ILQUA At TREC 2006

Min Wu **Tomek Strzalkowski**
ILS Institute
University at Albany SUNY
1400 Washington Ave. Albany, NY, USA
{minwu,tomek}@cs.albany.edu

1 Introduction

This year, we made changes to the passage/sentence retrieval component of ILQUA in handling factoid and list questions. All the other components remain same.

ILQUA is an IE-driven QA system. To answer "Factoid" and "List" questions, we apply our answer extraction methods on NE-tagged passages or sentences. The answer extraction methods adopted here are surface text pattern matching, n-gram proximity search, and syntactic dependency matching. Although surface text pattern matching has been applied in some previous TREC QA systems, the patterns used in ILQUA are better since they are automatically generated by a supervised learning system and represented in a format of regular expressions which contain multiple question terms. In addition to surface pattern matching, we also adopt n-gram proximity search and syntactic dependency matching. N-grams of question terms are matched around every named entity in the candidate sentences or passages and a list of named entities are generated as answer candidate. These named entities then go through a multi-level syntactic dependency matching component until a final answer is generated. To answer "Other" questions, we parsed the answer sentences of "Other" questions in previous main task and built syntactic patterns combined with semantic features. These patterns are later applied to the parsed candidate sentences to extract answers of "Other" questions.

Figure 1 shows the diagram of the ILQUA architecture.

2 Question Analysis

ILQUA classifies questions by syntactic structure and answer target. This year, we use Stanford parser instead of the ME (Maximum Entropy) parser which we applied in our previous ILQUA systems.

Syntactic chunking splits questions into a list of question terms with syntactic tags. For example, the question "When were the first postage stamps issued in the United States" is chunked with the syntactic structure of "When_Be_NP_VP_NP". However, certain questions with special answer patterns are not chunked in this manner. They are categorized as "Born_When", "Born_Where", "Die_When", "Die_Where", "Abbreviation", etc.

Since the answer extraction is applied on NE-tagged passages, the answer targets of questions are classified into named entity types. We use BBN's Identifinder that can identify 24 types of named entities to annotate AQUAINT corpus. In addition, we developed a small NE tagger to annotate three types of named entities that are not included in Identifinder's list. The following shows all the named entity types that our system can process.

ANIMAL / CONTACT_INFO / DISEASE
EVENT / FAC / GAME / GPE / LANGUAGE
LAW / LOCATION / NATIONALITY
ORGANIZATION / PERSON / PLANT
PRODUCT / SUBSTANCE / WORK_OF_ART
CARDINAL / MONEY / ORDINAL / PERCENT
"QUANTITY / DATE / TIME
COLOR / GENRE / OCCUPATION

For questions beginning with the words "When", "Where" and "Who", the answer target assigned is "Date", "Location" and "Person/Organization", respectively. For questions beginning with pattern "How+Adj.", there are handcrafted rules to assign answer target. For questions beginning with "What_Be", "What_Entity", "Which_Be", "Which_Entity", a key term of noun phrase is extracted from the question and mapped to appropriate answer target type. We set up a

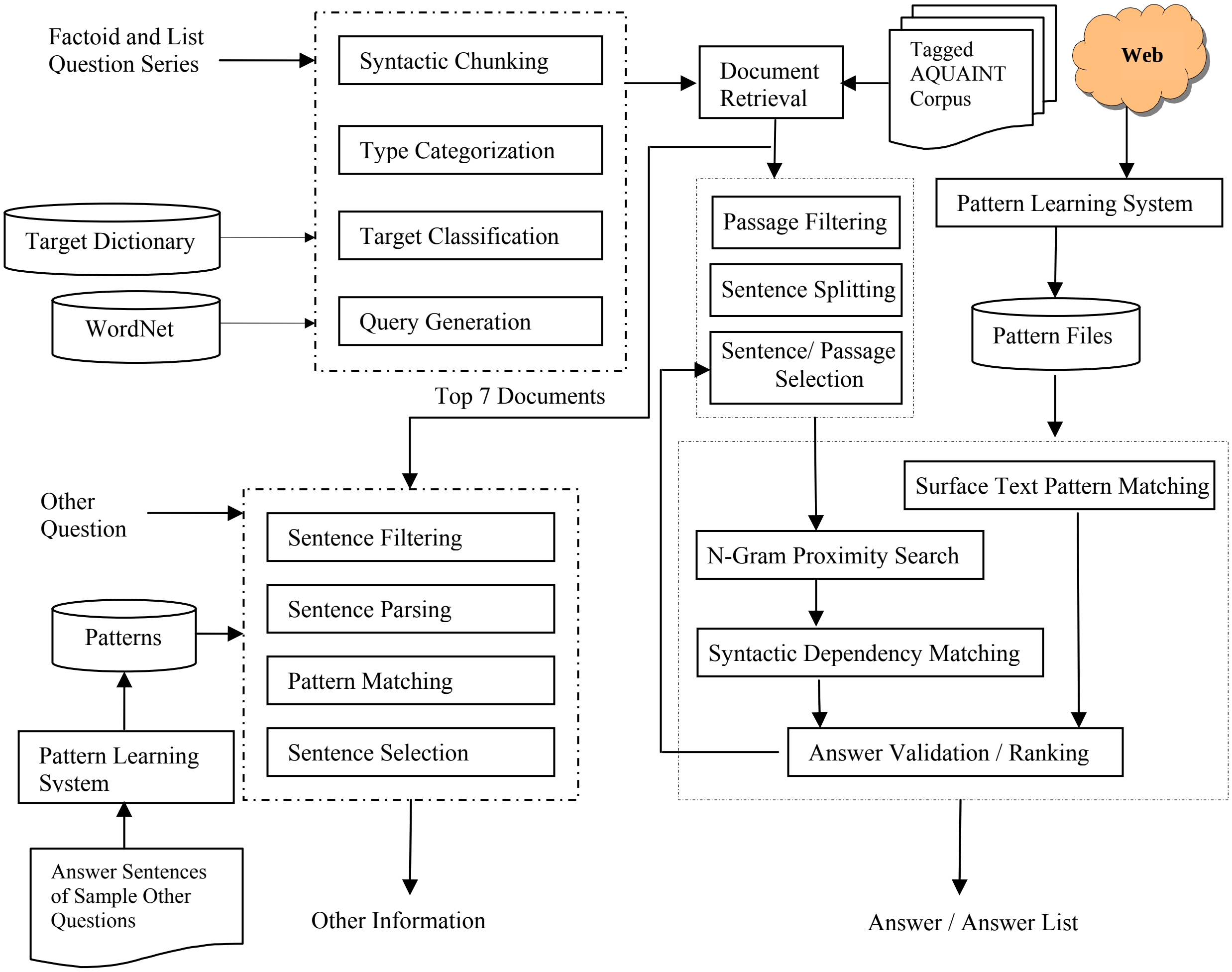

Figure 1 ILQUA System Architecture

noun-target map of 7885 entries to map noun to named entity type which can be processed by the system. The assigned entity type is set as the major answer target type while the noun is set as the minor answer target type. For example, if the major answer target is “Quantity”, the minor answer target could be “age”, “distance”, “height”, “speed”, etc. This two-level answer target categorization is helpful to answer validation.

Query expansion is done with the aid of WordNet to find the morphological forms and synonyms of verbs. Some common verbs are filtered out to increase the retrieval precision. We didn’t use the noun synonyms to expand the query mainly because the noise introduced by some synonyms would reduce the retrieval precision.

3 Passage/Sentence Retrieval

The IR engine we used is Inquery developed at University of Massachusetts at Amherst. The top 100 documents retrieved by Inquery are tagged and segmented into passages. These passages are then filtered by answer target types, question terms, and topic terms. Passages without named entity of answer target type are filtered out. All the NE tags in the passages, except the tags of answer target, are filtered out for later processing.

In our previous ILQUA system, the answer selection component was designed to work on passage level based on the observation that in many occasion answers to questions and question terms occur in consecutive sentences of same passage. However in some cases, this passage-level answer selection gave wrong answer candidate. So this year we introduced two-phase answer selection: we applied answer selection on candidate sentences first; in case the selected answer score was too low, we applied the answer selection on candidate passages as well. Accordingly the information retrieval component in this year’s ILQUA prepares both candidate passages and sentences to the answer selection module.

4 Surface Text Pattern Matching

Surface text pattern matching has been adopted by some researchers (Ravichandran & Hovy 2002, Soubbotin 2002) in building QA system during the last few years. The patterns used in ILQUA are automatically learned and extracted. They are sorted according to question types and can handle more anchor terms.

4.1 Pattern Format

Patterns are represented as regular expressions with terms of “NP”, “VP”, “VPN”, “ADVP”, “be”, “in”, “of”, “on”, “by”, “at”, “which”, “when”, “where”, “who”, “,”, “-“, “(“, etc. Since most TREC questions contain more than one noun phrase, we numbered these noun phrases according to their occurring order in the question. The following gives some sample patterns of question type “when_do_np_vp_np”.

```
NP1 VP NP2 in <Date>([^<>]+?)<\/Date>
NP1 VP NP2 on <Date>([^<>]+?)<\/Date>
NP1.{1,15}VP NP2 in <Date>([^<>]+?)<\/Date>
NP2 in <Date>([^<>]+?)<\/Date>.{1,15}NP1
<Date>([^<>]+?)<\/Date> NP1 VP.{1,15}NP2
NP1.{1,15}VP.{1,30} on <Date>([^<>]+?)<\/Date>
NP1.{1,15}NP2 on <Date>([^<>]+?)<\/Date>
NP1.{1,30}NP2 on <Date>([^<>]+?)<\/Date>
<Date>([^<>]+?)<\/Date>.{1,15}NP1.{1,50}VP
NP1's NP2 in <Date>([^<>]+?)<\/Date>
<Date>([^<>]+?)<\/Date>.{1,15}NP1 VP NP2
NP1 VPN.{1,15}NP2 in <Date>([^<>]+?)<\/Date>
```

When applying these patterns to specific question, the terms such as “NP”, “VP”, “VPN”, “ADVP” and “be” should be replaced with the corresponding question terms. The replaced patterns can be matched directly to the candidate passages and answer candidate can be extracted quickly with Java tools. The number of patterns depends on the specific question type. Some question types have up to 500 patterns. Only patterns with score greater than some empirically determined threshold are applied in pattern matching.

Patterns are sorted by question types and stored in pattern files. We have patterns for more than 50 question types. Figure 2 lists question types of “When” questions. Although adverb phrases as question terms are necessary in the patterns, to simplify categorization, we didn’t include them in question type labels.

1. **when_be_np**
   When was the first space shuttle flight?
   When was the Hellenistic Age?
2. **when_be_np_pp_np**
   When was the U.S. invasion of Haiti?
   When was the battle of Shiloh?
3. **when_be_np_pos_np**
   When is Mexico's independence?
4. **when_be_np_vp**
   When was "Cold Mountain" written?
   When was Carlos the Jackal captured?
5. **when_be_np_pp_np_vp**
   When was the battle of Chancellorsville fought?
   When was the city of New Orleans founded?
6. **when_be_np_vp_pp_np**
   When was Jim Inhofe first elected to the senate?
   When was the Panama Canal returned to Panama?
7. **when_do_np_vp**
   When did the first American lighthouse open?
   When did the shuttle Challenger explode?
8. **when_do_np_vp_np**
   When did the United States enter the World War II?
   When did Amtrak begin operations?
9. **when_do_np_vp_pp_np**
   When did "The Simpsons" first appear on television?
   When did Jack Welch retire from GE?

Figure 2 “When” Question Types

4.2 Pattern Extraction and Supervised Iterative Optimization

We used TREC11, TREC12 and TREC13 questions and answers as sample question-answer pairs. These questions are classified into question types. Questions with complex syntactic structure are omitted and questions whose answer target cannot be tagged by ILQUA are skipped too. Some frequently asked question types contain more sample questions than others. In order to overcome the data sparseness problem, we formulated some sample question-answer pairs for those question types with fewer questions.

The sample question is analyzed by the question analysis component of ILQUA and the expanded query includes answers to this question. In addition to retrieving documents from AQUAINT corpus, we also mined the web data. For each question-answer pair, we chose the top fifty documents retrieved by Google. We did not do deep mining beyond 50 documents, as it does not add to performance. Some questions with more than one

Question: *Where was the first Kibbutz founded*?

NP1 --- first Kibbutz VP --- found, founds, founded, establish, established, established

Annotated Passage:

<Location>Israel</Location>'s first Kibbutz was established in 1908 on the shores of the <Location>Sea of Galiee</Location> as the realization of the ideology of the communal. Its members were organized on the basis of public ownership. Its principle was self-reliance, equality and cooperation in production, consumption and education.

Extracted Patterns:

<Location>([^<>]+?)</Location>'s NP1 be VP --- activation count ++

NP1 be VP in.{1,30}<Location>([^<>]+?)</Location> --- activation count ++; correct activation ++

Figure 3 Pattern Extraction Illustration

correct answer will be retrieved many times with each answer in the query from web. The retrieved documents are tagged with Identifinder and filtered. The filtered passages are prepared as input of pattern extraction.

For every named entity in the passage, a pattern is extracted and validated. If the currently extracted pattern is new, it is appended to the pattern list that contains all the patterns extracted so far. The activation count of the pattern is thus increased by one. If the pattern is generated by the correct answer, the correct activation count is increased by one. Figure 3 shows how patterns are extracted and Figure 4 shows how to score the extracted patterns by their precision and frequency. During the experiment, it was observed that some rare patterns get high precision scores. This is caused by data sparseness. We introduced precision tuning parameter ε here to adjust the precision score.

Rather than stopping at only automatic pattern extraction, we found that supervised iterative optimization is necessary to get more accurate pattern distribution. In our experiment, we applied the patterns with score greater than 0.5 to the sample questions and examined the answer extracted. For some questions, correct answers not included in the training question-answer pairs were found. For example, question "When was Jerusalem invaded by the general Titus" was processed with "70 A.D.", "A.D.70" and "70 Ad" as answers. However, among the answers extracted by the learned patterns, we found answer "70AD" occur several times. Such case occurred frequently in our experiment. So we added the newly extracted answer to the training question-answer pairs and retrained the system. Retraining is done until no new patterns are found.

Pi (*i=1,2,... n*) --- pattern list
Ai (*i=1,2,... n*) --- activation counts
Ci (*i=1,2,... n*) --- correct activation counts

Expected Precision:

$$\mathrm{Pr}_i \approx \frac{C_i}{A_i + \varepsilon}$$

$\varepsilon = 0$ if $A_i \geq 5$; $\varepsilon = 1$ if $3 \leq A_i \leq 4$; $\varepsilon = 2$ if $A_i \leq 2$

Frequency:

$$F_i = \frac{A_i}{\sum_{k=1}^{n} A_k}$$

Final Score:

$$S_i = \mathrm{Pr}_i \times (1 + F_i)$$

Figure 4 Pattern Scoring

Finally, patterns are manually refined by merging similar patterns and removing bad patterns.

5 N-gram Syntactic Dependency Matching

Proximity search as an IR method has been used in QA too (Han, Chung and Kim 2004). To answer questions whose answer cannot be extracted by surface text pattern matching, we applied a combined method of n-gram proximity search and syntactic dependency matching. N-gram proximity search is an effective method to quickly filter out irrelevant information and focus the answer selection on viable candidates.

Around every named entity in the filtered candidate passages, question terms as well as topic terms are matched as n-grams. A question term is tokenized by word. We matched the longest possible sequence of tokenized word within the 100 word sliding window around the named entity. Once a sequence is matched, the corresponding word tokens are removed from the token list and the same searching and matching is repeated until the token list is empty or no sequence can be matched. The candidate named entity is scored by the average weighted distance score of question terms and topic terms.

Let $Num(t_i...t_j)$ denotes the number of all matched n-grams, $d(E, t_i...t_j)$ denotes the word distance between the named entity and the matched n-gram, $W1(t_i...t_j)$ denotes the topic weight of the matched n-gram, $W2(t_i...t_j)$ denotes the length weight of the matched n-gram. If $t_i...t_j$ contains topic terms or question verb phrase, 0.5 is assigned to *W1,* otherwise 1.0 is assigned. The value assigned to length weight *W2* is determined by λ, the ratio value of matched n-gram length to question term length. How to assign the value of *W2* is illustrated as follows.

$$W2(t_i...t_j)=0.4 \quad if\ \lambda<0.4$$
$$W2(t_i...t_j)=0.6 \quad if\ 0.4 \leq\lambda<0.6$$
$$W2(t_i...t_j)=0.8 \quad if\ \lambda>0.6$$
$$W2(t_i...t_j)=0.9 \quad if\ \lambda<0.75$$

The weighted distance score *D(E,QTerm)* of the question term and the final score *S(E)* of the named entity are calculated as follows.

$$D(E,QTerm) = \frac{\sum_{t_i...t_j} \frac{d(E,t_i...t_j)\times W1(t_i...t_j)}{W2(t_i...t_j)}}{Num(t_i...t_j)}$$

$$S(E) = \frac{\sum_{i}^{N} D(E,QTerm_i)}{N}$$

The n-gram proximity search generates a list of named entities as answer candidates. The syntactic dependency matching component takes the top 20 as input and gives the final answer. Figure 5 shows the top 5 answer candidates of question “When was the first Burger King restaurant opened?” after the n-gram proximity search.

When was the first Burger King restaurant opened?

1. The number of ***Burger King*** fast-food ***restaurant***s have reached 100 throughout Turkey since ***first*** was ***opened*** in **1995**, reported the Anatolia News Agency on Sunday.
2. Coke has supplied Burger King for most of the ***restaurant*** chain's history, starting with ***the first Burger King*** that ***opened*** in Miami in **1954**.
3. ``The company chose an available Pillsbury pancake mix brand name, Hungry Jack's, in its place. . . . ***Burger King opened*** its ***first*** four company-owned ***restaurant***s (under the Burger King name) in Sydney, New South Wales . . . in **December 1997**."
4. why some BK look-alike restaurants in Australia are named Hungry Jack's while others bear the Burger King moniker. ``The Burger King brand name was not available for use by ***Burger King*** Corp. in **1971**," BK says. ``The company chose an available Pillsbury pancake mix brand name, Hungry Jack's, in its place. . . . Burger King ***opened*** its ***first*** four company-owned ***restaurant***s .....
5. When the recall was ***first*** announced **Dec. 27**, ***Burger King*** placed an ad in USA Today, posted signs in its ***restaurant***s and sent out notices to 56,000 pediatricians.

Figure 5 Answer Candidate List

We use MINIPAR (DeKang Lin) to parse the question and the sentences containing the answer candidates. The syntactic relation triples in the question are matched one by one against the parsed sentences. We use the dependency-based word similarity list (developed by DeKang Lin) to match the synonyms or highly related words. To improve the matching accuracy, we introduced the forward matching propagation and the backward matching propagation. If there are two syntactic relations A:R1:B and B:R2:C in the question, suppose A:R:B is not matched against any relations in the parsed sentence, the forward propagation will consider the relation A:R1:C or A:R2:C. Suppose A:R1:B is matched with the parsed sentence and B:R2:C is not matched with any relations in the parsed sentences, the matching score of A:R1:B will be adjusted according to the rule of backward propagation.

The parsed dependency relation triples of the question in the Figure 5 are listed as follows.

```
~           Q:wha:A         when
~           Q:head:YNQ      ~
~           YNQ:inv-be:be   be
~           YNQ:head:V      open
open    V:s:N       restaurant
restaurant          N:det:Det           the
restaurant          N:post:PostDet      first
restaurant          N:nn:N  Burger King
Burger King         N:lex-mod:U         Burger
```

There are two main syntactic dependency relations: the first relation is between "when" and "open" and the second relation is between "open" and "the first Burger King restaurant".

The relations are then matched with the parsed sentences in the Figure 5. In the first round of syntactic matching, answer candidates "1995", "1954" and "December 1997" are matched. In the second round of matching, answer candidate "1954" get higher score because "the first Burger King" is more close to the question term "the first Burger King restaurant". So the final answer will be "1954".

6 Answering Definition Questions with Syntactic-Semantic Patterns

We applied syntactic patterns to answer "Other" questions. The patterns are learned from the previous TREC QA topics and answer sentences.

There are 65 topics in 2004 QA main task. We split the topics into two sets for training and testing. For each topic, the answer nuggets and a list of sentences containing answer nuggets (created by Ken Litkowski) are provided on TREC website.

The answer sentences are parsed and the parse trees are bottom-up traversed. At each level of the parse tree, the answer nuggets are compared and the syntactic patterns are extracted if the nuggets are matched. The syntactic patterns are scored according to the answer nuggets matched. If the "Vital" answer nuggets are matched, the syntactic patterns are assigned higher scores. After the pattern extraction, we found that some common syntactic patterns such "NP VP", "NP NP", "NP PP" get high scores. These common patterns extract useful information as well as a lot of irrelevant information. To address this problem, we append semantic features to the patterns. These semantic features include comparative adjectives, digits, topic related verbs and topic phrases.

These syntactic patterns with semantic features are applied to the test questions. The results are compared with the answer nuggets. The scores of the patterns are adjusted. The patterns that extract more "Vital" or "OK" information get higher scores and patterns which extract more irrelevant information get lower score or be removed. Finally we kept 34 patterns. Here are some sample patterns:

VBD PP PP_t PP_d
NP JJS NN NN_t
NP JJS NN NNS_t

When answer "Other" question, we select the top 7 documents retrieved from the "Factoid" and "List" questions in the topic series. The documents are split into sentences and filtered by topic key words.

The candidate sentences are parsed and the parse trees are traversed bottom-up to do pattern matching. Perfect match is not always guaranteed. The matching score is calculated according to how well the semantic features are matched. The final score is the product of the pattern score and matching score. Redundancy filtering removes the duplicate information nuggets and the information nuggets with length greater than 125 bytes are filtered out. Finally we chose the top 30 nuggets as the answers.

7 Experiments and Evaluation Results

We submitted one run this year and Table 1 shows the evaluation result.

Compared with last year's evaluation result, the system performance is lowered. Except the F-score of "List" questions improved by 1%, the accuracy of "Factoid" questions dropped nearly 4% and the F-score of "Other" questions dropped nearly 8%. We think the decreasing of "Factoid" questions accuracy was caused mainly by the more difficult questions and more strict evaluation standard because the number of "inexact" answers of this year witnessed an increasing of 25%. However, the obviously decreasing of F-score of "Other" questions shows the incompleteness caused by the syntactic-semantic patterns.

Table 1 ILQUA Evaluation Result

	Factoid	List	Other	Average Per Series
	(Accuracy Score)	(F Score)	(F Score)	
ILQUA-2006	0.266	0.129	0.128	0.171
Best-2006	0.578	0.433	0.250	0.394
Median-2006	0.186	0.087	0.125	0.134
ILQUA-2005	0.309	0.118	0.207	0.241
Best -2005	0.713	0.468	0.248	0.534
Median-2005	0.152	0.053	0.156	0.123

References

Chu-Carrol, J., J. Prager, C. Welty, K. Czuba and D. Ferrucci. “A Multi-Strategy and Multi-Source Approach to Question Answering”, In Proceedings of the 11th TREC, 2003.

Cui, H., K. Li, R. Sun, T.-S. Chua and M.-Y. Kan. “National University of Singapore at the TREC 13 Question Answering Main Task”. In Proceedings of the 13th TREC, 2005.

Echihabi, A., U.Hermjakob, E. Hovy, D. Marcu, E. Melz and D. Ravichandran. “Multiple-Engine Question Answering in TextMap”. In Proceedings of the 12th TREC, 2004.

Han, K.-S., H. Chung, S.-B. Kim, Y.-I. Song, J.-Y. Lee, and H.-C. Rim. “Korea University Question Answering System at TREC 2004”. In Proceedings of the 13th TREC, 2005.

Harabagiu, S., D. Moldovan, C. Clark, M. Bowden, J. Williams and J. Bensley. “Answer Mining by Combining Extraction Techniques with Abductive Reasoning”. In Proceedings of 12th TREC, 2004.

Hovy, E. L. Gerber, U. Hermjakob, M. Junk and C.-Y. Lin. “Question Answering in Webclopedia”. In Proceedings of the 9th TREC, 2001.

Katz, B. and J. Lin. “Selectively Using Relations to Improve Precision in Question Answering”. In Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering. April 2003.

Moldovan, D. and V. Rus. “Logical Form Transformation of WordNet and its Applicability to Question Answering”. In Proceedings of the ACL, 2001.

Prager, J., E. Brown, A. Coden and D. Radev. “Question-Answering by Predictive Annotation”. In Proceedings of SIGIR 2000, pp. 184-191. 2000.

Prager, J., J. Chu-Carroll and K. Czuba. “Question Answering Using Constraint Satisfaction: QA-By-Dossier-With-Constraints”. In Proceedings of the 42nd ACL. 2004.

Ravichandran, D. and E. Hovy. “Learning Surface Text Patterns for a Question Answering System”. In Proceedings of 40th ACL. 2002.

Soubbotin, M. and S. Soubbotin. “Patterns of Potential Answer Expressions as Clues to the Right Answers”. In Proceedings of 11th TREC. 2003.

Voorhees, E. “Using Question Series to Evaluate Question Answering System Effectiveness”. In Proceedings of HLT 2005. 2005.

Wu, M., M. Duan, S. Small and T. Strzalkowski, ILQUA – An IE-Driven Question Answering System. In Proceedings of TREC-14. 2006.

Wu, M. and T. Strzalkowski, Utilizing Co-occurrence of Answers in Question Answering. In Proceedings of COLING-ACL 2006. July 2006.