The *Alyssa* System at TAC QA 2008

Michael Wiegand Saeedeh Momtazi Stefan Kazalski
Fang Xu Grzegorz Chrupała Dietrich Klakow
Spoken Language Systems
Saarland University
D-66125 Saarbrücken, Germany
lsv_trec_qa@lsv.uni-saarland.de

Abstract

We present the Alyssa QA system which participated in the TAC 2008 Question Answering Track. The system consists of two parallel streams: the blogger stream which is used in order to deal with questions which ask for lists of blog authors, and the main stream which processes other opinion questions. We also use a named entity detection component specialized to the entertainment domain. Evaluation results show that our system exhibits systematically better performance on blogger questions than on other rigid questions.

1 Introduction

In this paper we describe the *Alyssa* QA system developed at Saarland University. We focus on the modifications made to our system in order to participate in the TAC 2008 Question Answering Track.

The principal challenge in this year's competition has been answering opinion questions based on the corpus of blog posts (Blog06). Since our system so far has been mostly aimed at answering standard factoid questions, we needed to implement substantial additional components in order to be able to deal with the new setting.

Given the importance of detecting and classifying opinion questions, we experimented with several question polarity detection methods and as a result chose a robust rule-based approach which performed well on the TAC sample questions.

We noticed that many of the TAC sample questions need to list bloggers having a certain opinion about a certain topic, and thus we developed a whole separate stream devoted to detecting blogger questions and answering them.

Based on inspection of the sample questions and the Blog06 corpus, we suspected that detecting named entities from the entertainment domain would again prove useful to this year's task, and accordingly we gathered new resources and reengineered the dedicated component for detecting such entities from last year's system. The complete system thus consists of two parallel processing streams: the main stream answers standard opinion questions, while the blogger stream is specialized to dealing with questions asking for lists of blogger names.

The remainder of the paper is organized as follows: in Section 2 we present an overview of the *Alyssa 2008* system. In Section 3 we present the components developed or modified for this year's competition: question polarity classification (Section 3.1), query expansion (Section 3.2), named entity detection for the entertainment domain (Section 3.3), squishy list answer extraction 3.4), answer validation (Section 3.5), the blogger stream (Section 3.6) and fusion (Section 3.7). In Section 4 we describe the configuration for the three submitted Alyssa runs and present the evaluation results for the complete task as well as for the subset consisting of blogger questions. Finally in Section 5 we conclude and discuss future work.

2 System Overview

For the TAC 2008 Question Answering Track, our system Alyssa – which successfully participated in last years' competitions – has changed its focus from factoid questions to opinion questions. Figure 1 shows the architecture of Alyssa. For this year, we have defined two streams in our system. The first stream is an adapted version of our factoid stream from last year (Shen et al., 2007) and the second stream is a completely new stream

which has been designed for the questions asking for bloggers. A rule-based component, blogger question detection, classifies the questions into two types. The questions asking for bloggers run through both the main stream and the blogger stream whereas the other questions only run through the main stream.

In the main stream, we first perform a linguistic analysis of the question. This encompasses syntactic parsing and named entity tagging. This information is used later for answer extraction. The semantic type of a question is determined in a separate step called semantic question typing. This year we replace our previous classification module using language modeling (LM) by a model using support vector machines (SVM). We decided in favor of SVM since they produced a higher classification accuracy both on the sample questions provided by NIST for this year's TAC competition and our own set of opinion questions.

Beside the semantic question typing, the polarity of opinion questions is determined by the polarity question typing component. Then, a query is constructed from the question, based on this analysis.

Following query construction, query expansion techniques based on *Google* and *Wikipedia* are applied. The expanded query is run against document retrieval on the Blog06 corpus. A form of dynamic document fetching (Shen et al., 2007) is used to determine the number of retrieved documents according to the question type. The sentence retrieval component retrieves the relevant sentences based on language modeling.

In the next step, a new opinion sentence retrieval module extracts opinionated sentences from the retrieved sentences. Sentence polarity classification is applied to the retrieved opinionated sentences in order to classify the sentences into positive and negative sentences. Which of these groups is selected for further processing depends on the result of polarity question typing. The next step is answer extraction for squishy list questions.

There are two types of linguistic processing which may be applied to a rigid list question. If the question asks for a named entity from the entertainment domain, we automatically annotate the retrieved documents with named entities of the appropriate type. Otherwise, only the opinionated retrieved sentences with the correct polarity are annotated. The reason for document-level annotation is that for entertainment-related rigid questions answer extraction on sentence level is not feasible (see Section 3.3 for more details).

After the extraction of candidate answers from the annotated documents or sentences, duplication removal is applied. Our new answer validation component re-ranks the resulting list of unique candidate answers as the final answers to the rigid list questions.

The blogger stream of Alyssa begins after document fetching of the main stream. In the blogger stream the retrieved documents undergo blogger detection to split the document into smaller segments and find the author/blogger of each segment. Each segment is assigned three scores determined by three different components: topic relevance ranking searches for the relevant segments to the question, opinion classification computes the degree of opinionatedness, and polarity classification measures how much the polarity of a segment overlaps with the polarity of its question. A final score obtained by interpolating the scores of the individual components is assigned to each segment. After that, the segments are ranked in the blogger ranking according to that score. In the fusion module, the result of blogger questions is merged with the output of the main stream which creates a unique list for blogger rigid list questions.

3 New Experiments

3.1 Question Polarity Classification

The task of this module is decide whether a TAC question has a positive or negative polarity. Using the sample questions for this year's TAC QA competition[1], we tested three types of classifiers, one trained on the MPQA corpus (Wiebe et al., 2003), one trained on the English NTCIR data (Seki et al., 2007), and one rule-based component using the Subjectivity Lexicon (SL) by (Wilson et al., 2005). MPQA and NTCIR were chosen as these corpora contain annotation usable for sentence-level classification. SL was chosen as a polarity lexicon since it is one of the largest manually constructed lexicons currently available in English. In addition to the prior polarity of a lexical unit it contains the strength of polarity[2] and part-of-speech informa-

[1] `http://www.nist.gov/tac/tracks/2008/qa/QA08_sample_questions.xml.txt`

[2] either *weak* or *strong*

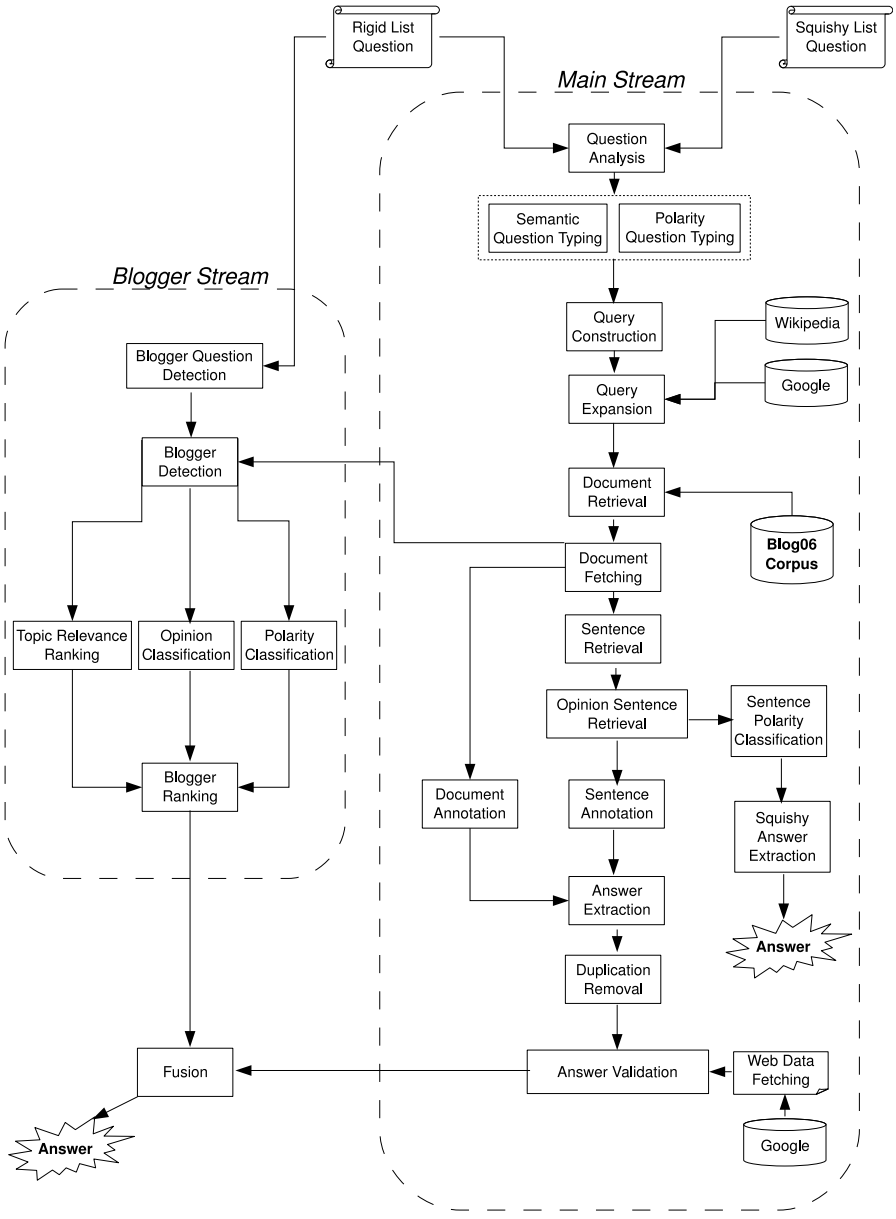

Figure 1: Architecture of *Alyssa*.

tion which may serve as basic word sense disambiguation. We made use of all this information in our rule-based classifier.

From the MPQA corpus, we extracted all sentences with *direct subjective elements* and *expressive subjective elements* (Wiebe et al., 2003) having either positive or negative polarity. If a sentence contained both positive and negative polarity according to these elements, we manually annotated the sentence according to the overall polarity[3]. We trained two SVM-based classifiers on this dataset: one trained on overall vocabulary of the dataset, the other trained on all polarity expressions from SL. Both classifiers employ stemming. From the NTCIR corpus, we extracted all polar sentences where at least two of the three judgments agreed in labeling. Again we trained two classifiers on this dataset analogous to the former dataset.

The algorithm of the rule-based (unsupervised) classifier is as follows:

1. **Title Removal:** Discard all words within quotations[4].

2. **Normalization:** Stem the question[5] and normalize case.

3. **Feature Extraction with Crude Word Sense Disambiguation:** Look up all words of the question in the lexicon. A word must not only match the lexical form but also the part-of-speech tag of an entry.

4. **Negation Modeling:** Scan the question for negation expressions. In case a negation expression is found, reverse the polarity of all subsequent polar expressions.

5. **Score Assignment:** Assign scores to each polar expression. Assign 0.5 to weak polar expressions and 1.0 to strong polar expressions.

6. **Classification:** Sum the scores for each polarity. Classify the question as *positive*, if the sum of scores of the positive expressions is larger than the sum of scores of negative expressions. Otherwise classify the question as *negative*.

The current version of the rule-based classifier has a bias towards negative polarity. This is due to the fact that negative polarity is more difficult to detect[6].

Table 1 displays the performance of the different approaches on the sample questions for this year's TAC QA competition. Clearly, the data-driven methods perform very poorly. Slight improvement is achieved by only using polar expressions as features. We found that this is more effective than just removing question-specific information from the test data, such as interrogative pronouns and question marks. We suspect that the poor performance of the data-driven methods is due to a domain mismatch. The rule-based method is significantly better than any other approach which is why we used this method for our QA system.

3.2 Query Expansion

In this year's system we replaced our existing retrieval component with a component more suitable for opinion retrieval. Whereas our old retrieval system merely considered terms from the question and the target, the new retrieval includes a relevance-based query expansion using the web and a set of hand-selected opinion markers. We also improved the sentence boundary detection by adding heuristics for processing ungrammatical sentences contained in blogs.

The query construction happens in three stages:

1. **Extraction of Query Term Seeds:** Noun and verb phrases contained in the question are identified by using Brill's part of speech tagger (Brill, 1992) and Abney's chunk parser (Abney, 1991). The target and the phrases extracted from the question are considered the set of query term seeds.

2. **Retrieval of Feedback Terms:** Each query seed term is sent to different web search engines[7] and Wikipedia to create a set of feedback documents. Feedback terms are extracted by applying *relevance-based language models* (Lavrenko and Croft, 2001) on the feedback documents.

[3] Sentences with mixed overall polarity, such as *Peter likes the book while Mary hates it*, were discarded.

[4] Consider question *Why did people dislike the movie "Good Night and Good Luck"?* where taking the occurrences of *Good* into account might cause the question to be wrongly classified as positive.

[5] All entries in SL were stemmed as well.

[6] even with the inclusion of negation modeling

[7] We used Google, MSN Live Search and Yahoo.

Corpus	Method	Feature Set	Accuracy
MPQA	data-driven	in-domain vocabulary	57.58
MPQA	data-driven	Subjectivity Lexicon	60.61
NTCIR	data-driven	in-domain vocabulary	54.55
NTCIR	data-driven	Subjectivity Lexicon	63.64
—	rule-based	Subjectivity Lexicon	**93.94**

Table 1: Performance of Different Question Polarity Classifiers on TAC Sample Questions.

Queries	Relevance		Opinion	
	MAP	P@10	MAP	P@10
Old Retrieval – System of 2007				
BLOG06	0.2737	0.720	0.1782	0.4540
BLOG07	0.2984	0.7000	0.2214	0.4480
New Retrieval – System of 2008				
BLOG06	0.3668	0.7640	0.2490	0.5260
BLOG07	0.4292	0.7620	0.3225	0.5380

Table 2: Performance of Retrieval on TREC Blog Topics.

3. **Query Expansion:** The final set of query terms is the union of query seed terms, feedback terms and a pre-defined set of opinion markers.

To further improve the performance of the retrieval we changed our retrieval engine from Lemur[8] to Indri[9]. The inference network allowed us to flexibly combine the phrases of the query and the expansion terms with weights and connectors. This makes the retrieval more robust against noisy expansion terms.

3.3 Named Entity Detection for the Entertainment Domain

We speculated that fine grained named entities from the entertainment domain (e.g. *actor names*, *book names*, *song titles*) would be highly relevant for this year's opinion task since a great proportion of opinionated content in blogs covers items from this domain. For virtually all of these types, we are totally reliant on look-up lists since alternative methods to detect these entities, for example by using linguistic cues are not effective. We extended last year's typologies by a few categories. The current list of types along the corresponding web sites from which we extracted these types is displayed on Table 3.

[8] www.lemurproject.org

[9] www.lemurproject.org/indri

A major problem of the newly extracted entity lists is the large amount of noise. In particular, the book list contains a fairly high proportion of partial entries, entire abstracts rather than titles, or just spurious entries. Due to time restrictions we resorted to heuristic measures. We excluded one-word entries since they fired far too often. In order to remove highly ambiguous entries (e.g. book titles which are also person names or location names etc.), we used gazetteers from common named entity types (dates, person names and locations) as a filter. The problem of noise is far from solved. We assume that a more sophisticated treatment of named entities would increase the overall performance of our QA system.

The strategy of retrieving answers for entertainment-related list questions was kept fairly simple. We noted that named entities from the entertainment domain are very sparsely distributed among Blog06 documents. Since our sentence retrieval hardly retrieved any sentences with these types, we extracted entities from retrieved documents. We did not factor in any sentiment aspect into the retrieval of these entities as the restrictiveness of potential polarity modeling would have badly penalized the recall of retrieval. We noted that there is usually a strong bias towards a specific polarity given a particular answer entity. Our attempts of classifying polarity are obviously too unstable to remove this bias while preserving an acceptable recall. For example, a question like *List unpopular movies starring Tom Hanks* is very difficult to answer, since the overwhelming majority of movie titles starring the afore-mentioned actor found in the web are popular movie titles. We found that for questions like the one above mentioned answering the mere factoid question, i.e. *List movies starring Tom Hanks*, is more likely to produce correct answers than trying to additionally postulate a negative polar context. Note that the opposite polar question, i.e. *List popular movies by Tom Hanks*

Named Entity Type	Source
actors/actresses	IMDB
authors	IMDB
books	Abebooks.com
musical artists	Discogs.com
movie titles	IMDB
song names	Discogs.com
tv programs	IMDB

Table 3: Named Entity Types from the Entertainment Domain with their Web Sources.

would not require any opinion modeling, since in this case topic relevance strongly correlates with polarity relevance.

3.4 Squishy List Answer Extraction

For this year's squishy list questions we experimented with two different models, one standard model using sentence retrieval (Section 3.4.1) and a refined model using passage retrieval (Section 3.4.2).

3.4.1 Model I

The standard squishy list model uses the factoid sentence retrieval from our last year's system (Shen et al., 2007) to extract topic relevant sentences. Afterwards we remove all sentences which do not have the polarity of the question as classified by our polarity question typing (Section 3.1).

The polarity filtering of the retrieved sentences happens in two stages:

In the first stage all opinionated sentences are extracted. Opinions are detected by a classifier trained to distinguish between opinionated and factual content. For this, we use an SVM classifier trained on Wikipedia topics (for factual content) and Rate-It-All[10] reviews (for opinionated content). Rate-It-All seemed a preferable source since it covers various domains, in particular those which are relevant for previous TREC Blog Track and TREC QA Track topics (i.e. politics, entertainment, IT-products, and sports). For SVM we employed χ^2 feature selection (Yang and Pederson, 1997).

For polarity classification we built another classifier from a subset of the Rate-It-All data. Unlike the question polarity classification (Section 3.1), we deliberately decided against a rule-based classifier in this task, since we observed that current polarity lexicons (which are the backbone of such a classifier) have a poor coverage on blog data. Since we did not have sufficient time to build a lexicon for the blog domain, we decided to use a data-driven classifier trained on labeled in-domain data. We only considered reviews with only one sentence because larger reviews tend to have mixed opinions and serve less well for sentence-level polarity classification.

[10] `http://www.rateitall.com`

3.4.2 Model II

Squishy list questions ask for the reasons people like or dislike something. In most cases the reason for the sentiment is not located within the statement itself expressing the sentiment towards a particular target but within some nearby sentence. Our alternative squishy list question model tries to account for this behavior.

The answer extraction for squishy questions in our alternative model happens in two stages: First, relevant segments with the correct polarity are extracted with the segment retrieval components used in segment ranking (Section 3.6.3). The text segments are created by applying a form of *text tiling* (Hearst, 1997). Then, the most relevant sentence and the nearby sentences are extracted. The detection of the relevant sentences is done using word frequencies of the query words. The ranking of the result corresponds to the ranking of the segments.

3.5 Answer Validation

Answer validation focuses on using the redundancy of web data to further validate top-ranked answer candidates returned by answer extraction. The hypothesis for web validation is that the number of documents that can be retrieved from the web in which the question and answer co-occur is a significant clue to the validity of the answer (Magnini et al., 2002b). Our answer validation component consists of the following steps:

1. Given a question, construct three different kinds of queries: bag of words (BOW), phrase chunk (CHK) and declarative form (DEC) (Shen et al., 2007).

2. Remove duplicates and noisy data from the results of answer extraction.

3. Combine each query with an answer candidate.

Question Type	Input	Snippet	Output
Non-Blogger Rigid	15	20	15
Blogger Rigid	15	15	15
Squishy	15	15	15

Table 4: Parameter Settings for Answer Validation.

4. Submit the pair to the Google search engine.[11]

5. Estimate the *validity score (VS)* from the results returned by the search engine.

Last year, we only used the frequency of the candidates within the web data for the validity score for answer selection. This year, we experimented with more complex validity metrics to select the N-best answers for list questions. We employed both Maximal Likelihood Ratio (MLHR) (Dunning, 1993) and Co-occurrence Weight (COW) (Magnini et al., 2002a). MLHR makes use of the asymptotic distribution of the generalized likelihood ratio, which allows comparisons to be made between the significance of the occurrences of both rare and common textual phenomena. This metric is practical for calculating word co-occurrence from sparse data resulting from complex queries. COW measures the association strength by the distance between a candidate answer and keywords, and considers more contextual information in each snippet.

The final validity score (VS) combining MLHR and COW is calculated by the following formula:

$$VS = COW^{\alpha} \cdot MLHR^{\beta} \quad (1)$$

For TAC list questions, we set $\alpha = 3$ and $\beta = \frac{1}{2}$ based on experiments on TREC2006 list questions (89 questions). In the new answer validation component, we can set three parameters: number of input answer candidates (Input), number of snippets returned by Google (Snippet) and number of output answers (Output). The parameter settings are displayed in Table 4.

3.6 Blogger Stream

After inspecting the sample questions for this year's TAC QA competition, we noticed a recurring question type among the *rigid list questions*, which we thought required separate processing. These are opinion holder questions, i.e. questions asking for persons who express an opinion concerning a particular topic. Most of them do not make any restriction towards the opinion holder, such as *What people have good opinions of Sean Hannity?* We found that the highest proportion of correct answers to these questions are *blogger names*.

Conventional approaches to extract opinion holders from natural language text, i.e. either by very shallow contextual features or linguistic features encoding relations between opinion bearing words and opinion holders, such as those used in (Choi et al., 2005; Kim and Hovy, 2006), will not work for blogger detection. This is due to the fact bloggers are usually not mentioned within a blog post or comment but either precede or follow them. The remoteness of blogger names from the topic relevant terms within blog posts and comments also means that sentence retrieval is an inappropriate input for blogger detection. We therefore came up with a rule-based solution which takes as input entire blog documents, heuristically segments them and assigns blogger names to the resulting segments.

3.6.1 Blogger Question Detection

The task of this module is to determine whether a question asks for a blogger or not. We use a rule-based classifier based on a small set of regular expressions. If an input question is classified as a blogger question according to the rule-based classifier the question is processed by the main stream and the blogger stream. All other questions are only processed by the main stream.

3.6.2 Blogger Detection

The blogger detection component attempts to segment the retrieved blog documents and assign to each segment its author. A typical blog document consists of a post by the blog owner, plus a number of comments by the readers. Both the blog owner and the commentators can sign their contributions with their real names or nicknames. In order to segment documents and find authors for the segments we exploit two classes of commonly occurring patterns used to indicate authorship in many blogs. The *text-then-author* pattern is a chunk of text followed by an expression indicating the author, such as for example "TEXT posted by AUTHOR on DATE". In the *author-then-text* variation the author name is followed by their con-

[11] The priority of query types is: DEC $\succ$ CHK $\succ$ BOW

tribution, e.g. as in "AUTHOR said: TEXT".

If none of the patterns from the above two classes matches, we assign to the whole blog document the default author string, which we try to find in the document body, header or URL using another set of patterns.

3.6.3 Blogger Ranking

In this component we combine three scores for the retrieved segments in order to rank them. The scores are topic relevance (s_1), opinionatedness score (s_2), and polarity score (s_3).

The topic relevance is computed by using *relevance-based language models* (Lavrenko and Croft, 2001) (RLM). We decided in favor of RLM in this ranking task, since standard language modeling performed worse on this task using the TAC sample questions. In our initial experiments on these data we also found that *conditional sampling* outperformed *i.i.d. sampling* which is why we used it for the official evaluation.

The opinion detection we used for this module is identical with the one used in Model I of the squishy list answer extraction (Section 3.4.1).

Polarity was classified by another SVM-based classifier. We extracted those data from labeled Rate-It-All reviews. Unlike the polarity classification within the basic model of squishy list answer extraction (Section 3.4.1), we did not confine ourself to reviews being only one sentence but considered any possible length. The output value of the SVM classifier is normalized in such a way that it is high if the polarity of the segment agrees with the polarity of the question and low otherwise.

The output of each subcomponent was used to rank the blog segments. In order to come up with scores from the subcomponents which can be reasonably combined, we used inverted ranks as the individual scores: s_1, s_2, and s_3. The final score s is a simple linear interpolation:

$$s = \lambda_1 s_1 + \lambda_2 s_2 + (1 - \lambda_1 - \lambda_2) s_3. \quad (2)$$

We tried a few different weight combinations on the TAC 2008 sample questions and found that setting each λ uniformly to $\frac{1}{3}$ gave one of the best results, and thus we used this configuration for the submitted runs.

3.7 Fusion

For some blogger and entertainment-related list questions from the sample TAC question set, we retrieved too few answer entities from the pertaining modules which exclusively retrieve correct answer types. In order to be able to always provide a fixed number of answer entities for each list question we add entities retrieved from our factoid answer extraction in case the entity type specific modules retrieved too little output.

4 Results

We carried out our TREC experiments on three nodes part of a Linux Beowulf cluster. The performance (F-score) of these three runs we submitted are shown in Table 5.

The configuration variations of our runs are as follows: the first run uses a new version of answer validation (Section 3.5). The squishy answer generation in this run is based on text tiling, i.e. Model II (Section 3.4.2).

In the second run, the squishy answer generation is based on standard sentence retrieval with opinion and polarity classification, i.e. Model I (Section 3.4.1). In the second run we also used an alternative version of polarity question typing (we changed the default polarity for ties[12]). The answer validation is the same as the one used at TREC 2007 (Shen et al., 2007).

The third run uses an alternative version of named entity recognition (we used a less aggressive filter for the look-up dictionaries); otherwise, the settings are as in the second run.

As the numbers show, there is no considerable difference between the three runs. The median average for rigid list questions computed over 17 runs is 0.063; and the median average for squishy list questions computed over the same number of runs is 0.091. These results show that in rigid list task, our performance is significantly better than median; but in squishy list task, we are in the median.

Run ID	F-score Rigid List	F-score Squishy List	F-score All
Alyssa1	0.097	0.087	0.094
Alyssa2	0.103	0.091	0.102
Alyssa3	0.106	0.090	0.104

Table 5: LSV Group Runs and Results Submitted to TREC 2008.

[12] I.e. if the scores for positive and negative polarity are equal, we consider the polarity of the question to be *positive* instead of *negative* (Step 6 in Section 3.1).

Since we have developed a new stream in our system, i.e. the blogger stream (Section 3.6), we are very interested in its impact on our results. With the current version of our system, the blogger questions are processed by the blogger stream and the non-blogger questions are processed by the main stream. The blogger questions only use the main stream as a back-off. For evaluating this new stream, we computed the F-score exclusively for the blogger questions. Our classification of the rigid list question is based on the output of the blogger question detection. According to this module, there are 56 blogger questions and 34 non-blogger questions among 90 rigid questions[13]. Table 6 shows the results of the three runs on blogger questions and non-blogger questions. The numbers show that on every run the performance on blogger questions is significantly better than on the remaining rigid list questions.

Run ID	F-score Rigid List Blogger	F-score Rigid List Non-Blogger	F-score Rigid List All
Alyssa1	0.126	0.049	0.097
Alyssa2	0.134	0.051	0.103
Alyssa3	0.144	0.044	0.106

Table 6: LSV Group Results on Blogger and Non-Blogger Questions.

5 Conclusion & Future Work

We have presented our modified QA system Alyssa for the TAC 2008 QA Track. The greatest challenge of this year's competition has been the development of an opinion-based QA system without almost any training data. Some components could only be built at short notice due to the late release of some sample questions just a few weeks before the submission deadline. Though the official results clearly show that there is considerable room for improvement, the last-minute construction of components, such as the blogger stream, proved beneficial for the overall performance of the system.

In future work, we would like to carry out a thorough error analysis using this year's evaluation results and thus identify and rectify bottlenecks of the system. We assume that a more sophisticated opinion/polarity ranking for all questions will be vital in achieving this goal. A comprehensive opinion-holder detection beyond the identification of bloggers might also be worthwhile. Last but not least, the judgments files from the TAC QA competition should allow us to optimize the parameter settings of various components of our system.

[13] According to our own evaluation, the accuracy of classification is approximately 95%.

Acknowledgements

Michael Wiegand was funded by the German research council DFG through the International Research Training Group "IRTG" between Saarland University and University of Edinburgh.

Saeedeh Momtazi and Fang Xu were funded by the German research council DFG through the Partnership for Research and Education "PIRE" between Saarland University, Charles University, Brown Laboratory for Linguistic Information Processing, and The Johns Hopkins University Center for Language and Speech Processing.

Gregorz Chrupała was funded by the BMBF project NL-Search under contract number 01IS08020B.

References

Steven Abney. 1991. Parsing by Chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing*, Dordrecht. Kluwer Academic Publishers.

Eric Brill. 1992. A Simple Rule-based Part of Speech Tagger. In *Proceedings of the Third Conference on Applied Natural language processing (ANL)*, Trento, Italy.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, Vancouver, BC, Canada.

Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1).

Marti A. Hearst. 1997. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1).

Soo-Min Kim and Eduard Hovy. 2006. Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sydney, Australia.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-Based Language Models. In *Proceedings of the Annual ACM Conference on Research and Development in Information Retrieval (SIGIR)*, New Orleans, LA, USA.

Bernardo Magnini, Matteo Negri, Reberto Prevete, and Hristo Tanev. 2002a. Comparing Statistical and Content-based Techniques for Answer Validation on the Web. In *Proceeding of the VIII Convegno AI*IA*, Siena, Italy.

Bernardo Magnini, Matteo Negri, Reberto Prevete, and Hristo Tanev. 2002b. Is It the Right Answer? Exploiting Web Redundancy for Answer Validation. In *Proceeding of the 40th Annual Meeting of the Association for Computational Linguisitcs (ACL)*, Philadelphia, PA, USA.

Yohei Seki, David Kirk Evans, Lun-Wei Ku, Hsin-Hsi Chen, Noriko Kando, and Chin-Yew Lin. 2007. Overview of Opinion Analysis Pilot Task at NTCIR-6. In *Proceedings of the 6th NTCIR Workshop Meeting*, Tokyo, Japan.

Dan Shen, Michael Wiegand, Andreas Merkel, Stefan Kazalski, Sabine Hunsicker, Jochen L. Leidner, and Dietrich Klakow. 2007. The Alyssa System at TREC QA 2007: Do We Need Blog06? In *Proceedings of the 16th Text Retrieval Conference (TREC)*, Gaithersburg, MD, USA.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2003. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*, 1:2.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. In *Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, Vancouver, BC, Canada.

Yiming Yang and Jan Pederson. 1997. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings the 14th International Conference on Machine Learning (ICML)*, Nashville, TN, USA.