) slot0</APTN>
   <APTN>slot0( (,|-))? (var2) of ANSWER</APTN>
   <APTN>slot0( (,|-))? ANSWER 's (var2)</APTN>
   <APTN>(var1) of slot0( (,|-))? ANSWER </APTN>
   <APTN>slot0 's (var1)( (,|-))? ANSWER</APTN>
   <APTN>ANSWER (var2)( ,)? slot0</APTN>
   <APTN>ANSWER( (,|-)) (var1) (by|of) slot0</APTN>
   <APTN>ANSWER( (,|-)) ANSWER 's slot0</APTN>
</APTN_SET>
</CLASS>
```

Figure 7: Example of answer patterns for class *WHAT_CREATION*.

3.5.2 Correlation of Dependency Relation Path

If none of candidate answers are matched to APTNs, we further compare dependency relations between candidate answers and mapped question chunks in sentences with corresponding relations in questions. A correlation measure is proposed to calculate distance between two dependency relation paths. The parameters of the measure are estimated on a set of question answer pairs in previous TREC. Next, the correlations are incorporated in a Maximum Entropy-based ranking model which estimates path weights from training. Lastly, top-ranked candidate answer in each sentence is returned. (Shen and Klakow, 2006) presents the detailed information of the model. Results show that it extracts 65 among 403 factoid questions in TREC 2006.

3.6 Fusion

One issue with fusion of different processing streams and different modules in a QA system is that scores are calculated in various ways and in particular that they are hardly ever probabilities. This makes fusion a difficult task.

However, it is well known that probability distributions in natural language applications (but also beyond this) are often Zipf-distributed. As all the modules produce a ranking on one way or other, we use Zipf's Law to convert ranks into probabilities. A simple version of this idea is an arithmetic average of inverse ranks which was proposed in (Whittaker et al., 2005).

Here we use for the probability of an answer created by module i being correct a Mandelbrot distribution which is a refinement of the Zipf distribution. It has one more parameter that allows to tune the flatness of the distribution for top ranked answers. This is important where modules put several good answers on top but can not discriminate between the alternatives. The Mandelbrot function is defined by

$$P_i(A) = \frac{\mathcal{N}}{(r_i(A) + \mu)^\beta} \tag{3}$$

where $r_i(A)$ is the rank on which module i puts the answer A and μ and γ are parameters determine on the TREC 2004 data. $\mathcal{N}$ is a normalization factor depending on μ and γ.

The actual fusion is a linear interpolation

$$P(A) = \sum_{i=1}^{N} \lambda_i P_i(A) \tag{4}$$

where N is the number of components to be fused and λ_i are the interpolation weights satisfying the normalization constraint $1 = \sum_{i=1}^{N} \lambda_i$.

All the parameters are optimized on the TREC 2004 data. We fuse the results of our answer extraction with the sentence retrieval results, the Wikipedia system and the output of the web validation module. Note that the web validation modules was not used.

4 Results

We carried out our TREC experiments on three nodes part of a Linux Beowulf cluster with 2.6 GHz Intel Xeon multi-core CPUs (512 MB RAM each). While the cluster is equipped with a master node featuring a 1 TB RAID system, we only used the nodel-local 300 GB hard disk drives to avoid delays caused by NFS overhead.

Table 3 shows the three runs we submitted and our results obtained. The first run uses only the AQUAINT corpus to retrieve answer candidates from, whereas the second run adds Wikipedia candidates that are used to validate AQUAINT candidates. The third run uses more answer extraction patterns (in addition to the ones already used in the other runs).

Run ID	AQUAINT stream	WIKIPEDIA stream	More manual patterns	Accuracy FACTOID	F-score LIST	F-score OTHER
`lsv2006a`	+			0.174	0.077	0.107
`lsv2006b`	+	+		**0.191**	0.096	0.109
`lsv2006c`	+	+	+	0.186	**0.098**	**0.110**

Table 3: LSV Group Runs and Results Submitted to TREC 2006.

As the numbers show, despite the fact that *Alyssa*was implemented in just a few months prior and during TREC, it already performed better than the median of participants. We attribute this success to

- the flexibility of the system architecture, which allowed us to experiment with a variety of approaches systematically (while the system implementation was still in progress), combined with
- our data-driven approach: at any point in time we try to improve the component that was the bottleneck (in terms of most errors introduced).

Surprisingly, our LIST and OTHER performance is *not* dramatically worse that the average despite no specific efforts were made to address these types, which indicates that the state of the art is still very basic. This may be attributed to the fact that we used Wikipedia rather than AQUAINT as the source for definition questions.

Unlike other groups, we did not observe an increase of unsupported answers. The lesson we learn from this is to use Wikipedia, but to use it wisely, i.e. to support AQUAINT answers rather than to extract candidate answers directly from it.

5 Summary and Conclusions

5.1 Summary

We have presented the new open-domain Q&A system *Alyssa* and our experiments for the TREC 2006 question answering track using the system to explore a statistically-inspired approach to question answering.

Using a flexible software architecture that allowed to switch methods easily we carried out a series of experiments. Our best results outperformed the median over all TREC 2006 Q&A track participants.

5.2 Future Work

In future work, we are planning to use the existing system to carry out further experiments, such as studying the impact of the substitution of pronouns by their antecedents as a recall-enhancing device.

We are also planning to improve our system along four directions: First, we would like to develop and integrate more fine-grained named entity tagging. Second, we are planning to investigate more sophisticated learning methods for the integration of the evidence, both at the fusion step and the component level. Third, making use of existing patterns from the answer extraction could be used for precise Web validation patterns. Finally, we would like to carry out an ablation study analyzing the errors made by the systems.

Acknowledgements

We thank Ciprian Raileanu, Alexandru Chitea and Andrea Heyl for implementing the modules for Web validation (using the Google Web Service), anaphora resolution module (using GATE's ANNIE) and pattern construction respectively.
Jochen Leidner was funded in part by a SOCRATES grant from the European Union and in part by the BMBF project SmartWeb. Dan Shen was funded by the German research council DFG through the International Graduate College "IGK" between Saarland University and the University of Edinburgh.
Andreas Merkel was funded by the BMBF project SmartWeb.
We thank to Johan Bos and Sebastian Padó for discussions.

References

Steven Abney. 1989. Parsing by chunks. *The MIT Parsing Volume.*

C. Clarke, G. Cormack, D. Kisman, and T. Lynam. 2000. Question answering by passage selection (MultiText experiments for TREC 9.

Hamish Cunningham. 2000. *Software Architecture for Language Engineering.* Ph.D. thesis, Deptartment of Computer Science, University of Sheffield, Sheffield, England, UK.

Munawar Hussain, Andreas Merkel, and Dietrich Klakow. 2006. Dedicated backing-off distributions for language model based passage retrieval. In *Hildesheimer Informatik-Berichte, LWA 2006.*

M. Kaisser and T. Becker. 2004. Question answering by searching large corpora with linguistic methods. In *Proceedings of the Text Retrieval Conference (TREC 2004)*. NIST.

Jochen L. Leidner, Johan Bos, Tiphaine Dalmas, James R. Curran, Stephen Clark, Colin J. Bannard, Mark Steedman, and Bonnie Webber. 2004. The QED open-domain answer retrieval system for TREC 2003. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*, NIST Special Publication 500-255, pages 595–599, Gaithersburg, MD, USA.

Jochen L. Leidner. 2003. Current issues in software engineering for natural language processing. In *Proceedings of the Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS) held at HLT/NAACL 2003*, pages 45–50, Edmonton, Alberta, Canada.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING*.

Dekang Lin. 1994. PRINCIPAR—an efficient, broad-coverage, principle-based parser. In *Proceedings of COLING*, pages 42–488.

David R. Miller, Tim Leek, and Richard M. Schwartz. 1999. A hidden markov model information retrieval system. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 214–221, Berkeley, CA, USA.

Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of ACL2006*.

S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering.

E. Whittaker, P. Chatain, S. Furui, and D. Klakow. 2005. TREC 2005 question answering experiments at Tokyo Institute of Technology. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*.

M. Wu, M. Y. Duan, S. Shaikh, S. Small, and T. Strzalkowski. 2005. University of Albany's ILQUA in TREC 2005. In *Proceedings of the Text Retrieval Conference (TREC 2005)*. NIST.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *SIGIR '03: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 26–32.