Learning Strategies for Story Comprehension: A Reinforcement Learning Approach

Eugene Grois E-GROIS@UIUC.EDU
David C. Wilkins DCW@UIUC.EDU
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA

Abstract

This paper describes the use of machine learning to improve the performance of natural language question answering systems. We present a model for improving story comprehension through inductive generalization and reinforcement learning, based on classified examples. In the process, the model selects the most relevant and useful pieces of lexical information to be used by the inference procedure. We compare our approach to three prior non-learning systems, and evaluate the conditions under which learning is effective. We demonstrate that a learning-based approach can improve upon "matching and extraction"-only techniques.

1. Introduction

This paper presents a model for the automated acquisition of behavior policies for tasks that are not readily cloned due to the difficulty of gathering explicit traces of successful execution. We apply our model to the task of automatically learning strategies for natural language question answering from examples composed of textual sources, questions, and answers. Our approach is focused on one specific type of text-based question answering known as *story comprehension*. Most TREC-style QA systems are designed to extract an answer from a document contained in a fairly large general collection [Voorhees, 2003]. Story comprehension requires a similar approach, but involves answering questions from a single narrative document. An important challenge in text-based question answering in general is posed by the syntactic and semantic variability of question and answer forms, which makes it difficult to establish a match between the question and answer candidate. This problem is particularly acute in the case of story comprehension due to the rarity of information restatement in the single document.

Several recent systems have specifically addressed the task of story comprehension. The Deep Read reading comprehension system [Hirschman *et al.*, 1999] uses a statistical bag-of-words approach, matching the question with the lexically most similar sentence in the story. Quarc [Riloff and Thelen, 2000] utilizes manually generated rules that select a sentence deemed to contain the answer based on a combination of syntactic similarity and semantic correspondence (i.e., semantic categories of nouns). The Brown University statistical language processing class project systems [Charniak, *et al.*, 2000] combine the use of manually generated rules with statistical techniques such as bag-of-words and bag-of-verb matching, as well as deeper semantic analysis of nouns. As a rule, these three systems are effective at identifying the sentence containing the correct answer as long as the answer is explicit and contained entirely in that sentence. They find it difficult, however, to deal with semantic alternations of even moderate complexity. They also do not address situations where answers are split across multiple sentences, or those requiring complex inference.

Statistical learning techniques have been applied to the task of finding answers to questions. For example, [Berger *et al.*, 2000] induces questions-answer correspondence from FAQs and call center data. These techniques perform better in domain-specific applications than in general-purpose ones. Also, they match explicit and complete answers from a list to a given question. They do not perform translation of text to generate non-explicit or composite answers.

Our framework, called QABLe (Question-Answering Behavior Learner), draws on prior work in learning action and problem-solving strategies [Tadepalli and Natarajan, 1996; Khardon, 1999]. We represent textual sources as sets of features in a sparse domain, and treat the QA task as behavior in a stochastic, partially observable world. QA strategies are learned as sequences of transformation rules capable of deriving certain types of answers from

Appearing in the proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005.

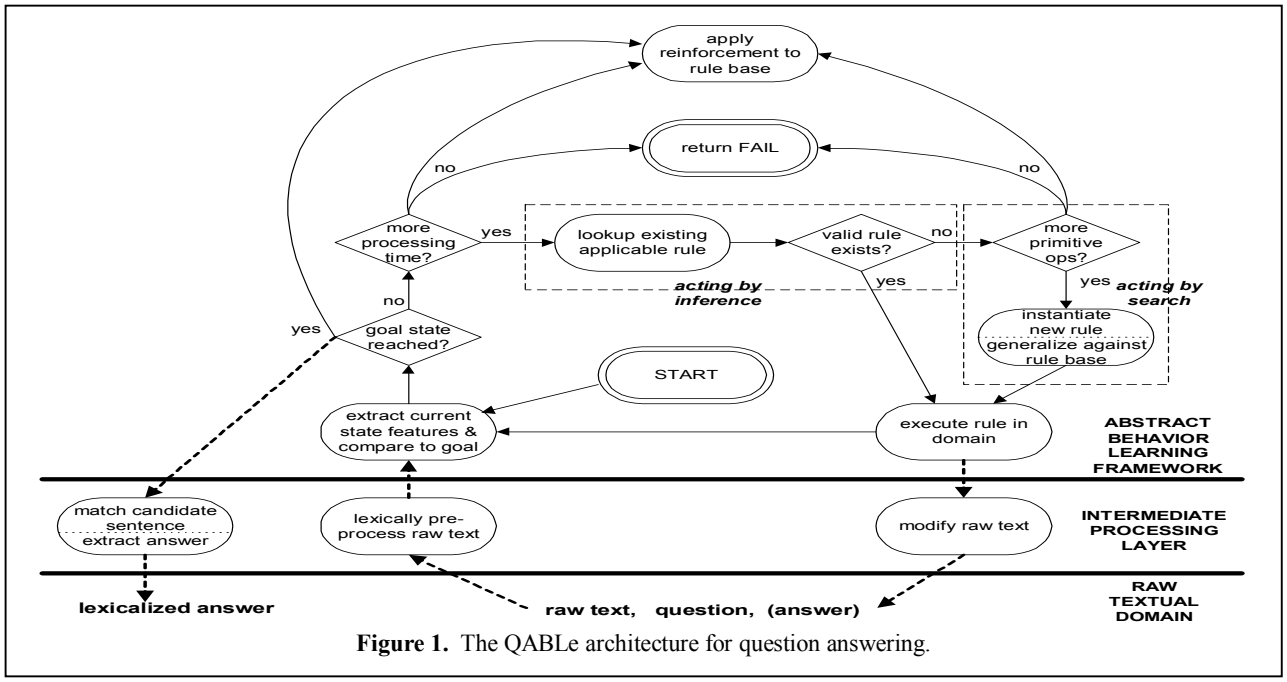

Figure 1. The QABLe architecture for question answering.

particular text-question combinations. Strategies determine which transformation rules to apply when. The transformation rules are generated by instantiating primitive domain operators in specific feature contexts. A process of reinforcement learning [Kaebling, *et al.*, 1996] is used to select and promote effective transformation rules. Representations of the underlying domain features are learned in the course of interacting with the domain, and encode the features at the levels of abstraction that are found to be conducive to successful behavior. This selection effect is achieved by a fusion of abstraction space generalization [Knoblock, 1992] and reinforcement learning elements.

Our approach is similar to a technique known as behavior cloning, in which control/action rules are induced from traces generated by a "teacher" [Sammut, et al., 1992; Bratko, et al., 1998]. Behavior cloning has been applied successfully in various control and planning applications, where traces of explicit actions or solution steps are easily obtainable. Question-answering, however, is not an explicitly traceable task. The exact process by which a human analyzes and answers a question, and the internal representations used to facilitate that process, are in large part hidden and not amenable to explicit modeling techniques. Therefore, our goal for QABLe is not to clone human QA behavior, but rather to independently learn rules and behavior policies composed of these rules that produce results similar to those of the human teacher.

Conceptually, our approach resembles the learning of reactive action models [Benson, 1995], wherein decision lists of teleo-operators [Benson and Nilsson, 1995] are induced through experimentation in the domain. Our approach is also similar in spirit to the noisy channel-based translation of strings proposed by [Marcu and Popescu, 2005].

The rest of this paper is organized as follows. Section 2 presents the details of the QABLe framework. In section 3 we describe preliminary experimental results which indicate promise for our approach. In section 4 we summarize and draw conclusions.

2. QABLe – Learning to Answer Questions

2.1 Overview

Figure 1 shows a diagram of the QABLe framework. The bottom-most layer is the natural language textual domain. It represents raw textual sources, questions, and answers. The intermediate layer consists of processing modules that translate between the raw textual domain and the top-most layer, an abstract representation used to reason and learn.

This framework is used both for learning to answer questions and for the actual QA task. While learning, the system is provided with a set of training instances, each consisting of a textual narrative, a question, and a corresponding answer. During the performance phase, only the narrative and question are given.

At the lexical level, an answer to a question is generated by applying a series of *transformation rules* to the text of the narrative. These transformation rules augment the original text with one or more additional sentences, such that one of these explicitly contains the answer, *and* matches the form of the question.

On the abstract level, this is essentially a process of searching for a path through problem space that transforms the world state, as described by the textual source and question, into a world state containing an appropriate answer. This process is made efficient by learning answer-generation strategies. These strategies store procedural knowledge regarding the way in which answers are derived from text, and suggest appropriate transformation rules at each step in the answer-generation process. Strategies (and the procedural knowledge stored therein) are acquired by explaining (or deducing) correct answers from training examples. The framework's ability to answer questions is tested only with respect to the kinds of documents it has seen during training, the kinds of questions it has practiced answering, and its interface to the world (domain sensors and operators).

In the next two sections we discuss lexical pre-processing, and the representation of features and relations over them in the QABLe framework. In section 2.4 we look at the structure of transformation rules and describe how they are instantiated. In section 2.5, we build on this information and describe details of how strategies are learned and utilized to generate answers. In section 2.7 we explain how candidate answers are matched to the question, and extracted.

2.2 Lexical Pre-Processing

Several levels of syntactic and semantic processing are required in order to generate structures that facilitate higher order analysis. We currently use MontyTagger 1.2, an off-the-shelf POS tagger based on [Brill, 1995], for POS tagging. At the next tier, we utilize a Named Entity (NE) tagger for proper nouns a semantic category classifier for nouns and noun phrases, and a co-reference resolver (that is limited to pronominal anaphora). Our taxonomy of semantic categories is derived from the list of unique beginners for WordNet nouns [Fellbaum, 1998]. We also have a parallel stage that identifies phrase types, a form of structural relation. Table 1 gives a list of phrase types currently in use, together with the categories of questions each phrase type can answer. In the near future, we plan to utilize a link parser to boost phrase-type tagging accuracy. We also would like to extract deeper semantic relations. For questions, we have a classifier that identifies the semantic category of information requested by the question. Currently, this taxonomy is identical to that of semantic categories. However, in the future, it may be expanded to accommodate a wider range of queries. A separate module reformulates questions into statement form for later matching with answer-containing phrases.

Phrase Type	Comments
SUBJ	"Who" and nominal "What"
VERB	event "What"
DIR-OBJ	"Who" and nominal "What"
INDIR-OBJ	"Who" and nominal "What"
ELAB-SUBJ	descriptive "What" (eg. what kind)
ELAB-VERB-TIME	time
ELAB-VERB-PLACE	place
ELAB-VERB-MANNER	manner
ELAB-VERB-CAUSE	"Why"
ELAB-VERB-INTENTION	"Why" as well as "What for"
ELAB-VERB-OTHER	smooth handling of undefined verb phrase types
ELAB-DIR-OBJ	descriptive "What" (eg. what kind)
ELAB-INDIR-OBJ	descriptive "What" (eg. what kind)
VERB-COMPL	WHERE/WHEN/HOW concerning state or status

Table 1. Phrase types used by QABLe framework.

2.3 Representing the Question-Answering Domain

In this section we explain how features are extracted from raw textual input and tags which are generated by pre-processing modules.

A sentence is represented as a sequence of words $\langle w_1, w_2, \ldots, w_n \rangle$, where $word(w_i, \text{word})$ binds a particular word to its position in the sentence. The k^{th} sentence in a passage is given a unique designation s_k. Several simple functions capture the syntax of the sentence. The sentence *Main* (e.g., main verb) is the controlling element of the sentence, and is recognized by $main(w_m, s_k)$. Parts of speech are recognized by the function *pos*, as in $pos(w_i, \text{NN})$ and $pos(w_i, \text{VBD})$. The relative syntactic ordering of words is captured by the predicate $before(w_i, w_j)$. It can be applied recursively, as $before(w_i, before(w_j, w_k))$ to generate the entire sentence starting with an arbitrary word, usually the sentence *Main*. The integrity of the sentence is checked by the function $inSentence(w_i, s_i) \Rightarrow main(w_m, s_k) \wedge (before(w_i, w_m) \vee before(w_m, w_i))$ for each word w_i in the sentence. A consecutive sequence of words is a *phrase entity* or simply *entity*. It is given the designation e_x and declared by a binding function, such as $entity(e_x, NE)$ for a named entity, and $entity(e_x, NP)$ for a syntactic group of type *noun phrase*. Each phrase entity is identified by its *head*, as $head(w_h, e_x)$, and we say that the phrase head controls the entity. A phrase entity is defined as $head(w_h, e_x) \wedge inPhrase(w_i, e_x) \wedge \ldots \wedge inPhrase(w_j, e_x)$.

We also wish to represent higher-order relations such as functional roles and semantic categories. Functional dependency between pairs of words is encoded as, for example, $subj(w_i, w_j)$ and $aux(w_j, w_k)$. Functional groups are represented just like phrase entities. Each is assigned a designation r_x, declared for example, as $func\_role(r_x, SUBJ)$, and defined in terms of its head and members (which may be individual words or composite entities). Semantic categories are similarly defined over the set of words and syntactic phrase entities – for example,

sem_cat(c_x, *PERSON*) ∧ *head*(w_h, c_x) ∧ *pos*(w_i, NNP) ∧ *word*(w_h, "John").

Semantically, sentences are treated as events defined by their verbs. A multi-sentential passage is represented by tying the member sentences together with relations over their verbs. We declare two such relations – *seq* and *cause*. The *seq* relation between two sentences, *seq*(s_i, s_j) ⇒ *prior*(*main*(s_i), *main*(s_j)), is defined as the sequential ordering in time of the corresponding events. The *cause* relation *cause*(s_i, s_j) ⇒ *cdep*(*main*(s_i), *main*(s_j)) is defined such that the second event is causally dependent on the first.

2.4 Primitive Operators and Transformation Rules

The system, in general, starts out with no procedural knowledge of the domain (*i.e.*, no transformation rules). However, it is equipped with 9 primitive operators that define basic actions in the domain. Primitive operators are existentially quantified. They have no activation condition, but only an *existence condition* – the minimal binding condition for the operator to be applicable in a given state. A primitive operator has the form $C^E \rightarrow \hat{A}$, where C^E is the existence condition and $\hat{A}$ is an action implemented in the domain. An example primitive operator is

primitive-op-1 : $\exists\, w_x, w_y \rightarrow$ add-word-after(w_y, w_x)

Other primitive operators delete words or manipulate entire phrases. Note that primitive operators act directly on the syntax of the domain. In particular, they manipulate words and phrases. A primitive operator bound to a state in the domain constitutes a transformation rule. The procedure for instantiating transformation rules using primitive operators is given in Figure 2. The result of this procedure is a universally quantified rule having the form $C \wedge G^R \rightarrow A$. A may represent either the name of an action in the world or an internal predicate. C represents the necessary condition for rule activation in the form of a conjunction over the *relevant* attributes of the world state. G^R represents the expected effect of the action. For example,

$$before(w_1, w_2) \wedge before(w_2, w_3) \wedge G^R(before(w_3, w_4)) \rightarrow add-word-after(w_4, w_3)$$

indicates that when the phrase "w_1 w_2 w_3" is found in the text, this operator is expected to attach w_4 to the end, generating the phrase "w_1 w_2 w_3 w_4".

An instantiated rule is assigned a *rank* composed of:

- priority rating (p)
- level of experience with rule (f)
- confidence in current parameter bindings (c)

The first component, priority rating, is an inductively acquired measure of the rule's performance on previous instances. The second component modulates the priority rating with respects to a frequency of use measure. The third component captures any uncertainty inherent in the underlying features serving as parameters to the rule. The rank of a rule is computed by the following function:

$$rank = p \times c \times (1 + \log(1 + f))$$

Each time a new rule is added to the rule base, an attempt is made to combine it with similar existing rules to produce more general rules having a wider relevance and applicability.

Given a rule $c_a \wedge c_b \wedge g_x^R \wedge g_y^R \rightarrow A_1$ covering a set of example instances E_1 and another rule $c_b \wedge c_c \wedge g_y^R \wedge g_z^R \rightarrow A_2$ covering a set of examples E_2, we add a more general rule $c_b \wedge g_y^R \rightarrow A_3$ to the strategy. The new rule A_3 is consistent with E_1 and E_2. In addition it will bind to any state where the literal c_b is active. Therefore the hypothesis represented by the triggering condition is likely an overgeneralization of the target concept. This means that rule A_3 may bind in some states erroneously. However, since all rules that can bind in a state compete to fire in that state, if there is a better rule, then A_3 will be preempted and will not fire. Figure 3 gives the rule generalization algorithm.

Instantiate Rule
Given:
- set of primitive operators
- current state specification
- goal specification

1. select primitive operator to instantiate
2. bind active state variables & goal spec to existentially quantified condition variables
3. execute action in domain
4. update expected effect of new rule according to change in state variable values

Figure 2. Procedure for instantiating transformation rules using primitive operators.

2.5 Generating Answers

Returning to Figure 1, we note that at the abstract level the process of answer generation begins with the extraction of features active in the current state. These features represent low-level textual attributes and the relations over them described in section 2.3.

Immediately upon reading the current state, the system checks to see if this is a *goal state*. A goal state is a state whose corresponding textual domain representation contains an explicit answer in the right form to match the questions. In the abstract representation, we say that in this state all of the goal constraints are satisfied.

Generalize
Given:
1. rule $R_1 \leftarrow C_1 \wedge G_1^R$, priority rating $pr(R_1)$
2. rule $R_2 \leftarrow C_2 \wedge G_2^R$, priority rating $pr(R_2)$

Generate a new rule R', such that $R_1, R_2 \subseteq R'$.

if $R_1 \subseteq R_2$ **then**
$R' = R_2$, and $pr(R') = \max(pr(R_1), pr(R_2))$

else if $C_1 \cap C_2 \neq \varnothing$ and $G_1^R \cap G_2^R \neq \varnothing$ **then**
$R' \leftarrow (C_1 \cap C_2) \wedge (G_1^R \cap G_2^R)$, and
$pr(R') = \max(pr(R_1), pr(R_2))$

Figure 3. Procedure for generalizing a pair of rules.

If the current state is indeed a goal state, no further inference is required. The inference process terminates and the actual answer is identified by the matching technique described in section 2.6 and extracted.

If the current state is not a goal state and more processing time is available, QABLe passes the state to the Inference Engine (IE). This module stores strategies in the form of decision lists of rules. For a given state, each strategy may recommend at most one rule to execute. For each strategy this is the first rule in its decision list to fire. The IE selects the rule among these with the highest relative rank, and recommends it as the next transformation rule to be applied to the current state.

If a valid rule exists it is executed in the domain. This modifies the concrete textual layer. At this point, the pre-processing and feature extraction stages are invoked, a new current state is produced, and the inference cycle begins anew.

If a valid rule cannot be recommend by the IE, QABLe passes the current state to the Search Engine (SE). The SE uses the current state and its set of primitive operators to instantiate a new rule, as described in section 2.4. This rule is then executed in the domain, and another iteration of the process begins.

If no more primitive operators remain to be applied to the current state, the SE cannot instantiate a new rule. At this point, search for the goal state cannot proceed, processing terminates, and QABLe returns failure.

When the system is in the training phase and the SE instantiates a new rule, that rule is generalized against the existing rule base. This procedure attempts to create more general rules that can be applied to unseen example instances.

Once the inference/search process terminates (successfully or not), a reinforcement learning algorithm is applied to all rules that were active in that session. This reinforcement affects the *priority rating* (*p*) (see sect. 2.4) of each rule that fired in the current run. Specifically, rules on the solution path receive positive reward, and rules that fired, but are not on the solution path receive negative reinforcement.

A new rule is initially given a nominal priority rating, p^0. In generalizing, a newly created abstract rule is assigned a priority rating slightly higher than those of its more specific precursor rules. This guarantees that the abstract rule will have an opportunity to be tested.

Reinforcement is applied as $p' = p + R$. The reward R is a constant value, and $|R| < p^0$. Once a rule's priority rating drops below a threshold *t*, the rule is effectively dropped and never used again. In this way, rules that consistently perform poorly are quickly discarded.

2.7 Candidate Answer Matching and Extraction

As discussed in the previous section, when a goal state is generated in the abstract representation, this corresponds to a textual domain representation that contains an explicit answer in the right form to match the questions. Such a candidate answer may be present in the original text, or may be generated by the inference/search process. In either case, the answer-containing sentence must be found, and the actual answer extracted. This is accomplished by the Answer Matching and Extraction procedure.

The first step in this procedure is to reformulate the question into a statement form. This results in a sentence containing an empty slot for the information being queried. For example, "How far is the drive to Chicago?" becomes "The distance of the drive to Chicago is _____." Recall further that QABLe's pre-processing stage analyzes text with respect to various syntactic and semantic types. In addition to supporting abstract feature generation, these tags can be used to analyze text on a lexical level. Thus, the question above is marked up as

```
[ELAB-VERB <quantity-distance> (WRB How) (RB
far)] [VERB (VBZ is)] [SUBJ <action> (DT the)
(NN drive)] [VERB-COMPL (TO to) (NNP <place>
Chicago)].
```

Once reformulated into statement form, this becomes

```
[ELAB-VERB <quantity-distance> _____ ] [VERB
(VBZ is)] [SUBJ <action> (DT the) (NN drive)]
[VERB-COMPL (TO to) (NNP <place> Chicago)].
```

The goal now is to find a sentence who's syntactic and semantic analysis matches that of the reformulated question's as closely as possible. Thus, for example the text may contain the sentence "The drive to Chicago is 2 hours" with the corresponding analysis

```
[SUBJ <action> (DT the) (NN drive)] [VERB-
COMPL (TO to) (NNP <place> Chicago)] [VERB
(VBZ is)] [VERB-COMPL <quantity-time> (CD 2)
(NNS hours)].
```

Notice that all of the elements of this candidate answer match the corresponding elements of the question, with the exception of the semantic category of the ELAB-VERB phrase. This is likely not the answer we are seeking. The text may contain a second sentence "The drive to Chicago is 130 miles", that analyses as

```
[SUBJ <action> (DT the) (NN drive)] [VERB-
COMPL (TO to) (NNP <place> Chicago)] [VERB
(VBZ is)] [VERB-COMPL <quantity-distance> (CD
130) (NNS miles)].
```

In this case, all of the elements match their counterparts in the reformulated question. Thus, the second sentence can be matched as the correct answer with high confidence.

2.8 Examples

In this section illustrate the application of two general phrase-based rules learned during training on the Remedia corpus. In these examples the rules are applied to previously unseen instances from the same corpus.

Example 1

Rule #112:

$$phrase(x) \wedge phrase(y) \wedge phrase(z) \wedge cause(phrase(z), phrase(y)) \wedge (phrase(y) \rightarrow phrase(x) \Rightarrow cause(phrase(z), phrase(x))$$

The (rm4-2) Remedia story contains the question

```
Why did Dr. Barry keep the secret?
```

Rule #112 was used to generate an answer from the sentence

```
It seems Dr. Barry hid the truth so she
could practice her love of medicine.
```

The phrase-level tagged representations are

```
Q:  [SUBJ Dr. Barry] [VERB kept the secret] <---
CAUSE---- [PHRASE __________]

S:  [SUBJ Dr. Barry] [VERB hid the truth]
<---CAUSE--- [ELAB-VERB she could practice her
love of medicine]
```

Based on the verbal implication

```
"keep secret" ----> "hide"
```

the transformation rule

```
[VERB hid the truth] ----> [VERB kept the
secret](conf. 0.68)
```

is activated. Combined with the sentence above through rule #112 it generates the answer:

```
A:  [SUBJ Dr. Barry] [VERB kept the secret] <---
CAUSE--- [ELAB-VERB she could practice her love
of medicine]
```

Example 2

Rule #39:

$$s1\begin{pmatrix}[\text{SUBJ a}][\text{VERB b}][\text{DIR - OBJ c}][\text{INDIR - OBJ d}]\\ [\text{ELAB - VERB - TIME e}]\end{pmatrix} \wedge s2([\text{SUBJ d}][\text{VERB f}]) \wedge seq(s1, s2) \Rightarrow s3([\text{SUBJ d}][\text{VERB f}][\text{ELAB - VERB - TIME e}])$$

The (rm3-42) Remedia story contains the question

```
When did the circus almost close?
```

Rule #39 was used to generate an answer from two sentences:

```
In 1938, not many people had money to go
to the circus.  So most of them closed.
```

The phrase-level tagged representations are

```
Q:  [SUBJ The circus] [VERB almost closed] [ELAB-
VERB-TIME _________]

S:  [SUBJ Not many people] [VERB had] [DIR-OBJ
money to go to] [INDIR-OBJ the circus] [ELAB-
VERB-TIME in 1938]
---SEQ---> [SUBJ Most of them] [VERB closed]
```

The state represented by S activates rule #39 and the answer is generated:

```
A:  [SUBJ The circus] [VERB almost closed] [ELAB-
VERB-TIME in 1938]
```

3. Experimental Evaluation

3.1 Experimental Setup

We evaluate our approach to open-domain natural language question answering on the Remedia corpus. This is a collection of 115 children's stories provided by Remedia Publications for reading comprehension. The comprehension of each story is tested by answering five *who*, *what*, *where*, and *why* questions.

The Remedia Corpus was initially used to evaluate the Deep Read reading comprehension system, and later also other systems, including Quarc and the Brown University statistical language processing class project.

The corpus includes two answer keys. The first answer key contains annotations indicating the story sentence that is lexically closest to the answer found in the published answer key (AutSent). The second answer key contains sentences that a human judged to best answer each question (HumSent). Examination of the two keys shows the latter to be more reliable. We trained and tested using the HumSent answers. We also compare our results to the HumSent results of prior systems. In the Remedia corpus, approximately 10% of the questions lack an answer. Following prior work, only questions with annotated answers were considered.

System	who	what	when	where	why	Overall
Deep Read	48%	38%	37%	39%	21%	36%
Quarc	41%	28%	55%	47%	28%	40%
Brown	57%	32%	32%	50%	22%	41%
QABLe-N/L-N/PR	48%	35%	51%	42%	28%	41%
QABLe-N/L-PR	48%	35%	52%	43%	28%	41%
QABLe-L-N/PR	55%	41%	55%	44%	29%	45%
QABLe-L-PR	56%	41%	56%	45%	35%	47%
QABLe-L+-N/PR	55%	41%	56%	45%	30%	45%
QABLe-L+-PR	59%	43%	56%	46%	36%	48%

Table 2. Comparison of QA accuracy by question type.

System	tot. # rules learned	tot. # rules on all solution paths	avrg. # rules on solution path per correct answer
QABLe-L-N/PR	11,082	1193	9.36
QABLe-L-PR	3,463	426	3.02
QABLe-L+-N/PR	60,058	1274	9.23
QABLe-L+-PR	16,681	411	2.85

Table 3. Analysis of transformation rule learning and use.

We divided the Remedia corpus into a set of 55 tests used for development, and 60 tests used to evaluate our model, employing the same partition scheme as followed by the prior work mentioned above. With five questions being supplied with each test, this breakdown provided 275 example instances for training, and 300 example instances to test with. However, due to the heavy reliance of our model on learning, many more training examples were necessary. We widened the training set by adding story-question-answer sets obtained from several online sources. With the extended corpus, QABLe was trained on 262 stories with 3-5 questions each, corresponding to 1000 example instances.

3.2 Discussion of Results

Table 2 compares the performance of different versions of QABLe with those reported by the three prior systems described above. We wish to evaluate three aspects of the QABLe framework:

- the particular contribution of transformation rule learning in the QABLe model
- the value of expanding the training set
- the value of abstracting the feature space through use of higher-order relational representations.

To this end, we compare the respective accuracies of answers returned by six versions of QABLe, characterized by two key parameters. One parameter varies the amount of learning – none: QA matching and extraction algorithm described in section 2.6 only (-N/L), learning with the Remedia training corpus only (-L), and learning with the expanded training corpus described above (-L+). The second parameter selects the presence (-PR) or absence (-N/PR) of phrase role tags. These correspond to the phrase types in Table 1, and represent higher-order structural/semantic relations over the raw text.

As expected, the accuracies of the no-learning versions are comparable to those of the earlier systems. The Remedia-only training set versions show a noticeable improvement over the no-learning (baseline) QABLe, and most of the prior system results. This is mostly due to an expanded ability to deal with semantic alternations in the narrative by learning transformation rules that reformulate the alternations into lexical form matching that of the question.

Table 3 gives a break-down of rule learning and use for the learning versions of QABLe during the course of an entire training/test cycle. The first column is the total number of rules learned by each system version. The second column is the total number of distinct rules that ended up being successfully used in generating all correct answers. The last column gives the average number of rules each system needed to produce an answer (where a correct answer was generated). As expected, versions augmented with phrase role tags executed far fewer rules than those without. This is because phrase role tags permit entire phrases to be treated as semantic entities, and to be used as arguments to transformations. More importantly, note that the QABLe-L+ versions used fewer rules on average to generate more correct answers than the QABLe-L versions. This is because QABLe-L+ versions had more opportunities to refine their policies controlling rule firing through reinforcement and generalization.

Notice, however, that the QABLe-L+ versions were trained on a corpus more than three time the size of that for QABLe-L. The overall improvement in accuracy bought by the substantial effort of generating the extra

training instances was negligible - 1% for QABLe-L+-PR, and practically no improvement at all for QABLe-L+-N/PR. Training on more examples certainly leads to wider domain coverage through the acquisition of more transformation rules. However, the variation among examples is so large as to make good domain coverage through training corpus expansion impractical.

At the same time, note that the use of phrase role tagging lead to improved accuracy with the standard Remedia training corpus alone, and even better results with the expanded training corpus. The reason for this is that phrase role entities support a more expressive representation, and thus permit better rule generalization from fewer training examples.

In summary then, a QA approach based on strategies consisting of learned transformation rules is clearly superior to "matching and extraction"-only techniques. Furthermore, the results of Table 3 indicate that an expanded training corpus offers more opportunities to refine the policy controlling rule firing through reinforcement and generalization. However, as can be seen from Table 2, this leads to only marginal improvement in coverage across the domain. Furthermore, expansion of the training corpus boosts accuracy only in combination with a sufficiently expressive representation of the domain. This is because an expressive representation supports more effective rule generalization.

We anticipate that a gradual improvement in the depth and accuracy of semantic and pragmatic pre-processing will permit a vastly richer and more compact representation of narrative text and questions, which will lead to dramatic boost in the accuracy of answers generated by the QABLe framework.

4. Conclusion

We apply our model to the NLP task of story comprehension and describe QABLe, a framework for learning strategies for question answering from examples composed of textual narratives, questions, and answers. These strategies are composed of ranked lists of transformation rules that when applied to an initial state consisting of an unseen text and question, can derive the required answer. The strategies are acquired through inductive generalization and reinforcement learning. In the process, the most relevant pieces of lexical information are selected. This approach was evaluated on the Remedia corpus and compared with three non-learning systems. QABLe was found to significantly improve upon non-learning techniques.

Acknowledgements

We gratefully acknowledge the helpful feedback provided by Dan Roth. This research was supported, in part, by ONR MURI grant N00014-00-1-0660.

References

[Benson, 1995] S. Benson. Inductive learning of reactive action models. *ICML-95*, 1995.

[Benson and Nilsson, 1995] S. Benson and N. Nilsson. Reacting, planning, and learning in an autonomous agent. In K. Furukawa, D. Michie, and S. Muggleton, eds. *Machine Intelligence 14*, The Claredon Press.

[Berger *et al*., 2000] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: Statistical approaches to answer-finding. *ACM SIGIR-00*, 2000.

[Bratko, et al., 1998] I. Bratko, T. Urbancic, and C. Sammut. Behavior cloning: phenomena, results and problems: Automated systems based on human skill. *IFAC Symposium*. Berlin.

[Brill, 1995] E. Brill. Transformation-based error driven learning and natural language processing: A case study in part of speech tagging. In *Computational Linguistics*, 21(4):543-565, 1995.

[Charniak, *et al.*, 2000] E. Charniak, Y. Altun, R. de Salvo Braz, B. Garrett, M. Kosmala, T. Moscovich, L. Pang, C. Pyo, Y. Sun, W. Wy, Z. Yang, S. Zeller, and L. Zorn. Reading comprehension programs in a statistical-language-processing class. *ANLP/NAACL-00*, 2000.

[Fellbaum, 1998] C. Fellbaum (ed.) *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[Hirschman *et al.*, 1999] L. Hirschman, M. Light, and J. Burger. Deep Read: A reading comprehension system. *ACL-99*, 1999.

[Kaebling, *et al.*, 1996] L. P. Kaebling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *J. Artif. Intel. Research*, 4:237-285, 1996.

[Khardon, 1999] R. Khardon. Learning to take action. *Machine Learning* 35(1), 1999.

[Knoblock, 1992] C. Knoblock. Automatic generation of abstraction for planning. *Artificial Intelligence*, 68(2):243-302, 1992.

[Marcu and Popescu, 2005] D. Marcu and A.M. Popescu. Towards developing probabilistic generative models for reasoning with natural language representations. *ICCLTP-05*, 2005.

[Riloff and Thelen, 2000] E. Riloff and M. Thelen. A rule-based question answering system for reading comprehension tests. *ANLP/NAACL-2000*, 2000.

[Sammut, *et al*., 1992] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. *ICML-98*, 1998.

[Tadepalli and Natarajan, 1996] P. Tadepalli and B. Natarajan. A formal framework for speedup learning from problems and solutions. *J. Artif. Intel. Research*, 4:445-475, 1996.

[Voorhees, 2003] E. M. Voorhees Overview of the TREC 2003 question answering track. *TREC-12*, 2003.