SEMANTIC EXTENSIONS OF THE EPHYRA QA SYSTEM FOR TREC 2007

Nico Schlaefer[1], *Guido Sautter*[2], *Jeongwoo Ko*[1], *Justin Betteridge*[1], *Manas Pathak*[1] *and Eric Nyberg*[1]

[1] Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{*nico, jko, jbetter, manasp, ehn*}@*cs.cmu.edu*

[2] Institute for Program Structures
and Data Organization
Fakultät für Informatik
Universität Karlsruhe (TH)
sautter@ira.uka.de

ABSTRACT

We describe recent extensions to the Ephyra question answering (QA) system and their evaluation in the TREC 2007 QA track. Existing syntactic answer extraction approaches for factoid and list questions have been complemented with a high-accuracy semantic approach that generates a semantic representation of the question and extracts answer candidates from similar semantic structures in the corpus. Candidates found by different answer extractors are combined and ranked by a statistical framework that integrates a variety of answer validation techniques and similarity measures to estimate a probability for each candidate. A novel answer type classifier combines a statistical model and hand-coded rules to predict the answer type based on syntactic and semantic features of the question. Our approach for the 'other' questions uses Wikipedia and Google to judge the relevance of answer candidates found in the corpora.

1. INTRODUCTION

In this paper we describe the Ephyra question answering system that has been evaluated in the TREC 2007 QA main track. The system extends the approach we reported on at last year's TREC evaluation [1] and the overall architecture and design goals have been described in past papers [2, 3]. Here we focus on recent improvements and new techniques that proved effective in this year's evaluation.

We augmented our answer extraction approaches for factoid and list questions with a high-accuracy semantic answer extractor that is based on semantic role labeling. The question is transformed into a semantic representation and answer candidates are extracted from phrases which match this representation. Different query generation techniques are used to retrieve relevant text passages, ranging from simple keyword queries to specific query strings such as reformulations of the question into answer patterns. WordNet is used to expand query terms with semantically related concepts. As full semantic role labeling of all retrieved sentences is a prohibitively time-intensive task, we carefully select candidates that are further analyzed and transformed into semantic representations. A fuzzy similarity metric is then used to compare these representations to the question representation to identify semantic structures that potentially contain an answer. The similarity measure was designed to be flexible and robust in order to maximize the recall of the answer extractor.

A statistical framework for answer selection combines the answer candidates produced by this semantic approach and previously developed answer type based and pattern based extractors [4]. The framework estimates the probability of an answer candidate based on a set of answer validation and similarity features. Validation features use external semantic resources to verify an answer, while similarity features measure the syntactic and semantic similarity to other candidates.

One of the crucial steps in answering factoid and list questions is the classification of the questions with respected to the expected answer type. For this purpose, we have developed a classifier that uses both manually encoded rules and a statistical model to predict the answer type given a set of syntactic and semantic features of the question. This hybrid approach outperforms our previous pattern-based classifier.

To answer the 'other' questions, we use Wikipedia and Google searches to identify keywords that frequently occur in the proximity of the target. Assuming that these keywords provide relevant information on the target, we favor those answer candidates that contain the frequent terms. Additional filtering techniques are used to drop redundant and non-informative answers to improve the precision of our responses.

In this year's TREC evaluation, answers were extracted from both the AQUAINT2 newswire corpus and the Blog06 corpus, a crawl of a large volume of Web logs. We submitted runs using only the newswire corpus and a combination of both corpora, which gave us an insight into the benefits and also the challenges arising from the use of a large corpus containing a significant proportion of noisy text as an additional source.

The remainder of this paper is organized as follows. Section 2 gives an overview of related approaches. Section 3 discusses recent improvements and extensions of our pipeline for factoid and list questions, while Section 4 deals with the 'other' questions. Section 5 describes our TREC runs and summarizes the evaluation results; Section 6 outlines open issues for future work.

2. RELATED WORK

Moldovan and Novischi [5] use relations in WordNet [6] to derive topically related terms for query expansion. Terms are considered semantically similar if they are linked through a lexical chain, a sequence of related WordNet synsets. For each of the WordNet relations, a weight has been determined empirically that reflects the degree of similarity between related synsets. We adopt the idea of lexical chains and reuse the proposed weights to calculate confidence scores for semantically related concepts.

Nyberg et al. [7] describe how the JAVELIN QA system has been extended with domain semantics to answer questions in a restricted domain. A manually created ontology covers frequent concepts in this domain and English expressions with domain-specific meanings. [8] introduces a lightweight knowledge-based reasoning framework for JAVELIN. Questions and text passages are transformed into uniform semantic representations and a flexible unification framework matches questions with relevant passages, using weighted semantic relations between terms.

The QA system from the National University of Singapore [9] used the semantic role labeling system ASSERT [10] to answer factoid and list questions in past TREC evaluations. Predicate-argument structures are extracted from the question and answer sentences. The predicates are then compared using a similarity metric composed of the similarity of the predicate verbs and the similarity of the arguments. We refined the similarity measure for predicates to make it more flexible and robust to parsing errors in order to improve the recall of this approach. The terms in the arguments are expanded with WordNet, and the semantic similarity of the arguments is measured in addition to their syntactic similarity.

To select the most probable answer(s) from an answer candidate list, QA systems have applied several different answer selection approaches. One of the most common approaches relies on external resources such as WordNet, CYC and gazetteers for answer validation [11, 12, 13]. The Web has also been used for answer reranking by exploiting search engine results produced by queries containing the answer candidate and question keywords [14]. Collecting evidence from similar answer candidates to boost the rank of redundant answers is also important for answer selection. One popular approach is to cluster identical or complementary answers [15, 16]. Our recent work [4] proposed a unified probabilistic answer ranking framework which combines different techniques to validate answers and exploit answer redundancy for the answer selection task. The results in this year's TREC evaluation demonstrate the effectiveness of this framework.

The evolution of answer type classification approaches mirrors that of natural language understanding algorithms in general: initial approaches consisting of handcoded rules [16] or patterns [17] were followed by a variety of data-driven approaches based on simple [18] or complex [19] features, with the occasional emergence of a hybrid algorithm [20]. While a machine learning technique based on syntactic and semantic features achieves one of the highest reported accuracies (89.3%) for classifying English questions with fine granularity (50 types) using a standard data set [19], it is also interesting to note that Day et al. [20] report that combining a rule-based classifier with a data-driven classifier yields higher accuracy on Chinese questions than either approach used in isolation. Notably absent is an empirical study comparing different global strategies for combining manually-encoded and automatically-acquired linguistic knowledge for answer type classification.

A simple but very effective approach for answering the 'other' questions has been introduced by Kaisser et al. [21]. Google is used to search for text snippets that contain the target of the question and the frequencies of the terms in the snippets are counted. Assuming that frequent terms provide important information on the target, answer candidates containing these terms are assigned higher confidence scores. We adopt a similar approach but use Wikipedia in conjunction with Google to determine the term frequencies.

3. FACTOID AND LIST QUESTIONS

Our approach for factoid and list questions is based on a pipeline architecture consisting of components for question analysis, query generation, search, and answer extraction and selection. While the pipeline layout has been described in some detail in previous papers [2, 3], this section focuses on recent extensions that were deployed for this year's TREC evaluation, including an improved answer type classifier (Section 3.1), a high-precision semantic answer extraction approach that is based on semantic role labeling (Sections 3.2 - 3.5), and a probabilistic framework for the combination and selection of answer candidates (Section 3.6).

3.1. Answer Type Classification

Classifying questions in terms of their expected answer type is a crucial first step in automatic question answering. Some QA systems rely solely on handcoded rules or heuristics to determine the expected type of the answer, while other systems rely solely on automatically-learned patterns or probabilistic models based on features to classify questions. For this year's TREC competition, we adopted a hybrid approach that combines manually-encoded rules with a learned model. The result of answer type classification is used downstream in the Ephyra system to aid answer extraction and selection.

Answer Types		
acronym	language	rate
angle	legalSentence	relation
birthstone	location	religion
bodyPart	material	score
causeOfDeath	medicalTreatment	size
color	money	socialTitle
creature	musicalInstrument	sport
crime	musicType	style
date	nationality	temperature
disease	number	time
dramaType	pathogen	timezone
drug	percentage	unit
duration	profession	url
food	properName	zodiacSign
frequency	range	

Table 1. Top-level answer types used in Ephyra.

3.1.1. Answer Type Hierarchy

A crucial but often variable aspect of answer type classification is the answer (or question) typology used. Early QA systems used a small set of coarse-grained types, but modern systems, including Ephyra, usually use a larger set of fine-grained types. Because of the underspecified nature of many questions (e.g. *Where is the conference?* as opposed to *What city is the conference being held in?*), the typology used is usually a hierarchy, allowing for varying levels of granularity in the classification. Ephyra uses a set of 154 answer types arranged in a hierarchy with 44 top-level categories, which are shown in Table 1. The hierarchy was designed to cover answer types frequently found in past TREC questions. Named entity recognizers have been devised to extract candidate answers of all of these types except for a few high-level categories such as *food* or *event*. We use the Stanford NE Recognizer [22] to extract answers of the types *person*, *organization* and *location* and rule- and list-based taggers we built for the remaining types.

3.1.2. Features

The features we use for answer type classification fall into three categories: lexical, syntactic, and semantic. They are described in detail below. All of our features actually appear as binary features when input to the learning algorithm.

Lexical Features

- UNIGRAM : Individual tokens present in the question.
- BIGRAM : Pairs of adjacent tokens in the question.

Syntactic Features

- FOCUS_ADJ : The focus adjective or adverb of the question; only applicable for *how* questions that ask about the degree of some property, such as *How fast can whales travel?*
- MAIN_VERB : The main verb of the question, determined from the syntactic parse of the question using Collins-style head rules.
- WH_WORD : The question word.
- WH_DETERMINER : Indicates whether the question word serves as a determiner to the focus word.

Semantic Features

- FOCUS_TYPE : The semantic type of the question focus word (e.g. *city* in *Which city hosted the 2002 Winter Olympics?*); only applicable for questions with a wh-word of *what* or *which*. The focus word is identified using a manually-compiled set of syntactic patterns which are matched against a syntactic parse of the question. The semantic type of the focus word is determined by
  - traversing the WordNet hypernym tree for each sense of the focus word,
  - looking up each hypernym synset in a manually-constructed, many-to-one mapping[1] from WordNet synsets to a set of general answer types,
  - adding any successful mappings to a set of candidate types, and
  - selecting the candidate type which corresponds to the hypernym synset with the shortest hypernym tree traversal.

  Consequently, an incorrect disambiguation of the focus word sense is possible, although not common. If the focus word's semantic type cannot be determined using this approach, the focus word itself is used as the value of this feature.

3.1.3. Classification Approach

Initially, the above features were meant to be used for training a model-based classifier. However, given information as predictive as the focus word type, focus adjective, and the question word, the answer type classification task becomes fairly straightforward. This motivated us to construct a simple rule-based classifier that tests the same features (including BIGRAM and UNIGRAM). We also built a hybrid classifier that combines the outputs of the model-based and rule-based classifiers using their associated confidence scores.

[1] Similar in spirit to the use of WordNet for answer type classification in [23].

Approach	Acc. (%)
Model-based	72.38
Rule-based	68.18
Hybrid	79.02

Table 2. Performance of different answer type classification strategies.

To compare these three approaches, we measured their classification accuracy on TREC 13 questions, with questions from TREC 8-12, 14 and 15 serving as training data for the model-based classifier and as development data for the rule-based classifier. The results are shown in Table 2. Because the hybrid classifier outperformed both the model-based classifier and the rule-based classifier, this is the approach we adopted. In the final system used for this year's evaluation, the model-based component of our approach was trained on questions from TREC 8-15.

3.2. Semantic Parsing of Questions

We augmented our answer type based and pattern based answer extraction techniques [2, 3] with a semantic parsing approach that generates a semantic representation of the question and extracts answer candidates from phrases in the corpus that match this representation.

The semantic role labeling (SRL) system ASSERT was used to label semantic structures in questions and the corpus, using the PropBank inventory of semantic roles [10]. SRL is a form of shallow semantic parsing that labels predicate-argument structures consisting of a target verb, usually describing an event, and a set of arguments along with their semantic roles in the event. For instance, the sentence *The CMU campus at the west coast was founded in 2002.* contains the predicate-argument structure

- TARGET: founded
- ARG1: The CMU campus
- ARGM_LOC: at the west coast
- ARGM_TMP: in 2002

where ARG1 refers to the *patient* or *theme*, ARGM_LOC to the *location* and ARGM_TMP to the *time*.

Since SRL systems often fail to correctly label the semantic roles in questions, we first transform the question string into a statement, using a number of simple syntactic transformation rules. An appropriate rule is selected based on the interrogative pronoun of the question, adjacent prepositions and noun phrases, and the expected answer type. Examples of transformation rules are given in Table 3. For instance, applying the second rule, the question *In what year was the CMU campus at the west coast established?* can be transformed into the statement *The CMU campus at the west coast was established now*. The phrase *now* is a placeholder for an argument that specifies the *time* and represents the answer to the question.

- TARGET: established
  - TERM: established (NE Types: -)
    Aliases: founded (Weight 0.8),
    launched (Weight 0.7)
- ARG1: the CMU campus
  - TERM: CMU (NE Types: Organization)
    Aliases: Carnegie Mellon
    (Weight 0.9)
  - TERM: campus (NE Types: -)
    Aliases: -
- ARGM_LOC: at the west coast
  - TERM: west coast (NE Types: Location)
    Aliases: -
- ARGM_TMP: *missing*

Fig. 1. Semantic representation of the sample question.

ASSERT is used to extract predicate-argument structures from the resulting statement. The placeholder argument is dropped and the corresponding semantic role is marked as *missing*, indicating that this is the information the question is seeking. The arguments are further split into terms, which are units of meaning consisting of one or more tokens (e.g. ”Carnegie Mellon”, ”west coast”). Terms are used for query generation and expansion (Section 3.3) and to measure the similarity between predicates in questions and corpus sentences (Section 3.4).

We use our NE recognizers and WordNet lookups to extract compound terms from the predicate arguments. WordNet is also used to expand terms with semantically similar concepts, following an approach similar to [9]. Each term is mapped to a synset in WordNet and a breadth-first search along WordNet relations identifies related synsets. We make use of relations such as *synonym*, *hypernym*, *hyponym*, *holonym* and *meronym* and restrict the search depth to a maximum of two relations. The related terms are assigned confidence values based on the relations on the path from the original term, adopting the weights suggested in [5].

Figure 1 shows the semantic representation that is generated for the sample question *In what year was the CMU campus at the west coast established?*

3.3. Query Generation and Expansion

We generate various types of queries, ranging from recall-oriented queries such as bags of keywords to specific query strings which retrieve text passages that closely match the structure of the question:

Interrogative Pronoun + Adjacent Phrases	Answer Type	Transformation Rule
where	any	drop *where*, move auxiliary verb to main verb, append placeholder argument *here*
[PP] + *what* + NP	*date*, *time* or subtype	drop [PP] + *what* + NP, move auxiliary verb to main verb, append placeholder argument *now*

Table 3. Question transformation rules. [PP] refers to an optional preposition, NP to a noun phrase.

- **Keyword queries** are duplicate-free sets of the content words in the question.
  Example: *CMU campus west coast established*
- **Term queries** consist of the question terms (single tokens or compound expressions), expanded with semantically related concepts.
  Example: *(CMU OR "Carnegie Mellon") campus "west coast" (established OR founded OR launched)*
- **Predicate queries** are formed from the predicate verb and arguments.
  Example: *"the CMU campus" "at the west coast" established*
- **Reformulation queries** are obtained by rephrasing the question into an answer pattern.
  Example: *"the CMU campus at the west coast was established in"*

The above queries are used to retrieve text passages from both the Web and the corpora used in this year's TREC evaluation (AQUAINT2 and Blog06), and answer candidates are extracted from both sources. The Web answers are matched with the answers found in the corpora to identify supporting documents, following the answer extraction approach described in [2].

We use Google to search the Web and the Indri search engine, which is part of the Lemur toolkit [24], to retrieve passages from the TREC corpora. For the top 100 Google snippets, we fetch the entire Web documents and convert them to plain text. Both the Web documents and the passages from the corpora are segmented into sentences.

3.4. Extraction of Candidate Answers

Semantic parsing is a time-intensive task and not all of the retrieved sentences are equally likely to contain an answer. Thus we first narrow down the number of candidate sentences before we parse them with ASSERT. A sentence is considered relevant if it meets the following constraints:

- The number of tokens in the sentence falls between upper and lower thresholds that are typically satisfied by a natural language sentence.
- The sentence contains the predicate verb from the question or a semantically related verb.
- If the answer type of the question is known, the sentence must contain an entity of that type.
- The sentence contains at least one additional term that is semantically similar to a term in the question.

Sentences that pass the above tests are parsed and transformed into a semantic representation similar to the one given for the question in Figure 1. The semantic structure of each sentence is compared to the question and a similarity score is calculated as described in the following.

The **term similarity** of two terms t_1 and t_2 is defined as the Jaccard coefficient of the sets of content words W_1 and W_2 in the terms:

$$S_T(t_1, t_2) := J(W_1, W_2) = \frac{|W_1 \cap W_2|}{|W_1 \cup W_2|}$$

The **expanded term similarity** between an answer term t_a and a question term t_q takes into account that each question term t has related concepts $R = \{r_1, ..., r_n\}$ with weights $w(r_1), ..., w(r_n)$:

$$S_{ET}(t_a, t_q) := \max_{t \in \{t_q\} \cup R} (w(t) \times S_T(t_a, t)) \\ \text{where} \quad w(t_q) := 1$$

The **verb similarity** of an answer predicate p_a and a question predicate p_q is the expanded term similarity of the predicate verbs v_a and v_q:

$$S_V(p_a, p_q) := S_{ET}(v_a, v_q)$$

The **argument similarity** of an answer predicate p_a and a question predicate p_q is determined by comparing the sets of terms within the arguments of the predicates, denoted T_a and T_q. We have extended the concept of the Jaccard coefficient to take the semantic similarity of terms into account, rather than just distinguishing between common terms and terms that appear exclusively in one set:

$$S_A(p_a, p_q) := \frac{\sum_{t_a \in T_a} \max_{t_q \in T_q} (S_{ET}(t_a, t_q))}{|T_q| + \underset{t_a \in T_a}{\text{count}} \left(\max_{t_q \in T_q} (S_{ET}(t_a, t_q)) = 0 \right)}$$

Each term in T_a is compared to all terms in T_q and the maximum of the similarity scores is computed. If the maximum is larger than 0, then the term is assumed to be covered by both predicates and the numerator of the coefficient is incremented by this score, else the denominator is incremented by 1.

Finally, the **predicate similarity** of an answer predicate p_a and a question predicate p_q is the product of their verb and argument similarity scores:

$$S_P(p_a, p_q) = S_V(p_a, p_q) \times S_A(p_a, p_q)$$

This scoring mechanism has been designed to be flexible and robust to parsing errors in order to maximize the recall of the answer extraction. The idea of using a Jaccard coefficient to measure the similarity of all arguments as a whole was introduced in [9]. It takes into account that SRL systems often fail to assign the correct semantic roles to the arguments, which makes a per-argument comparison infeasible. We extended this idea to perform a fuzzy matching not only for arguments but also at the level of terms.

If the confidence score of a predicate is larger than 0, it is considered semantically similar to the question and one of the following strategies is used to extract factoid answers:

- If the answer type of the question is known, entities of the expected type are extracted from *all* arguments of the answer predicate. This takes into account that SRL systems often mislabel the arguments.
- If the answer type is unknown and the answer predicate has an argument with the role that is missing in the question, this argument is extracted as an answer.

The confidence score of an answer is the sum of the confidence scores of all the predicates it was extracted from.

3.5. Score Normalization and Combination

The answer candidates retrieved with this semantic approach are combined with candidates from our answer type based and pattern based extractors [2, 3]. Since these extraction techniques use different underlying scoring mechanisms that produce incomparable scores, it is necessary to normalize the confidence scores before merging the answers. We trained an AdaBoost classifier [25] that uses a decision tree as the underlying weak learner to classify answer candidates into correct and incorrect ones. The classifier is applied to unseen answer candidates and the probability of the positive class is used as a normalized confidence score. The following features were used to train the classifier:

- Score assigned to the candidate by the extractor.
- Answer extractor that found the candidate.
- Predicted answer type(s) of the question.
- Number of candidates from the same extractor.
- Minimum and maximum score over all candidates.

For one of our runs, we used a simple score combination scheme (known as CombMNZ [26]) to merge the normalized scores of a candidate found by multiple extractors: The combined score is the sum of the (normalized) scores from all extractors, multiplied by the number of extractors that found the answer. However, this technique was outperformed by the more general answer selection approach described in the following section, which was used for the remaining runs.

3.6. Answer Selection

The Answer Generator (AG) is responsible for selecting the correct answers from the candidates produced by our answer extractors. A statistical framework estimates the probability of an individual answer candidate given a set of *validation features* that predict its relevance according to external resources, and a number of *similarity features* that exploit redundancy among the answer candidates. The AG has been described in detail in our previous work [4].

To estimate the relevance of an answer candidate, we use four external resources. Gazetteers and WordNet are used in a knowledge-based approach (e.g. to check whether a candidate satisfies the relationship described in the question such as *IS-A(Shanghai, city)* or *IS-IN(Shanghai, China)*). The Web and Wikipedia are used in a data-driven approach. For instance, if there is a Wikipedia document whose title matches the answer candidate, the document is analyzed to obtain a tf-idf score, which is used as a relevance feature. Web snippets are used to calculate a word distance between an answer candidate and question keywords.

To identify similar answer candidates found by different extractors, we combine various syntactic and semantic similarity features. String distance metrics such as the Levenshtein distance and the cosine similarity are used to measure the syntactic similarity of answer candidates. A database of synonyms was generated from WordNet, the CIA World Factbook and Wikipedia.

As we use three different approaches to extract answer candidates, and each extractor sometimes produces more than 100 candidates, the AG only estimates probabilities for the top 50 answer candidates from each extractor. The answer candidates are then reranked according to these probabilities. For factoid questions, the highest ranked candidate is chosen as the final answer.

For list questions, we return all candidates with estimated probabilities of at least 25% of the probability of the top answer. In our experiments on previous TREC questions, we first attempted to maximize the F1 score by returning all candidates with probabilities of at least 0.5. However, the probabilities assigned by the AG turned out to be rather unreliable estimates of the correctness of an answer candidate because

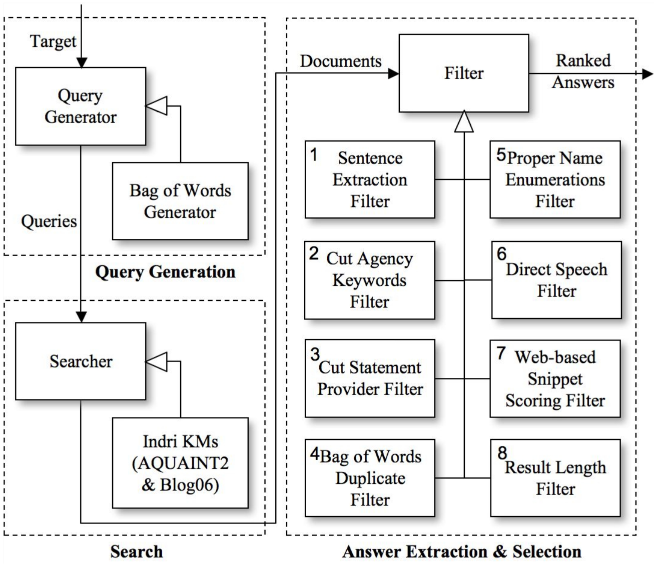
Target
Query Generator
Bag of Words Generator
Queries
Query Generation
Searcher
Indri KMs (AQUAINT2 & Blog06)
Search
Documents
Filter
Ranked Answers
1 Sentence Extraction Filter
2 Cut Agency Keywords Filter
3 Cut Statement Provider Filter
4 Bag of Words Duplicate Filter
5 Proper Name Enumerations Filter
6 Direct Speech Filter
7 Web-based Snippet Scoring Filter
8 Result Length Filter
Answer Extraction & Selection

Fig. 2. Pipeline layout for the 'other' questions.

of the high variance of the original scores from our answer extractors. For instance, the score of the top answer was sometimes very low, which resulted in estimated probabilities below the threshold of 0.5.

4. 'OTHER' QUESTIONS

Our approach for the 'other' questions uses answer projection from the Web onto the corpora, picking up the idea of Kaisser et al. [21]. However, we use Wikipedia in addition to Google to retrieve terms that are important in the context of the target. A Google query can be ambiguous and the results may be unrelated to the target, while we can retrieve information from Wikipedia unambiguously by searching for an article on the target. We furthermore assume that in the online encyclopedia, users make sure the most relevant information on the subject (target) is given in a concise and concentrated fashion with little noise. For targets not found in Wikipedia, we extract terms from Google snippets as a fallback solution.

For selecting the relevant snippets from the corpus, we further deploy several of the filtering techniques introduced last year [2] to eliminate nonsensical, redundant or irrelevant information. The modular architecture of Ephyra enabled us to evaluate various combinations of filters, parameterizations and orders. In the following we describe the key components of our pipeline, illustrated in Figure 2.

4.1. Snippet Retrieval

We first retrieve whole paragraphs from the corpora, using the Indri IR engine from the Lemur toolkit [24]. As the initial set of passages can be quite large, we run them through a number of filtering mechanisms before selecting answer candidates:

Reduction of Snippet Size. In order to allow more fine-grained filtering operations and to increase the precision of our answers, we reduce the size of the passages as much as possible. Previously reported results [27, 2], and an analysis of the snippets that were judged vital or OK in past TREC evaluations, led us to the conclusion that the most effective answers are sentences or sentence fragments. Thus we split paragraphs into sentences, and we further segment long sentences into their atomic clauses (Figure 2, Filter 1). This helps us to deal with additional material outside these clauses, such as explanatory prefixes (e.g. "PARIS (France) AFP ...") and indirect speech (e.g. "Secretary Gates said that ..."). By retaining only clauses that contribute relevant information we further increase the precision (Figure 2, Filters 2 & 3).

Elimination of Redundancy. In order to minimize the time required for the Web-based scoring process, duplicate snippets are eliminated. Semantic duplicates, i.e. snippets providing the same information with slightly different wording, are detected with a bag-of-words comparison mechanism, which stems content words and ignores stop words in the similarity calculation (Figure 2, Filter 4).

Elimination of Useless Snippets. The snippets from the AQUAINT2 corpus turned out to include two special types of likely useless snippets, which mainly appear if the target is a person's name. In particular, these include long lists of proper names without any further information, and snippets that consist of a person's statement in direct speech. We filter out lists of proper names based on the observation that most of their tokens are either stop words, or parts of a proper name and thus capitalized (Figure 2, Filter 5). Our experiments on previous TREC questions proved that the risk of useful snippets being lost to this filter is very low. Direct speech formulations citing a person's statement sometimes contain useful information, but in general they deliver opinions rather than facts, so we decided to also filter them out (Figure 2, Filter 6).

4.2. Snippet Selection

To select the actual answer snippets from the ones returned by the retrieval and filtering step, we adopted the Web-based approach proposed in [21] with Wikipedia as an additional information source. We assume retrieving an article from Wikipedia is considerably less ambiguous than retrieving text snippets from a Web search engine using the target as a keyword query. Only if we do not find a Wikipedia article, we use Google as a fallback. Experimental results on past TREC data encouraged us to use this combined approach. The complete scoring process consists of several steps:

4.2.1. Query Generation

In order to become less vulnerable to different possible ways of naming the target, we generate several different queries from the target string, as proposed in [21]. In particular, the generation process comprises two steps: First, we identify the type of the target, using the Stanford NE Recognizer [22]. Second, we vary the target string using several modification rules that depend on the target type:

- If the target contains a part enclosed in brackets, we use the part inside and the part(s) outside as individual queries.
- If the target contains a proper name, we use it as an individual query.
- If the target is an organization, we produce variations with and without determiner, and we produce an acronym query from the organization's name.
- For the Google fallback, we also generate quoted versions of all queries that consist of a proper name.

4.2.2. *Web Term Retrieval*

Since Wikipedia is our primary source of Web terms, we first try to retrieve a Wikipedia article on the target. If that succeeds, we do not retrieve any further Web documents. Otherwise, we try to retrieve articles on the parts we extracted from the target during the query generation. If this fails as well, we obtain the top 100 snippets from Google for each of our generated queries. Once we have obtained all necessary Web documents according to the policy above, we extract all the terms from them and count their frequencies.

4.2.3. *Snippet Scoring*

Our scoring mechanism for the corpus snippets uses the general method of score computation reported by Kaisser et al. [21] (Figure 2, Filter 7), with the following refinements: (a) A count decrease parameter indicates how the counter of a term is decreased after it has contributed to the score of a snippet that is actually selected. (b) We do not simply score the snippets using the terms they contain, because this would favor longer snippets over shorter ones. Instead we normalize the score of a snippet using the logarithm of the number of terms it contains. (c) We also found that using the plain count of Web terms for computing the scores of the corpus snippets over-weights common terms, while under-weighting more specific terms. To compensate for this effect, we normalize the term counters by the logarithm of the term's global frequency, which we obtain from a dictionary.

4.2.4. *Answer Length*

To make sure our answer snippets do not exceed the maximum of 7000 characters, we finally apply a filter that drops all snippets after the top ones have a combined length of some threshold ≤ 7000 (Figure 2, Filter 8). The relatively low impact of precision might encourage to return the maximum allowed number of characters, but our experiments with previous TREC targets revealed that with almost every parameter combination for the scoring, the optimal total length of the answer was 3000 characters. Consequently, we used 3000 as the cutoff length.

5. EVALUATION RESULTS

We submitted 3 runs, differing in the answer selection approach and the corpora used for answer projection (factoid and list questions) and to extract information nuggets ('other' questions). *Ephyra1* exclusively used the AQUAINT2 corpus, while *Ephyra3* combined the AQUAINT2 and Blog06 corpora and treated them as a single knowledge source. *Ephyra2* deployed both corpora to find supporting documents for factoid and list questions, but restricted the nugget extraction for the 'other' questions to the AQUAINT2 corpus. In the runs *Ephyra1* and *Ephyra3* the Answer Generator (AG, cf. Section 3.6) was used to select and combine candidate answers. *Ephyra2* applied a simple score combination approach to merge candidates from different extractors (cf. Section 3.5). Figure 4 shows our evaluation results and compares them to the median over all 51 runs.

The best setup for factoid and list questions (*Ephyra3*) projected the candidate answers found in the Web onto both the AQUAINT2 and the Blog06 corpus and deployed the AG to generate the final list of ranked answers. A comparison to the runs *Ephyra1* and *Ephyra2* shows to what extend the Blog06 corpus and the AG improved the overall performance.

For the 'other' questions, it proved most effective to extract information nuggets from the AQUAINT2 corpus only. *Ephyra1* and *Ephyra2* were identical runs that restricted the search to the newswire text. The nuggets extracted from the Blog06 corpus in the run *Ephyra3* rarely contained relevant information on the target, but often they were meaningless sentence fragments or not even natural language phrases.

6. FUTURE WORK

A major bottleneck of the previously described semantic approach for question analysis and answer extraction is the coverage and reliability of the semantic parser. It has been shown that by integrating multiple semantic role labeling (SRL) systems, the robustness can be improved significantly [28]. A combination of SRL systems is particularly beneficial if the systems use different syntactic parsers [29]. Yet there remains a significant portion of questions and answer sentences with semantic structures that do not fit into the schema of predicate verbs and arguments. It would therefore be desirable to cover a wider range of semantic structures such as the semantic frames used in FrameNet [30].

	Ephyra1	Ephyra2	Ephyra3	Median (51 runs)
Corpora	AQUAINT2	AQUAINT2 Blog06 (factoid, list)	AQUAINT2, Blog06	-
Answer Generator	yes	no	yes	-
Unsupported (U)	28	21	23	-
Inexact (X)	18	18	23	-
Locally correct (L)	1	3	1	-
Factoid accuracy	0.206	0.203	**0.208**	0.131
List F_1	0.140	0.123	**0.144**	0.085
Other F_3 (pyramid score)	**0.189**	0.188	0.156	0.118
Average per-series score	**0.181**	0.171	0.172	0.108

Table 4. TREC 16 evaluation results.

Currently we do not pre-annotate the TREC corpora but we build a full-text index and use simple boolean queries for both the Web search and to retrieve text passages from the corpora. By annotating the corpora with semantic information and integrating these annotations in the index, we could (a) improve the runtime performance and (b) formulate structured queries that combine evidence from different documents. For instance, one could search for all organizations X in the corpora that satisfy the constraints imposed by the predicate-argument structures *based(ARG1: X, ARGM-LOC: Japan)* and *manufacture(ARG0: X, ARG1: SUV)* to obtain a list of Japanese car makers that offer SUVs.

The Answer Generator can be further improved by incorporating additional features to validate individual answer candidates and to identify semantically similar answers among the candidates. We will also need to conduct additional experiments to determine a more flexible cutoff strategy for list questions. Furthermore, we consider extending the AG to perform answer selection not only for factoid answers but also for more complex answers such as the definitional phrases retrieved for the 'other' questions.

Extensions to our answer type classifier would include analyzing precisely how the knowledge encoded by the rules augments the knowledge contained in the training data, in the hope of discovering whether it helps to resolve noise, cover gaps, or do both. It also seems important to address questions such as: How can we most effectively cover gaps in the data with hand-crafted rules? How can we most effectively cover gaps in the hand-crafted rules by learning from data sets that target specific, variable aspects of language? Which of these strategies yields the best performance? Are there general linguistic principles that can guide the decision of whether to treat a particular phenomenon using rules or data?

Experiments on past TREC data revealed that the effectiveness of our answers to an 'other' question highly depends on the choice of (a) the source of Web terms, (b) the parameter values for the scoring process, and (c) the cutoff length. Therefore we need to thoroughly investigate characteristics of the individual targets and find categories of targets for which an individual combination of term source and parameter values yields the optimal result. We then need to find criteria for assigning each target to a category, so that we can choose a setup for our scoring mechanism accordingly. Furthermore, additional filtering techniques and more robust parsing and sentence segmentation approaches will be required to extract useful information nuggets from resources with a high content of noise, such as the Blog06 corpus.

Acknowledgements

We would like to thank the interACT Exchange Program for making this collaboration between Carnegie Mellon University and Universität Karlsruhe possible.
This work was supported in part by the AQUAINT program under grant NBCHC040164.

7. REFERENCES

[1] H.T. Dang, J. Lin, and D. Kelly, "Overview of the TREC 2006 question answering track," *Proceedings of the Fifteenth Text REtrieval Conference*, 2006.

[2] N. Schlaefer, P. Gieselmann, and G. Sautter, "The Ephyra QA system at TREC 2006," *Proceedings of the Fifteenth Text REtrieval Conference*, 2006.

[3] N. Schlaefer, P. Gieselmann, T. Schaaf, and A. Waibel, "A pattern learning approach to question answering within the Ephyra framework," *Proceedings of the Ninth International Conference on TEXT, SPEECH and DIALOGUE*, 2006.

[4] J. Ko, L. Si, and E. Nyberg, "A probabilistic framework for answer selection in question answering," *Proceedings of NAACL-HLT*, 2007.

[5] D. Moldovan and A. Novischi, "Lexical chains for question answering," *Proceedings of COLING 2002, pp.674-680*, 2002.

[6] C. Fellbaum, “WordNet: An electronic lexical database,” *The MIT Press*, 1998.

[7] E. Nyberg, T. Mitamura, R. Frederking, V. Pedro, M.W. Bilotti, A. Schlaikjer, and K. Hannan, “Extending the JAVELIN QA system with domain semantics,” *Proceedings of the Question Answering in Restricted Domains Workshop at AAAI*, 2005.

[8] B. Van Durme, Y. Huang, A. Kupsc, and E. Nyberg, “Towards light semantic processing for question answering,” *Proceedings of HLT-NAACL*, 2003.

[9] R. Sun, J. Jiang, Y.F. Tan, H. Cui, T.-S. Chua, and M.-Y. Kan, “Using syntactic and semantic relation analysis in question answering,” *Proceedings of the Fourteenth Text REtrieval Conference*, 2005.

[10] S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky, “Shallow semantic parsing using support vector machines,” *Proceedings of HLT-NAACL*, 2004.

[11] J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel, “TREC 2002 QA at BBN: Answer selection and confidence estimation,” *Proceedings of the Eleventh Text REtrieval Conference*, 2002.

[12] J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, and D. Ferrucci, “A multi-strategy and multi-source approach to question answering,” *Proceedings of the Eleventh Text REtrieval Conference*, 2002.

[13] D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano, “Cogex: A logic prover for question answering,” *Proceedings of HLT-NAACL*, 2003.

[14] B. Magnini, M. Negri, R. Pervete, and H. Tanev, “Comparing statistical and content-based techniques for answer validation on the Web,” *Proceedings of the VIII Convegno AI*IA*, 2002.

[15] C. Clarke, G. Cormack, and T. Lynam, “Exploiting redundancy in question answering,” *Proceedings of ACM SIGIR Conference on Research and Development on Information Retrieval*, 2001.

[16] C. Kwok, O. Etzioni, and D. Weld, “Scaling question answering to the Web,” *Proceedings of the Tenth Text REtrieval Conference*, 2001.

[17] M.M. Soubbotin and S.M. Soubbotin, “Patterns of potential answer expressions as clues to the right answer,” *Proceedings of the Tenth Text REtrieval Conference*, 2001.

[18] T. Solorio, M. Perez-Coutino, M. Montes y Gomez, L. Villasenor-Pineda, and A. Lopez-Lopez, “A language independent method for question classification,” *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, 2004.

[19] X. Li and D. Roth, “Learning question classifiers: the role of semantic information,” *Natural Language Engineering*, 2005.

[20] M. Day, C.W. Lee, S.H. Wu, C.S. Ong, and W.L. Hsu, “An integrated knowledge-based and machine learning approach for Chinese question classification,” *Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLPKE)*, 2005.

[21] M. Kaisser, S. Scheible, and B. Webber, “Experiments at the University of Edinburgh for the TREC 2006 QA track,” *Proceedings of the Fifteenth Text REtrieval Conference*, 2006.

[22] J.R. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by Gibbs sampling,” *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

[23] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Buneascu, R. Girju, V. Rus, and P. Morarescu, “FALCON: Boosting knowledge for answer engines,” *Proceedings of the Ninth Text REtrieval Conference*, 2000.

[24] “Lemur toolkit for language modeling and information retrieval,” http://www.lemurproject.org/.

[25] R.E. Schapire, “The boosting approach to machine learning: An overview,” *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[26] E. Fox and J. Shaw, “Combination of multiple searches,” *Proceedings of the Second Text REtrieval Conference*, 1993.

[27] R. Gaizauskas, M.A. Greenwood, H. Harkema, M. Hepple, H. Saggion, and A. Sanka, “The University of Sheffield’s TREC 2005 Q&A experiments,” *Proceedings of the Fourteenth Text REtrieval Conference*, 2005.

[28] L. Màrquez, M. Surdeanu, P. Comas, and J. Turmo, “A robust combination strategy for semantic role labeling,” *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 2005.

[29] V. Punyakanok, D. Roth, and W. Yih, “The necessity of syntactic parsing for semantic role labeling,” *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

[30] J. Ruppenhofer, M. Ellsworth, M.R.L. Petruck, C.R. Johnson, and J. Scheffczyk, “FrameNet II: Extended theory and practice,” http://framenet.icsi.berkeley.edu/.