The JAVELIN Question-Answering System at TREC 2003: A Multi-Strategy Approach with Dynamic Planning

E. Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupść, L. V. Lita, V. Pedro, D. Svoboda and B. Van Durme

Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Abstract

The JAVELIN system evaluated at TREC 2003 is an integrated architecture for open-domain question answering. JAVELIN employs a modular approach that addresses individual aspects of the QA task in an abstract manner. The System implements a planner that controls the execution and information flow, as well as a multiple answer seeking strategies used differently depending on the type of question.

1 System Overview

JAVELIN is an object-oriented architecture that separates the processing details of individual operations (e.g. taggers, parsers) from the contexts in which they are used. JAVELIN is a flexible and extensible platform which supports component-level evaluation, so that different strategies can be tested individually and then integrated into the system in a straightforward manner.

The current system (Figure 1) brings together a large number of modular components that perform various question-answering tasks, such as: question analysis, document and passage retrieval, answer candidate extraction, answer selection, answer justification, and planning. The Planner module uses abstract interface definitions for each component to select and order the execution of individual components, allowing the system to dynamically generate multiple processing strategies and replan when necessary.

The details of execution are handled by the Execution Manager, a component which uses a Data Repository for storing session data (process steps, intermediate and final results). JAVELIN incorporates a user interaction component (GUI) for question input and user clarification. The system also includes an Answer Justification module, which provides a browsable view of the process history, intermediate data structures and final results.

1.1 Question Analysis

The Question Analyzer (QA) produces a programmatic representation of each input question for use by the rest of the JAVELIN system. This representation (referred to as a Request Object or RO) contains three major features: the answer type, keyword sets, and a logical representation of the question. Although many systems generate the first two features through surface analysis of the input question, JAVELIN also attempts to apply natural language processing to obtain a logical representation of the question. The logical representation attempts to capture the semantics (meaning) of the question; a similar representation has been reported for question node templates in (Harabagiu et al., 2000).

In the QA module, natural language processing occurs in two steps: lexical and syntactic parsing. Since the inputs to be analyzed are open-domain questions, most specific dictionaries provide inadequate coverage. Instead, we use several external tools designed for open-domain text:

- Brill tagger (Brill, 1995) for part-of-speech tagging;
- BBN IdentiFinder (Bikel et al., 1999) for named entity tagging (people, organizations, locations, etc.);
- WordNet (Fellbaum, 1998) for use in finding hypernym relationships for semantic categorization and for extracting the root form of a word;
- KANTOO Lexifier (Nyberg and Mitamura, 2000) for finding verb valencies.

The QA module uses the KANTOO parser (Nyberg and Mitamura, 2000) and hand-built grammar for syntactic analysis. The module extract semantic information produced by KANTOO's lexical processing rules, and specific rule-based patterns are inserted as needed. Specialized grammar rules are used on a per-answer-type basis to recognize syntactic and semantic similarities among questions. The grammars of fundamental

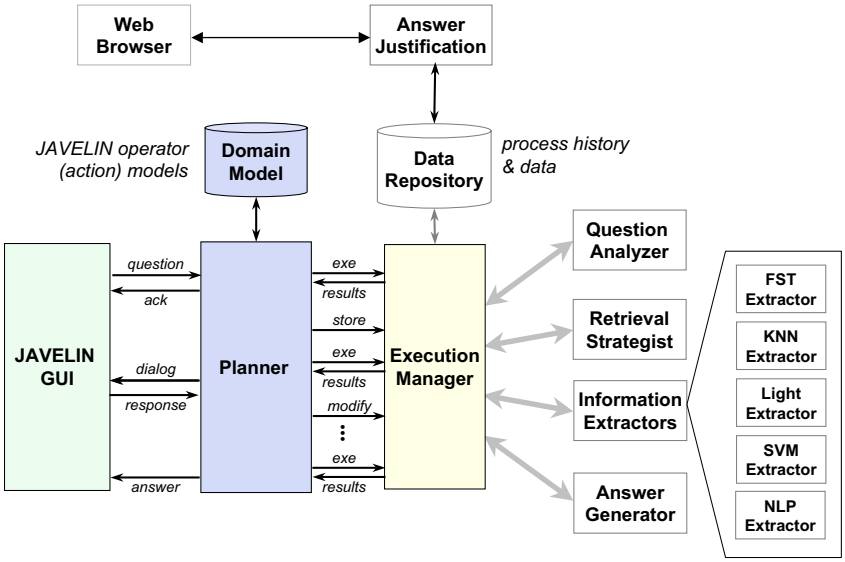

Figure 1: The JAVELIN architecture. The Planner operates as a service for the user interface and controls execution of the individual components via the Execution Manager. The Planner selects from a set of 5 different extraction modules according to its domain model of the QA process.

constituents are modularized (e.g. NPs and VPs) to support straightforward combination of different constituent-level grammars into sentence-level grammars.

1.2 Retrieval Strategist

The main purpose of the Retrieval Strategist (RS) module is to retrieve likely answer documents from the document repository in response to a query. The RS module also acts as a document server for other parts of JAVELIN which trigger lookup requests for specific documents or passages. The RS module operates on the Request Object produced by the Question Analyzer. The salient subparts of the Request Object are a) the set of keywords produced for the question, and b) a set of constraints associated with the retrieval process.

The keywords are words or phrases which the QA module has deemed likely to be present in the answer. Each keyword is one of three types: a) a single word; b) a short phrase, e.g. 'electoral college'; c) a proper name, e.g. 'Elvis Presley'. No query expansion is currently performed by the RS module on the keywords. However, the QA module may specify a set of alternates for any given keyword, which are treated as synonyms for retrieval.

The set of constraints present in the Request Object includes:

- upper and lower limits on the number of ranked documents to be retrieved;
- the likely answer type;
- one or more subtypes for the given answer type;
- the total time allowed for processing.

Since TREC 2002, we have switched from using the proprietary Inquery retrieval system to the Lemur 2.0 toolkit (Oglivie and Callan, 2002). Lemur is open-source and supports a variety of retrieval models, including support for Inquery-style structured queries.

Stemming is done at indexing time using the Lemur Porter stemmer. Before indexing, our source documents are preprocessed with the BBN Identifinder (v1.7) named entity tagger (Bikel et al., 1999) to identify named entities such as 'Organization', 'Time', 'Date', 'Person', 'Place Name', 'Currency Amount', 'Number', 'Percentage', and object types such as 'Animal', 'Plant', 'Product', 'Game', etc. This analysis attempts to focus retrieval on documents containing not only the relevant keywords, but also relevant data types. At indexing time, any terms within a span of text identified as a named entity are stored in the index using a set of corresponding special fields.

To process a query at runtime, if a likely answer type is specified, it is mapped to a set of named entity types. For example, an answer expected to be of 'temporal' type might map to either a 'Time' or 'Date' named entity field. These named entity fields are treated as special 'keywords' to be included in the terms passed to Lemur.

The current search algorithm is similar to the algorithm used in TREC 2002, and proceeds using an incremental query relaxation technique; starting from an initial query that is highly constrained, the algorithm searches for all the keyword terms and data types in close proximity to

each other. However, last year's algorithm (TREC 2002) always included all keywords at every relaxation step, while this year's algorithm attempts to be more flexible by assigning a priority to each keyword. This priority is based on a function of the likely answer type, keyword type (word, proper name, or phrase) and the inverse document frequency (idf) value of the keyword term(s). Keywords deemed more likely to exist near the answer are given higher priority. At each iteration, a priority threshold is adjusted which may result in lower-priority keywords being discarded from the query.

As before, at each relaxation step, the algorithm also relaxes other parameters in the query such as the word proximity window. This assumes that more likely answer documents will have clusters of relevant question keywords and data types in closer proximity. Another new feature of this year's algorithm is the use of a hybrid IR model in which earlier, more constrained query steps use the structured query retrieval model, while later steps switch to a tf.idf model. The algorithm terminates once the requested document list is full, or there are no more relaxation steps possible. The complete set of relaxation parameters includes:

1. The Inquery-style proximity/belief operator used to combine keywords. At each relaxation step, we either keep the same operator but expand the window size, or start with a new, more general operator. The operator applies to all keywords given in the query. For example, initially all keywords must be found within a proximity of three words. We then relax the operator to consider unordered 20-, 100-, and 250-word windows, followed by document-wide probabilistic AND, and so on.

2. Phrase proximity, for any phrase keywords. This is usually kept at 3 words or less, until later in the relaxation regime, when the window is slightly expanded.

3. Proper name proximity, for any proper name keywords. Like phrase keywords, this is usually kept at 3 words or less until very late in the relaxation regime, when the window is slightly expanded.

4. The inclusion or exclusion of the special named entity 'keywords' corresponding to the answer type. This alternates between 'on' and 'off' at every relaxation step.

5. The keyword priority threshold, which starts low and is slowly increased with further steps.

The iterative relaxation technique may be considered as an implicit scoring strategy in which the relevance of a document relative to the question is inversely proportional to its window span size, and directly proportional to the tf.idf sum of keywords appearing in the window. The final RS module output is a ranked list of document IDs.

1.3 Information Extractor

The Information Extractor (IX) module extracts candidate answers from relevant documents retrieved by the RS module. JAVELIN implements a multi-strategy approach to answer extraction (Czuba et al., 2003), and includes several different implementations of the IX module. The assumption is that optimal performance in extracting different answer types may require a number of separate strategies, ranging from simple finite state transducers to classifiers trained to separate correct answers from incorrect answers. Each information extraction method scores candidate answers (with their corresponding passages) and passes them down the pipeline to the Answer Generator for canonicalization and clustering. Scoring functions differ among the IX modules, and their scores are not normalized individually.

The first and most straightforward extraction approach (Light IX) implements a non-linear weighted proximity metric that spans a passage of three sentences. This approach identifies candidate answers of the appropriate type and assigns a score based on proximity to question terms and their synonyms. This IX module implements specific strategies for definition questions, relationship questions, and person biography questions.

Another strategy is to combine statistical features emerging from a passage and an answer, and train a classifier to separate correct answers from incorrect ones for each specific answer type. The classifiers were trained on Trec9 and Trec10 datasets, and include a support vector machine (SVM IX) classifier as well as a k-nearest neighbors (KNN IX) classifier. The positive versus negative data ratio was tuned for best performance.

The final strategy is based on traditional information extraction and implements a finite state transducer (FST IX). This approach is modeled after each question type/relationship that the Question Analyzer identifies. The FST IX is based on lexical features, Wordnet features, and surface form flags. It is most appropriate for question types where answers can be extracted using a handful of simple patterns (Ravichandran and Hovy, 2002)[1].

1.4 Answer Generator

The Answer Generator (AG) module is responsible for producing a ranked list of answers from the set of answer candidates produced by the IX modules. The AG also

[1]We are also working to add an Information Extractor based on NLP analysis of answer passages (NLP IX); although this module is shown in Figure 1, it was not integrated into the system evaluated for TREC 2003.

makes use of the Request Object produced by the QA module. The AG performs several normalization functions, such as combining similar answer candidates and applying answer type checking to filter out inappropriate answers.

Type checking is very important when selecting proper answers from a candidate list. Earlier modules can sometimes produce irrelevant answers, leading to candidates which do not match the question. For example, for the question, "What continent is Egypt on?" the AG might receive the candidates, "Middle East", "Arab", "Islam", and "Africa". The answer type is location (continent), so the AG should select "Africa" as the final answer.

Currently, the AG uses WordNet, the Tipster Gazetteer (TIPSTER, 1992), the Web, and type-specific patterns to support answer type checking. WordNet provides hypernym and meronym relationship information used by the AG to determine the relationships between candidate answers and the target answer type (Cardie et al., 2000). The Web is also used as a resource to see how strongly these relationships are supported: the AG creates validation patterns from the answer type and answer candidates, sends a request to the Web, and generates a score from the number of retrieved documents (Magnini et al., 2002). For location questions, the AG examines both WordNet and the Gazetteer; the latter provides extensive information on various city, state, and country names.

When no adequate answer can be found, the AG communicates this information to the Planner. This allows a new strategy to be applied to the question, potentially resulting in a correct answer.

1.5 Execution Manager and Repository

The Execution Manager (EM) module coordinates communication between the Planner and all other modules. Additionally, it communicates with the Repository, a centralized SQL database containing all of the information and analysis generated by JAVELIN during question answering. Aside from creating a complete history of system behavior, this allows analysis data to be collected in a manner such that the Answer Justification module can easily create justification results for display to the user. The EM's runtime coordination and communication with the Repository provide fundamental support for JAVELIN's object-oriented, modular architecture.

1.6 Planner

The Planner module is responsible for controlling the question-answering process. It selects and issues sequences of module calls to maximize the expected utility of the information JAVELIN produces, taking into account the current information state and system resources. This increases the system's flexibility to generate different strategies at run-time, exploit different modules' strengths for specific question types, and more robustly handle module failures as they arise.

Figure 2 presents a high-level view of the Planner's architecture. The Planner operates as a service for the JAVELIN GUI, and communicates with the rest of the system via the Execution Manager.[2] Upon receiving a new question, the Planner calls the Question Analyzer (via the EM) and uses the resulting question analysis to generate a planning problem describing the *initial state* (features of the question analysis), and an *information goal* based on the expected answer type.

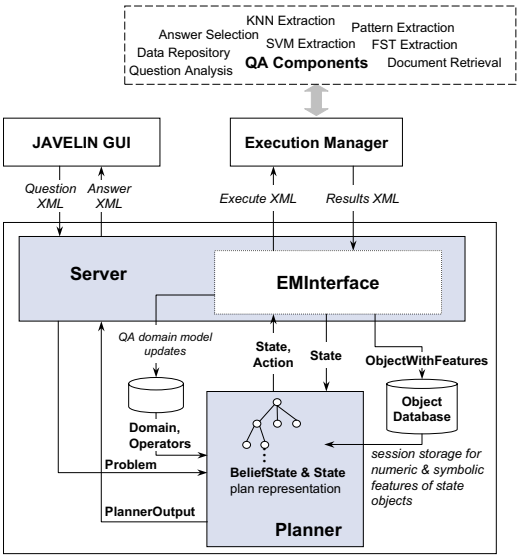

Figure 2: Relationship of the Planner to the JAVELIN QA system. Within the Planner module, a server component provides the QA domain-specific functionality, while the Planner component provides domain-independent planning functionality.

The planning and execution process is based on a forward-chaining utility-based algorithm that performs a best-first search across the set of possible information states (Hiyakumoto and Veloso, 2002). Beginning with the initial state representation and an empty plan, the Planner evaluates all actions applicable in the current state, selecting the one whose projected outcome states have the highest expected utility. Expected utility is estimated using a weighted combination of the states' information quality metrics and likelihoods. The internal planning state is then projected forward to reflect the pos-

[2] It should be noted that the normal operating mode for the GUI and Planner includes an option for the Planner to solicit user-feedback during the QA process. This functionality was disabled during the TREC evaluation, and a batch-mode script was used in place of the standard GUI to issue questions to the Planner module.

sible outcomes of the chosen action, and the selection process repeats. At each step, the algorithm also decides between executing the first unexecuted action in the plan and continuing to plan with the uncertainty of the projected states. If a module call is executed (e.g., documents are retrieved), the results are used to update the internal information state model and a replanning decision is made. The process continues until the goal is satisfied (the system has found an answer with the correct answer type and an expected utility exceeding a predefined threshold), or available resources have been exhausted. Upon terminating, the Planner returns the system's answer or a failure message.

Critical to this decision-making process is the Planner's *domain model* of the QA process, which is defined in terms of *types*, *literals*, *metrics* and *operators*. Table 1 highlights key features of the QA domain model used by the Planner for TREC12. Types define the categories of objects created and transformed by the process, including the system's question and answer type hierarchy (as subtypes of qtype and atype respectively) and intermediate data objects (e.g., a docset). Literals define information features (e.g., the number of question keywords extracted during question analysis), data relationships (e.g., the relationship between a set of answer candidates and the documents used to create them), and system status (e.g., module availability). Metrics represent system resources (e.g., system_time) and quality estimates (e.g., docset_quality) for the information objects present in the current state, and are used in the calculation of a state's utility. Together, metrics and literals comprise the Planner's internal state representation. Operators (actions) define the QA processing functions that the Planner can control, described in terms of their preconditions (literal and metric conditions that must be true in the current state in order for the action to be *applicable*), and a set of probabilistic effects (possible changes to the information state that may occur upon execution).

The domain operators implemented for TREC 12 essentially provide a one-to-one mapping to the QA system components, with the exception of **check_answers**, which triggers an internal sanity check before the Planner terminates (to verify the answer confidence is nonzero). Obviously, these operators are not the only ones we could have defined: we could have chosen to give the Planner finer-grained control of the system (e.g., defining multiple retrieval operators specifying different retrieval methods) assuming we identify state features that reliably indicate appropriate contexts for each operator. However, given the time constraints, we decided to focus solely on the extraction performance. Thus, the effective role of the Planner in TREC12 was to dynamically select amongst the four extraction components.

The IX selection strategy implemented for factoid

types: question, qtype, atype, docset, fillset, anslist, ix-name
subtypes of qtype: entity, activity,...
subtypes of atype: temporal, location,...

literals:
(interactive_session)
(satisfies <*question*> <*atype*> <*anslist*>)
(request <*question*> <*qtype*>)
(retrieved_docs <*docset*> <*qtype*>)
(no_docs_found <*qtype*>)
(no_more_docs <*qtype*>)
(candidate_fills <*fillset*> <*qtype*> <*docset*> <*ix-name*>)
(no_fills_found <*qtype*> <*docset*> <*ix-name*>)
(ranked_answers <*anslist*> <*qtype*> <*fillset*>)
(no_answers <*qtype*> <*fillset*>)
(displayed <*anslist*>)

metrics:
system_time
request_quality
docset_quality
fillset_quality
answer_quality

operators:
retrieve_documents
extract_KNN_candidate_fills
extract_FST_candidate_fills
extract_SVM_candidate_fills
extract_Light_candidate_fills
rank_candidates
check_answers

Table 1: Partial listing of the TREC12 QA domain model.

questions was determined by comparing the performance of each IX module on the TREC 9 and 10 question sets. The sole state feature used in our strategy decisions was the expected answer type, and feedback loops to call multiple IX modules were considered only as a means for failure recovery when prior execution (at either the extraction or answer generation stages) did not identify any potential answers. The resulting extraction module preference orders (summarized in Table 2) attempt to exploit differences in their precision and recall to maximize their combined question coverage. Preference is generally given to higher-precision modules that may fail to return an answer more often, with the less-accurate, higher-recall modules used as a fallback method. Two examples of action sequences produced by the Planner are presented in Figure 3 illustrating the use of this strategy.

Both definition and list questions were handled with a single fixed strategy by enabling only the extraction operator for the IX module we estimated was "best" for each (namely the Light and KNN modules, respectively). This decision was made because only the Light extractor was capable of generating longer answers, and the KNN extractor had the highest recall of unique answers when evaluated on past TREC list questions.

Q: What movie won the Academy Award for best picture in 1989?

A: 1. Driving Miss Daisy
2. Chariots of Fire
3. Saving Private Ryan
...

```
<retrieve_documents DS6024 RO6637>
<extract_SVM_candidate_fills FS18637 RO6637
DS6024>
<rank_candidates AL5184 RO6637 FS18637>
<check_answers A5046 AL5184 Q2694>
```

Q: What country is Aswan High Dam located in?

A: 1. Egypt
2. Saudi Arabia

```
<retrieve_documents DS5957 RO6570>
<extract_FST_candidate_fills FS17958 RO6570
DS5957>
<extract_SVM_candidate_fills FS17962 RO6570
DS5957>
<rank_candidates AL5119 RO6570 FS1962>
<check_answers A4983 AL5119 Q2504>
```

Figure 3: Sample action sequences generated and executed by the JAVELIN system. The second question shows the Planner's ability to recover from the FST extractor's failure to produce any candidates by invoking the SVM extractor.

location	FST , SVM , Light , KNN
temporal	FST , KNN , SVM , Light
causal-antecedent	Light , KNN , FST , SVM
causal-consequence	Light , KNN , FST , SVM
other[3]	SVM , FST , KNN , Light

Table 2: Extraction module preference ordering for factoid questions. Each extractor is called in succession until the system generates an answer or all options have been exhausted. No ordering strategy was used for definition and list questions: in these cases the Planner just invoked the single "best" extractor for each (Light and KNN respectively).

2 Results

In the main (factoid) task, JAVELIN answered correctly 55 questions, 3 inexact, and 3 unsupported, for an accuracy of 0.133. The system achieved the highest accuracy on location questions: 0.3125 (Table 3). For the 50 definition questions in TREC12, the JAVELIN system achieved an average F score of 2.16. JAVELIN obtained a 0.052 score on list questions (Table 4).

3 Analysis

Table 5 shows the F scores of the list questions according to the answer type. As can be seen in Table 6, 32% of the questions were misclassified.

The Question Analyzer's accuracy in assigning answer types has been analyzed using the TREC 9, 10, and 11 corpora. Since grammar development for analysis is based on the TREC 9 and 10 corpora, we have separated our results and provide TREC 11 as an example

Assigned Answer Type	#Qs	Accuracy
action	1	0
definition	1	0
description	3	0
lexicon	13	0.1538
location	64	0.3125
numeric-expression	97	0.1649
object	93	0.0430
organization-name	22	0.0455
person-name	33	0.1818
process	34	0
proper-name	1	0
regexp	1	0
relation	1	0
temporal	46	0.1304
type-of	1	0
unknown	2	0

Table 3: Answer type accuracy for factoid questions.

of unsupervised accuracy; a breakdown by answer type appears in tables 7 and 8. The average supervised and unsurpervised accuracies are 97.4% and 92.0%, respectively. Both are acceptable, but the five percent drop in unsupervised performance indicates that we have room to improve. However, the training data used for the Question Analyzer has a different question distribution from the TREC 2003 list questions. The low score in list questions reflect this problem through an increase in misclassification rate in TREC 2003.

Our experiments show that the FST IX seems to better cover the location question answer space, while the

				Full Parse		Partial Parse		Pattern	
Answer Type	Questions	Correct ATypes	Accuracy	Corr.	Err.	Corr.	Err.	Corr.	Err.
temporal	120	120	100%	113	0	8	0	0	0
object	206	200	97%	174	4	22	2	4	0
location	205	204	100%	180	1	18	0	6	0
proper-name	220	216	98%	196	4	15	0	5	0
numeric-expression	141	136	96%	123	3	12	0	1	2
lexicon	63	54	86%	54	5	0	3	0	1
definition	165	159	96%	159	3	0	1	0	2
person-bio	39	39	100%	39	0	0	0	0	0
regexp	4	4	100%	4	0	0	0	0	0
causal	14	14	100%	10	0	3	0	1	0
procedural	4	4	100%	4	0	0	0	0	0
action	6	6	100%	6	0	0	0	0	0
relation	1	1	100%	1	0	0	0	0	0
Overall	1188	1157	97.39%	1063	20	78	6	17	5

Table 7: Performance of the Question Analyzer on TREC 9 and 10 questions.

Answer Type	Questions	Correct ATypes	Accuracy
temporal	99	99	100%
object	80	69	86%
location	111	106	95%
proper-name	100	91	91%
numeric-expression	74	65	88%
lexicon	25	19	76%
definition	5	5	100%
person-bio	0	0	-
regexp	0	0	-
causal	0	0	-
procedural	6	6	100%
action	0	0	-
relation	0	0	-
Overall	500	460	92.00%

Table 8: Performance of the Question Analyzer on TREC 11 questions.

Question Set	Best	Median	Worst	JAVELIN
Factoid	0.7	0.177	0.034	0.133
Definition	0.555	0.192	0.0	0.216
List	0.396	0.069	0.0	0.052

Table 4: JAVELIN TREC12 scores. JAVELIN scores are compared with the best, median, and worst accuracy scores in the TREC 2003 QA track.

AnswerType	#Qs	Average FScore
location	12	0.06475
proper-name	1	0.017
person-name	10	0.0248
organization-name	3	0.19767
object	11	0.02627

Table 5: TREC12 List Questions Task.

causal strategy implemented in the proximity based IX handles causal-antecedent/causal-consequence questions better. The definition question strategy in the IX and AG seems to extract long and complete definitions, mostly from apposition contexts, with reasonable success.

To examine the impact of answer type checking on JAVELIN system performance, we ran two tests on the TREC 9 corpus, one with and one without type checking. The results appear in table 9. Both the TREC score and the MRR score increased approximately 12% with type checking enabled. For location-specific questions from TREC 9, 10, and 11, seen in table 10, these scores im-

AnswerType	#Qs	#Correct	Misclassifications
location	12	10	object(2)
proper-name	1	0	object(1)
person-name	10	4	proper-name(1) org-name(1) object(3) QA failure(1)
org-name	3	2	object(1)
object	11	9	causal(1) lexicon(1)
Total	37	25	12 (32%)

Table 6: TREC12 List Question Classification.

	Without TC	With TC
Trec Score	0.193	0.217
MRR	0.24	0.271

Table 9: TREC9 results with/without type checking (TC)

proved a full 23% because of the additional information available from the Gazetteer.

We tested the system with the Planner's IX-ordering strategy on 500 additional questions from TREC 11. As the results in Table 11 show, the system's performance with the Planner (PL) was not appreciably better than it was using a fixed sequence with the best single SVM extractor (79 versus 77 questions correctly answered). We attribute this to our use of a relatively small data set for estimating the extractor performance (1193 questions from TREC 9 and 10; particularly problematic for the less-common answer types), coupled with the fact that the questions used to estimate the planning parameters were the same ones used to develop the extractors themselves, leading to possible overestimates of their performance.

Even if our ordering strategy had perfectly fit the question set (PL+), the Planner would have enabled the system to correctly answer just 8 more questions than the single-extractor fixed sequence version could answer. Thus, although a good operator model is likely to include answer type information, that by itself is not sufficient to correctly determine the context for applying each extraction method. To achieve the best possible performance from the current QA components (PL*) additional predictive features must be identified and incorporated into the Planner's model of the QA domain.

4 Current and Future Work

Our primary focus after TREC 2003 and for the future is on more complex relationship and scenario-based questions. For example, a human interacting with JAVELIN may wish to collect data on a terrorist organization incre-

	Without TC	With TC
Trec Score	0.177	0.218
MRR	0.207	0.259

Table 10: Results on 316 location questions from TREC 9, 10, and 11 with and without type checking (TC).

Answer Type	#Qs	**S**VM	**P**L	**P**L+	**P**L*
location	109	26	27*	26 (S)	33
temporal	99	28	27	28 (S)	43
object	90	1	2	3 (K)	5
num-expression	69	7	8	7 (S)	14
person-name	50	3	3	7 (K)	10
proper-name	31	5	5	7 (L)	8
lexicon	24	3	3	3 (S;L)	3
org-name	16	4	4	4 (S)	4
other	12	0	0	0	0
OVERALL	500	77	79	85	120

Table 11: JAVELIN performance on TREC 11 questions using a fixed sequence with the best single extractor (SVM) and the Planner with answer-type-based IX operator model (PL). The last two columns contain upper bounds for an optimal answer-type-based Planner model (PL+), and the best possible performance for the current QA components by chosing an optimal sequence for each question (PL*).

mentally for use with subsequent question answering. To facilitate this expansion, each module is focusing on enhancements necessary to process more complex question types and content.

The QA module is enhancing the depth and accuracy of its language processing output, in addition to creating and improving analysis for scenario-based question types.

A new NLP-based IX is being developed. This IX is based solely on linguistic analysis of the question and answer data, unifying syntactic and semantic forms of the question against similar analyses of candidate answer passages.

A Text Processing module is being added to the system to facilitate much of the newly-required natural language processing. This module centralizes textual analysis, providing much of the tagging, hypernym/meronym, syntactic, and semantic processing used by other JAVELIN modules.

A Linguistic Reasoner module is being created to serve as a more linguistically aware sub-planner. For complex questions not answerable by a "straight shot" through the existing JAVELIN system (e.g. questions containing an implied subquestion), the LR uses a dependency-based mechanism to find missing information and recursively communicate this need to the main JAVELIN Planner. This allows both more sophisticated linguistic reasoning and more intricate use of JAVELIN itself to answer com-

plex questions.

All of these enhancements are being developed to improve the JAVELIN system's capability to answer complex questions dependent on semantics or context. The results from our participation in the first NIST relationship QA pilot evaluation indicate that approaches which are currently common for factoid-style questions are not applicable to relationship and scenario questions. We hope to apply our flexible architecture and linguistic knowledge to areas of question answering which require deeper reasoning about the text. It will be useful to see how these linguistic techniques perform for factoid questions, but we believe that their primary strength will be in addressing other, more complex types of questions.

References

D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning, vol. 34, no. 1-3*.

E. Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. *VLC*.

C. Cardie, V. Ng, D. Pierce, and C. Buckley. 2000. Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering system. *ANLP*.

K. Czuba, J. Prager, and A. Ittycheriah. 2003. In question answering, two heads are better than one. *HLT-NAACL*.

Fellbaum. 1998. Wordnet - an electronic lexical database.

S. M. Harabagiu, M. A. Pasca, and S. J. Maiorano. 2000. Experiments with open-domain textual question answering. *COLING*.

L. S. Hiyakumoto and M. Veloso. 2002. Towards planning and execution for information retrieval. *Proceedings of AIPS Workshop on Exploring Real-World Planning*.

B. Magnini, M. Negri, R. Prevete, and H. Tanev. 2002. Is it the right answer? exploiting web redundancy for answer validation. *ACL*.

E. Nyberg and T. Mitamura. 2000. The kantoo machine translation environment. *AMTA*.

P. Oglivie and J. Callan. 2002. Experiments using the lemur toolkit. *TREC*.

D. Ravichandran and E Hovy. 2002. Learning surface text patterns for a question answering system. *ACL*.

TIPSTER. 1992. Tipster gazetteer 4.0. *ftp://crl.nmsu.edu/CLR/lexica/gazetteer/*.