DalTREC 2004: Question Answering using Regular Expression Rewriting

Vlado Kešelj and **Anthony Cox**
Faculty of Computer Science
Dalhousie University, Halifax, Canada
{vlado,amcox}@cs.dal.ca

Abstract

This is the first year that the Dalhousie University participated in TREC. We submitted three runs for the QA track. Our evaluation results are generally below the median (with one exception) but seem to be significantly higher than the worst scores, which is within our expectations considering a limited time spent on developing the system. Our approach was based on the regular expression rewriting and the use of external search engines (MultiText and PRISE). One run used Web-reinforced search.

1 Introduction

2004 is the first year that the Dalhousie University participated in TREC. The project DalTREC[1] was initiated in 2003 and serves as an umbrella for TREC-related activities at the Dalhousie university. The topics of interest were Question Answering, HARD, Genomics, and Web. We submitted the runs for the Question Answering (QA) track.

In our previous work, we explored the application of several approaches to question answering in the overlapping area of unification-based and stochastic NLP (Natural Language Processing) techniques (Kešelj, 2002; Cercone et al., 2002). Two novel methods that were explored relied on the notions of *modularity* and *just-in-time sub-grammar extraction.*

One of the learned lessons of the previous experiments was that the regular expression (RegEx) substitutions are a very succinct, efficient, maintainable, and scalable method to model many NL subtasks of the QA task. This was also observed in the context of lexical source-code transformations of arbitrary programming languages (Cox et al., 2004), where it is an alternative to manipulations of the abstract syntax tree. In this context, RegEx transformations are more robust in the face of missing header files, errors, usage of macros, templates, and other embedded programming language constructs.

2 Regular Expression Rewriting

The basic method used at various components of the QA system is *RegEx rewriting.* The open angle bracket (`<`) is used as a special escape character, hence we make sure that it

[1] http://www.cs.dal.ca/~trecacct — DalTREC URL

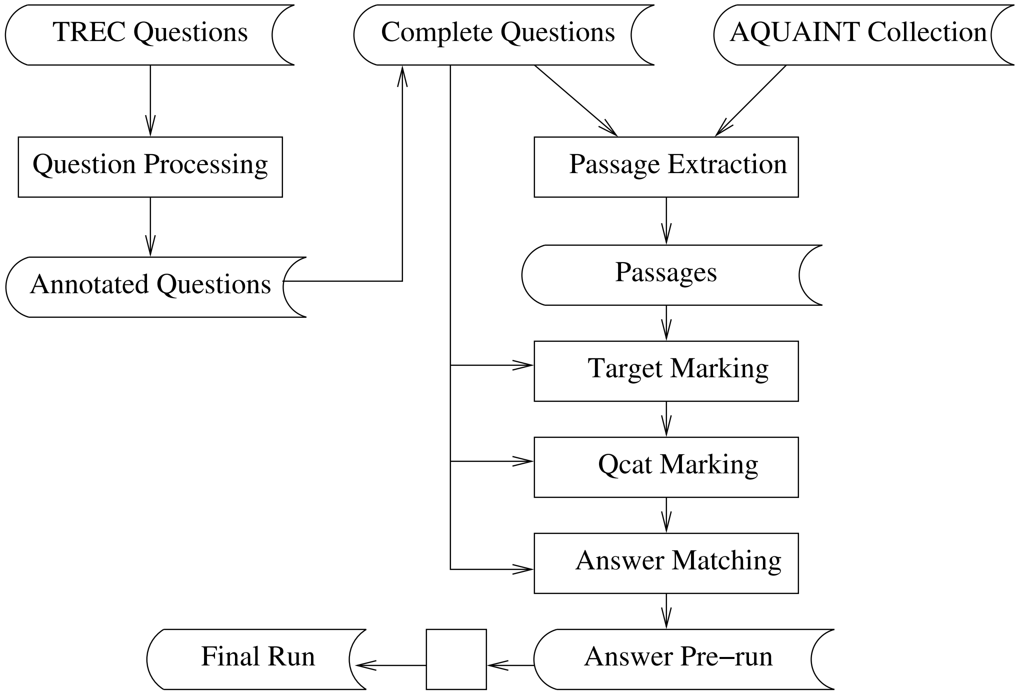

Figure 1: System Overview

does not appear in the source text, which is either a question or a passage. The basic text substrings, such as the target or named entities, are recognized using regular expressions and replaced with an angle-bracket-delimited expression. For example, the target is marked as `<TARGET>`. More commonly, a named entity e of type t is replaced with <t_e_s>, where e_s is the named entity e encoded as a string of printable characters that do not include <. The RegEx rewriting can be seen as a bottom-up deterministic parsing technique. For example, the rewriting in which "<NP_x> <VP_y>" is replaced with "<S_z>" corresponds to the context-free rule $S \rightarrow NP\ VP$. The value z is obtained by decoding x and y, concatenating them, and encoding the result again.

3 System Architecture

An overview of the system architecture is shown in figure 1 and it is described in more detail in the remaining of this section.

Question processing component takes the original TREC questions as the input and produces an annotated list of questions. The process includes question parsing, generating complete questions, detecting question category, and producing some additional auxiliary information. The original TREC 2004 questions are grouped by targets. A set of questions around a target may include just a anaphoric reference to the target and not the full target name (e.g., 'it', 'he'). In order to be able to independently use each question in passage extraction, they are rewritten so that they include the target name. We call this form a "complete question" or "full question". Additionally, as the result of parsing the questions, we obtain question category (i.e., the expected answer type), and some other optional information, such as type of the relation between the target and the answer. Question parsing and generating full questions is based on regular expression rewriting rules.

Generating the full question was done in the following way: We start with the original question. If target appears in the question, it is not changed. For a question of type "other," we generate the question "`What is <Target> ?`". Otherwise, we attempt the following substitutions:

```
elsif ($Qtype eq 'OTHER') { $_ = " What is <Target> " }
elsif (/ it /)      { s/ it / <Target> / }
elsif (/ its /)     { s/ its / <Target>'s / }
elsif (/ he /)      { s/ he / <Target> / }
elsif (/ his /)     { s/ his / <Target>'s / }
elsif (/ she /)     { s/ she / <Target> / }
elsif (/ her /)     { s/ her / <Target>'s / }
elsif (/ they /)    { s/ they / <Target> / }
elsif (/ their /)   { s/ their / <Target>'s / }
elsif (/ theirs /)  { s/ theirs / <Target>'s / }
elsif (/^ It /)     { s/ It / <Target> / }
elsif (/^ Its /)    { s/ Its / <Target>'s / }
elsif (/^ He /)     { s/ He / <Target> / }
elsif (/^ His /)    { s/ His / <Target>'s / }
elsif (/^ She /)    { s/ She / <Target> / }
elsif (/^ Her /)    { s/ Her / <Target>'s / }
elsif (/^ They /)   { s/ They / <Target> / }
elsif (/^ Their /)  { s/ Their / <Target>'s / }
elsif (/^ Theirs /) { s/ Theirs / <Target>'s / }
```

If none of them succeeds, we prepend "`<Target>, `" to the original question.

Passage Retrieval. The passage retrieval from the AQUAINT data set is performed by an external search engine using the full questions generated in the question processing phase. We used the passages retrieved by the MultiText search engine (University of Waterloo), and the documents provided by NIST (produced by the PRISE search engine). In both cases, the MultiText and PRISE, the results are treated in the same way—as passages relevant to the question.

Target Marking. Using RegEx rewriting rules, the target is identified in the passages and replaced with the `<TARGET>` tag.

Question Category (Qcat) Marking. All entities having the identified question category (Qcat) type are marked using RegEx rewriting.

Answer Matching is based on RegEx matching. For example, the following pattern-processing snippet

```
while (/<${Qcat}_([^>]*)> *<TARGET>/g) { ... }
```

matches the string "Italian Alberto Tomba", because after target marking and Qcat marking, the string becomes "`<NATIONALITY_Italian> <TARGET>`".

Pre-run to Run filtering. The result of matching is a list of answers for each question—an unlimited number of answers may appear for any question, or we may have no answers at all. The task of the pre-run-to-run filtering component is to prepare the final run using the following rules:

- only the first answer is passed for any factoid question (TREC requirement),
- a NIL answer is introduced for questions with no generated answers,
- not more than seven answers are allowed per list question (rule of thumb),
- all answers to a list or 'other' question have to be unique,
- additionally, answers to an 'other' question may not appear as an answer to any previous question about the same target, and
- any answer longer than 100 bytes is truncated.

In the case of web-reinforced question-answering (run Dal04x), this phase is used to locate relevant documents as well (see the next section).

4 Evaluation

We submitted the following three runs:

Dal04b — the "base" run using MultiText AQUAINT passages;

Dal04x — the "extended" run using MultiText AQUAINT passages and web passages, and re-finding the answer in the AQUAINT part, and

Dal04p — the "PRISE" run using the documents provided by TREC, from the PRISE search engine.

For the run Dal04x we used web reinforced question-answering: We used relevant passages returned by the MultiText engine from the AQUAINT data set, but also from the Web data collected by the MultiText group. This means that the answers found in the pre-run may not be supported by the AQUAINT documents. For those answers, the system makes attempt to find them in the AQUAINT data set, and only if found there, they are included in the final run with the appropriate document ID. This is adopted as a general method for using external knowledge resources. This year, it is only used in the Dal04x run.

4.1 Evaluation Summary

	Factoid Questions						
		Number of	NIL Questions				
Run	Accuracy	Correct Ans.	P	R	List	Other	F Overall
Dal04b	0.126	29	0.071	0.091	0.051	0.048	0.088
Dal04x	0.130	30	0.080	0.091	0.052	0.049	0.090
Dal04p	0.113	26	0.107	0.136	0.105	0.113	0.111
TREC best	0.770	177			0.622	0.460	
TREC median	0.170	39			0.094	0.184	
TREC worst	0.009	2			0.000	0.000	

4.2 Analysis

Our focus was on the factual questions, and in that respect the Dal04x had the best performance. It came as a surprise that Dal04p performed so well regarding the list questions; it is the only score which is larger than the TREC median, and it resulted in Dal04p having the best performance overall.

5 Conclusions and Future Work

We present a relatively simple QA framework based on regular expression rewriting. Three runs were submitted for the QA track. Our evaluation results are generally below the median (with one exception) but seem to be significantly higher than the worst scores. The results fall within our expectations since this is our first TREC participation and we could devote only a minimal number of person-hours to the project.

Acknowledgments

We would like to thank Charlie Clarke for providing MultiText results for the TREC questions from the AQUAINT data set and his Web collection.

References

Nick Cercone, Lijun Hou, Vlado Kešelj, Aijun An, Kanlaya Naruedomkul, and Xiaohua Hu. 2002. From computational intelligence to web intelligence. *IEEE Computer*, 35(11):72–76, November.

Anthony Cox, Tony Abou-Assaleh, Wei Ai, and Vlado Kešelj. 2004. Lexical source-code transformation. In *Proceedings of the STS'04 Workshop at GPCE/OOPSLA*, Vancouver, Canada, October.

Vlado Kešelj. 2001. Question answering using unification-based grammar. In Eleni Stroulia and Stan Matwin, editors, *Advances in Artificial Intelligence, AI 2001*, volume LNAI 2056 of *Lecture Notes in Computer Science, Springer*, pages 297–306, Ottawa, Canada, June.

Vlado Kešelj. 2002. Modular stochastic HPSGs for question answering. Technical Report CS-2002-28, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, June.