TAC2008 QUESTION ANSWERING EXPERIMENTS AT TOKYO INSTITUTE OF TECHNOLOGY

Matthias H. Heie, Edward W. D. Whittaker, Josef R. Novak, Joanna Mrozinski and Sadaoki Furui

Tokyo Institute of Technology, Department of Computer Science
2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8552 Japan,
{heie,edw,novakj,mrozinsk,furui}@furui.cs.titech.ac.jp

ABSTRACT

In this paper we describe Tokyo Institute of Technology's submission to the TAC2008 question answering (QA) track. Keeping the same theoretical QA model as for the TREC2007 track, developed for factoid QA, we investigated the effects of retrieving blog data versus web data for rigid list questions. For squishy list questions we relied on sentence retrieval, similar to how we approached "other" questions in TREC2007. While our performance on rigid list questions was poor, for squishy list questions we achieved a score only slightly lower than the highest score of all participants.

1. INTRODUCTION

In this paper we describe the application of our data-driven and non-linguistic framework for the QA track of TAC2008. Two runs were submitted for evaluation: `asked081` and `asked082`.

In previous years' TREC QA evaluations [1] [2] [3], our focus has been the factoid questions, which were not part of the TAC2008 QA track. For list questions we have relied on an extension to our factoid QA system. Due to the similarity between the TAC2008 QA rigid list task and TREC QA list tasks, we decided to employ a similar system for the rigid list questions. We experimented with supplying our answer extraction module sentences retrieved from the Blog06 corpus, versus supplying web data. For sentence retrieval we used a language model based approach similar to what we did in TREC2007. In this approach, a language model (LM) is generated for each sentence and these models are combined with document LMs to take advantage of contextual information. From the retrieved information, we extracted rigid answers using our answer filter model.

For the squishy list questions, we submitted the highest ranking sentences from our sentence retrieval module, as many as the byte limit permitted. This is similar to how we approached the "other" question task in TREC2007.

2. SENTENCE RETRIEVAL

This section explains the LM-based sentence retrieval method presented in [4].

Language modeling for IR has gained in popularity over the last decade since the approach was proposed [5]. Under this approach a LM is estimated for each document. The documents are then ranked according to the conditional probability $P(Q \mid D)$, the probability of generating the query Q given the document D.

A language model based approach to sentence retrieval for QA is presented in [6]. Due to lack of data to train the sentence specific LM, it is assumed that all words are independent, hence unigrams are used:

$$P(Q \mid S) = \prod_{i=1}^{|Q|} P(q_i \mid S), \tag{1}$$

where q_i is the ith query term in the query $Q = (q_1 ... q_{|Q|})$ composed of $|Q|$ query terms.

Smoothing methods are normally employed with LMs to avoid the problem of zero probabilities when one of the query terms does not occur in the document. This is typically achieved by redistributing probability mass from the document model to a background collection model $P(Q \mid C)$. We use Dirichlet prior, where the probability of a query term q given a sentence S is calculated as:

$$P_1(q \mid S) \quad = \quad \frac{c(q;S) + \mu \cdot p(q \mid C)}{\sum_w c(w;S) + \mu}, \tag{2}$$

where $c(q;S)$ is the count of query term q in sentence S, μ is a smoothing parameter, $p(q \mid C)$ is the unigram probability of q according to the background collection model and $\sum_w c(w;S)$ is the count of all words in S.

A problem with the model presented in [6] is that words relevant to the sentence might not occur in the sentence itself, but in the surrounding text. For example, for the question *Where was George Bush born?*, the sentence *He was born in Connecticut* in an article about George Bush should ideally be assigned a high probability, despite the sentence missing

important query terms. To account for this, we train document LMs, $P_1(q \mid D)$, in the same manner as for $P_1(q \mid S)$ in Eq. (2), and perform a linear interpolation between $P_1(q \mid S)$ and $P_1(q \mid D)$:

$$P_2(q \mid S) = (1 - \alpha) \cdot P_1(q \mid S) + \alpha \cdot P_1(q \mid D), \quad (3)$$

where $0 \leq \alpha \leq 1$ is an interpolation parameter.

3. ANSWER EXTRACTION

For answer extraction we used the same framework as in TREC2007 [3], described in detail in [7].

We model the most straightforward and obvious dependence of the probability of an answer A on a question Q:

$$P(A \mid Q) = P(A \mid W, X), \quad (4)$$

where A and Q are considered to be strings of l_A words $A = a_1, \ldots, a_{l_A}$ and l_Q words $Q = q_1, \ldots, q_{l_Q}$, respectively. Here $W = w_1, \ldots, w_{l_W}$ represents a set of features describing the "question-type" part of Q such as *when*, *why*, *how*, etc. while $X = x_1, \ldots, x_{l_X}$ represents a set of features that describe the "information-bearing" part of Q i.e. what the question is actually about and what it refers to. For example, in the questions, *Where was Tom Cruise married?* and *When was Tom Cruise married?*, the information-bearing component is identical in both cases whereas the question-type component is different.

Finding the best answer $\hat{A}$ involves a search over all available A for the one which maximizes the probability of the above model i.e.,

$$\hat{A} = \arg\max_A P(A \mid W, X). \quad (5)$$

Given the correct probability distribution, this is guaranteed to give us the optimal answer in a maximum likelihood sense. We don't know this distribution and it is still difficult to model but, using Bayes' rule and making various simplifying, modeling and conditional independence assumptions (as described in detail in [7]) Eq. (5) can be rearranged to give

$$\arg\max_A \underbrace{P(A \mid X)}_{\substack{answer \\ retrieval \\ model}} \cdot \underbrace{P(W \mid A)}_{\substack{answer \\ filter \\ model}}. \quad (6)$$

The $P(A \mid X)$ model we call the *answer retrieval model*. In this year's evaluation we didn't use the answer retrieval model, i.e. $P(A \mid X)$ is uniform.

The $P(W \mid A)$ model matches a potential answer A with features in the question-type set W. For example, it relates place names with *where*-type questions. We call this component the *answer filter model* and it is structured as follows.

The question-type feature set $W = w_1, \ldots, w_{l_W}$ is constructed by extracting n-tuples ($n = 1, 2, \ldots$) such as *Who*, *Where* and *In what* from the input question Q. A set of single-word features is extracted based on frequency of occurrence in our collection of example questions.

Modeling the complex relationship between W and A directly is non-trivial. We therefore introduce an intermediate variable representing classes of example questions-and-answers (q-and-a) c_e for $e = 1 \ldots |C_E|$ drawn from the set C_E. In order to construct these classes, given a set E of example q-and-a, we then define a mapping function $f : E \mapsto C_E$ which maps each example q-and-a t_j for $j = 1 \ldots |E|$ into a particular class $f(t_j) = e$. Thus each class c_e may be defined as the union of all component q-and-a features from each t_j satisfying $f(t_j) = e$. Finally, to facilitate modeling we say that W is conditionally independent of c_e given A so that

$$P(W \mid A) = \sum_{e=1}^{|C_E|} P(W \mid c_W^e) \cdot P(c_A^e \mid A), \quad (7)$$

where c_W^e and c_A^e refer respectively to the subsets of question-type features and example answers for the class c_e.

Assuming conditional independence of the answer words in class c_e given A, and making the modeling assumption that the jth answer word a_j^e in the example class c_e is dependent only on the jth answer word in A we obtain:

$$P(W \mid A) = \sum_{e=1}^{|C_E|} P(W \mid c_e) \cdot \prod_{j=1}^{l_{A^e}} P(a_j^e \mid a_j). \quad (8)$$

Since our set of example q-and-a cannot be expected to cover all the possible answers to questions that may be asked we perform a similar operation to that above to give us the following:

$$P(W \mid A) = \sum_{e=1}^{|C_E|} P(W \mid c_e) \prod_{j=1}^{l_{A^e}} \sum_{k=1}^{|C_A|} P(a_j^e \mid c_k) P(c_k \mid a_j), \quad (9)$$

where c_k is a concrete class in the set of $|C_A|$ answer classes C_A. The independence assumption leads to underestimating the probabilities of multi-word answers so we take the geometric mean of the length of the answer (not shown in Eq. (9)) and normalize $P(W \mid A)$ accordingly.

4. EXPERIMENTAL WORK

Two different runs (`asked081` and `asked082`) were submitted for evaluation, with their data sources given in Table 1.

4.1. Data pre-processing, indexing and retrieval

Raw text was extracted from the XML format of the Blog06 collection. This text was cleaned using a series of regular

Run	Rigid	Squishy
asked081	Blog06 sentences *incl.* context	Blog06 sentences *excl.* context
asked082	Web snippets	Blog06 sentences *incl.* context

Table 1. Data sources of the 2 runs submitted to TAC2008.

Run	Rigid					Squishy	Avg. per-series score
	Correct	Distinct	Unsupported	Non-exact	F-score	F-score	
asked081	11	10	6	8	0.011	0.173	0.093
asked082	2	2	5	7	0.003	0.132	0.068

Table 2. Performance of the 2 submitted runs, for 90 rigid list questions and 87 squishy list questions.

Rigid			Squishy		
High	Median	Low	High	Median	Low
0.156	0.063	0.000	0.186	0.091	0.018

Table 3. Summary of F-scores of all runs for all teams.

expressions. The pre-processed documents were then indexed using the Xapian search engine library[1]. A set of stopwords was used during indexing and retrieval. 150 documents were retrieved for each question. The documents were sentence segmented, using a rule-based algorithm, and the sentences were ranked using the language modeling approach explained in Section 2.

For web data we used the snippets returned by the *Yahoo* search engine. These snippets were cleaned in the same way as the Blog06 documents.

Questions were cleaned in the same way as retrieved text. If the target for a question did not appear character-for-character in the question string, it was simply appended to the end of the question string. Stopwords were removed, in addition to common question-type words such as *list* and *name*, etc. Each question was treated independently of all other questions.

4.2. Rigid list question task

In asked081, for each question, the top 100 Blog06 sentences and their contexts (the immediately preceding and succeeding sentence), were passed to the answer extraction module. In asked082 the top 100 web snippets were used.

Our factoid QA system always outputs a list of candidate answers ranked by their probabilities. The issue for the rigid list task is therefore to determine how many of the top answers to submit so as to maximize the F-score. We chose simply to submit the top 10 answers of the answer extraction module.

[1] http://www.xapian.org/

4.3. Squishy list question task

We used our sentence retrieval module to answer squishy list questions, as we did for "other" questions in the TREC2007 QA evaluation. In asked081 we submitted the highest ranking sentences, as many as the limit of 7000 bytes permitted. In asked082 we submitted the highest ranking sentences and their contexts (the immediately preceding and succeeding sentence), again as many as the byte limit permitted.

5. RESULTS AND DISCUSSION

The results for the 2 submitted runs on the 2 tasks are shown in Table 2.

Our system is essentially a factoid QA system. We have previously used an extended version of this system for TREC QA list questions, as explained in Section 4.2, but we have performed poorly on the list task in previous evaluations. The results show that in this year's TAC QA track our rigid list scores are considerably lower than the median, shown in Table 3. When considering only the best run of each participant, we had the lowest score. Many of the questions were "who" questions, where the system is asked to provide the name of opinion holders, such as the name of a blog or its owner, or a blog poster. Our system consistently failed to answer these questions correctly.

Using the Blog06 collection as data source yields a significantly higher F-score than using web data. To some extent this can be explained by the fact that only supported answers count in the calculation of the F-score. Finding a supporting document in the Blog06 collection is naturally more difficult if the answer is extracted from a different collection. But also when non-exact and unsupported questions are considered, we are able to extract more correct answers using the Blog06 collection than the web: 25 vs. 14.

Figure 1 shows to what extent we are able to retrieve correct answers in the sentence retrieval stage. The graph shows how many of the 90 rigid list questions have N or more unique reference answers within the retrieved sentences, as the number of sentences varies. Note that if *# sentences per*

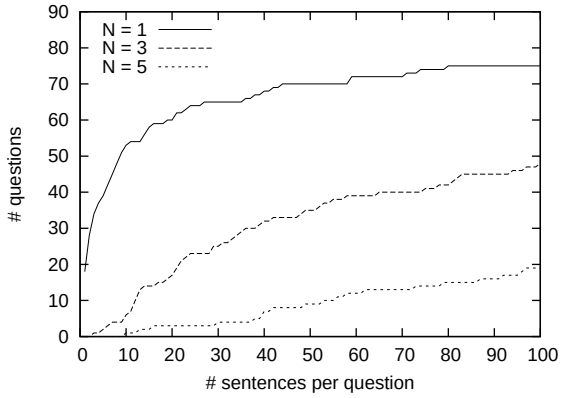

Fig. 1. Number of rigid list questions (out of 90) with at least N reference answers in the retrieved sentences, as the number of retrieved sentences varies.

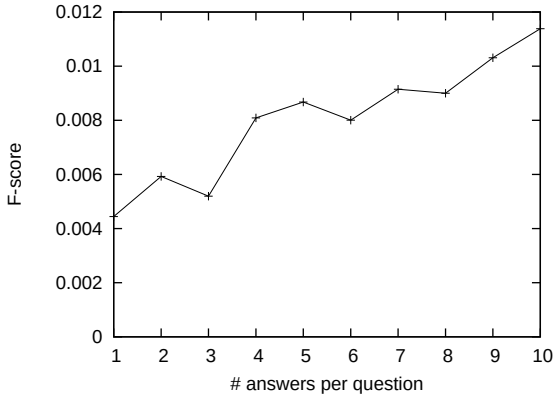

Fig. 2. Average F-score of 90 rigid list questions as number of submitted answers varies.

question is *M*, it means that the top *M* sentences, plus their immediately preceding and succeeding sentences, is retrieved, as in `asked081`.

We also investigated the appropriateness of submitting 10 answers per question. Figure 2 shows the average F-score as the number of submitted answers varies, using the setup of `asked081`. It shows that submitting fewer than 10 answers per question would have reduced our F-score.

The squishy list results show that the score of our best run is only slightly lower than the highest scoring run of all participants. When considering only the best run of each participant, we had the second highest score, despite not performing any opinion analysis. Our best run is `asked081`, i.e. we achieve a better result by not submitting surrounding sentences, which allows us to submit more lower-ranking sentences. Figure 3 shows the F-score as the number of submitted bytes varies. We achieve the highest score by submitting as many sentences as the byte limit allows (7000 bytes), as we did in the evaluation.

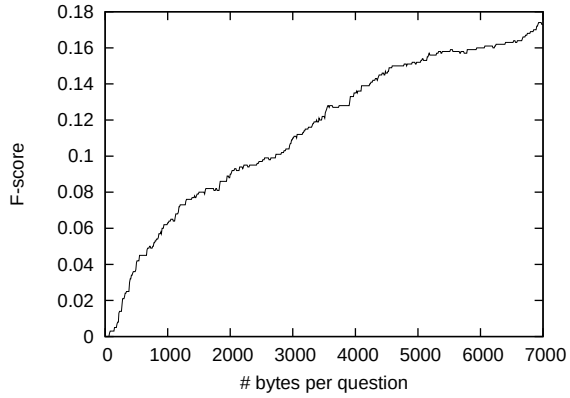

Fig. 3. Average F-score of 87 squishy list questions as number of submitted bytes varies.

6. CONCLUSION

In this paper we have given an overview of our methods and results for the TAC2008 question answering evaluation. Our results show that our squishy list QA system, based on sentence retrieval, is able to achieve a high score, relative to the other teams. Using our factoid QA system for the rigid list question task did not yield good performance. We are able to achieve a higher score on this task by relying on Blog06 data, rather than web data.

7. REFERENCES

[1] Whittaker, E., Chatain, P., Furui, S. and Klakow, D., "TREC2005 Question Answering Experiments at Tokyo Institute of Technology", *Proc. TREC-14*, 2005.

[2] Whittaker, E., Novak, J., Chatain, P. and Furui, S., "TREC2006 Question Answering Experiments at Tokyo Institute of Technology", *Proc. TREC-15*, 2006.

[3] Whittaker, Heie, M., Novak, J. and Furui, S., "TREC2007 Question Answering Experiments at Tokyo Institute of Technology", *Proc. TREC-16*, 2007.

[4] Heie, M., Whittaker, E., Novak, J. and Furui, S., "A Language Modeling Approach to Question Answering on Speech Transcripts", *Proc. ASRU*, 2007, pp. 219-224.

[5] Ponte, J. and Croft, W., "A Language Modeling Approach to Information Retrieval", *Proc. SIGIR*, 1998, pp. 275-281.

[6] Merkel, A. and Klakow, D., "Comparing Improved Language Models for Sentence Retrieval in Question Answering", *Proc. CLIN-17*, 2007.

[7] Whittaker, E., Furui, S. and Klakow, D., "A Statistical Pattern Recognition Approach to Question Answering using Web Data", *Proc. Cyberworlds*, 2005, pp. 421-428.