Building a Foundation System for Producing Short Answers to Factual Questions

Sameer S. Pradhan[*], **Gabriel Illouz**[†§], **Sasha J. Blair-Goldensohn**[†],
Andrew Hazen Schlaikjer[†], **Valerie Krugler**[*], **Elena Filatova**[†], **Pablo A. Duboue**[†], **Hong Yu**[†],
Rebecca J. Passonneau[†], **Steven Bethard**[*], **Vasileios Hatzivassiloglou**[†], **Wayne Ward**[*],
Dan Jurafsky[*], **Kathleen R. McKeown**[†], and **James H. Martin**[*]

[*]Center for Spoken Language Research
University of Colorado
Boulder, CO 80309, USA

[†]Department of Computer Science
Columbia University
New York, NY 10027, USA

Abstract

In this paper we describe question answering research being pursued as a joint project between Columbia University and the University of Colorado at Boulder as part of ARDA's AQUAINT program. As a foundation for targeting complex questions involving opinions, events, and paragraph-length answers, we recently built two systems for answering short factual questions. We submitted results from the two systems to TREC's Q&A track, and the bulk of this paper describes the methods used in building each system and the results obtained. We conclude by discussing current work aiming at combining modules from the two systems in a unified, more accurate system and adding capabilities for producing complex answers in addition to short ones.

1 Introduction

The Department of Computer Science at Columbia University (CUCS) and the Center for Spoken Language Research (CSLR) at the University of Colorado at Boulder are collaborating to develop new technologies for question answering. This project is supported by the ARDA AQUAINT (Advanced QUestion Answering for INTelligence) program. The project plans to integrate robust semantics, event detection, information fusion, and summarization technologies to enable a multimedia question answering system. The goal is to develop a system capable of answering complex questions; these are questions that require interacting with the user to refine and clarify the context of the question, whose answer may be located in non-homogeneous databases of speech and text, and for which presenting the answer requires combining and summarizing information from multiple sources and over time. Generating a satisfactory answer to complex questions requires the ability to collect all relevant answers from multiple documents in different media, weigh their relative importance, and generate a coherent summary of the multiple facts and opinions reported. In order to achieve these goals, we are developing and integrating four core technologies: semantic annotation (CSLR), context management (CSLR), event recognition and information tracking (Columbia), and information fusion and summary generation (Columbia).

Prior to the start of this project, neither site had developed a complete question answering system, so we had to build the foundation components and put them together in approximately six months. We elected to build two independent systems as a first step, one at each site, rather than attempting to integrate components across the two sites in

[§]Currently at LIMSI-CNRS, Orsay, France.

such a short period. This gave us the added benefit that each site was able to focus more of their effort on specific components they were interested in, and avoided the need for developing complex protocols for communication between the modules across the sites.

The paper focuses on our development of our foundation question answering systems for TREC, processing factual questions only and producing short answers. Most of the remainder of the paper (Sections 2 and 3) discusses the two architectures we developed, ways that each departs from standard question answering methodology, and our results on this year's TREC questions. We submitted the results of these two architectures as runs 1 and 2 with the tag `cuaq`. We conclude with a discussion of our current integration effort and ongoing work on adding advanced components for handling additional question types to our foundation system.

2 System A — The Boulder System

The novel feature of our approach in System A is the use of shallow semantic representations to enhance potential answer identification. Most successful systems first identify a list of potential answer candidates using pure word-based metrics. Syntactic and semantic information at varying granularity is then used to re-rank those candidates [10, 11]. However, most of these semantic units are quite specific in what they label. We identify a small set of *thematic roles*—viz., *agent*, *patient*, *manner*, *degree*, *cause*, *result*, *location*, *temporal*, *force*, *goal*, *path*, *percept*, *proposition*, *source*, *state*, and *topic*—in the candidate answer sentences, using a statistical classifier [9]. The classifier is trained on the FrameNet database [2].

2.1 Architecture

The following sequence of actions will be taken in response to an input query:

1. Classify the question according to type by identifying the Named Entity and Thematic Role of the expected answer type. This also defines a set of answer type patterns, and includes named entity tagging and parsing the question for thematic roles.
2. Identify the focus, i.e., certain salient words or phrases in the question that are very likely to be present in the answer string in one form or the other.
3. Extract a set of query words from the question, and apply semantic expansion to them.
4. Submit the query words to the `mg` (Managing Gigabytes) [15] IR engine and get back a rank-ordered set of documents.
5. Keep the top M (approximately 500) documents and prune the rest.
6. Segment documents into paragraphs and prune all but the top N paragraphs (approximately 2,500).
7. Generate scoring features for the paragraphs, including named entity tagging and parsing of paragraphs to add thematic roles.
8. Re-rank documents based on the set of features that we compute, including answer type patterns. Some of the answer type patterns are based on the semantic labels.
9. Compute for each paragraph a confidence measure that it contains some relevant information. This includes N-Best count as one of the features.
10. Send tagged paragraphs that exceed a confidence threshold for extraction of the short answer required for TREC.

For the problem of question answering, we are more concerned with precision than recall, so we have to be careful in expanding the query words to get answers that are expressed in words quite different from the ones mentioned in the question. Semantic expansion will be performed when the system's confidence in the best candidate answer string without expansion is found to be below a certain threshold. Our mechanism for expansion is:

a. Submit original query words to IR engine and get back a rank-ordered set of documents.

b. Generate set of target words from top k documents based on the *idf* values of the words. We are experimenting with k in the range of 1–100.
c. Generate a set of target words from WordNet [8] synsets of original keywords.
d. Take the intersection of the two sets and add to the keyword set.

2.2 Features

Answer Identification We now discuss the features used for ranking the documents. The features are roughly ordered by decreasing salience.

- **Answer type** — In order to extract the answer to a question, the system needs to identify the expected answer type. Answers for short answer questions generally can be categorized as named entities and/or propositions. Summary information is often required for descriptive and definition questions. The system classifies the answer type by two features: 1) named entity class and 2) thematic role. Named entity class specifies one (or more) of 56 classes as the named entity class of the answer. We use 54 real named entity classes, one class representing the case where the answer is expected to be a named entity but one not known to the system, and one class for cases where the answer is not expected to be a named entity. The thematic role class identifies the thematic role in which the potential answer would tend to be found in the answering sentence.

- **Answer surface patterns** — Once the question type is known the next step is to identify candidate answers. One technique we use is based on generating a set of expected surface patterns for the answer. Sentences or snippets matching these patterns would get better scores than ones that did not. The patterns specify word and named entity-based regular expressions that are derived from a large corpus annotated with named entities; (simplified) examples include:

  1. Some common question types, e.g., [<PERSON_DESCRIPTION><PERSON_NAME>] for questions like "Who is <PERSON_NAME>?"; [<ORGANIZATION>, <CITY>, <STATE>, <COUNTRY>] for questions asking about the location or address of an organization.
  2. Likely re-phrasings of the question, e.g., [<PERSON> invented <PRODUCT>] for questions like "Who was the inventor of <PRODUCT>?"
  3. Occurrence statistics of the pattern in the corpus, e.g., [<PERSON> (<YEAR>–<YEAR>)] for birth dates of people.

- **Named entities in answer** — In the case of questions that expect a specific named entity (including the unknown named entity type) as the answer, candidates that do not contain that named entity are penalized. In the case that the answer is expected to be an unknown named entity, then candidates that contain an untagged sequence of capitalized words (a strong indicator of unknown named entities) are preferred.

- **Presence of focus word** — The presence of the focus word is an important feature for the overall score. For our purposes, a focus word is a word in the question that, or its synonym, is very likely to appear in the sentence that contains the answer.

- **Thematic role patterns** — While surface patterns for answers can provide valuable information when a match is found, the specific nature of the patterns and the limited occurrences of the answer string within the reformulations obtainable from the question do not always guarantee a surface pattern match. We also provide a more general set of expected answer patterns based on thematic roles. We expect that these patterns will have higher coverage than the more specific surface patterns. This feature scores sentences based on the presence of expected thematic roles and named entities existing in specific thematic roles.

  Thematic role patterns help identify false positive answer candidates and help extract the exact answer boundary from the string This can be illustrated with the example in Figure 1

Question: Who assassinated President McKinley?

Parse: [$_{role=agent}$ Who] [$_{target}$ *assassinated*] [$_{role=patient}$ [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **McKinley**]]?

Keywords: assassinated President McKinley

Answer named entity (ne) Type: Person

Answer thematic role (role) Type: Agent of target synonymous with "assassinated"

Thematic role pattern: [$_{role=agent}$ [$_{ne=person}$ `ANSWER`] $\wedge$ [$_{target}$ `synonym_of(assassinated)`] $\wedge$ [$_{role=patient}$ [$_{ne=person}$ `reference_to(President McKinley)`]]

`This is one of possibly more than one patterns that will be applied to the answer candidates.`

False Positives:

`Note: The sentence number indicates the final rank of that sentence in the returns, without using thematic role patterns.`

1. In [$_{ne=date}$ **1904**], [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **Theodore Roosevelt**], who had succeeded the [$_{target}$ *assassinated*] [$_{role=patient}$ [$_{ne=person}$ **William McKinley**]], was elected to a term in his own right as he defeated [$_{ne=person\_description}$ **Democrat**] [$_{ne=person}$ **Alton B. Parker**].

4. [$_{ne=person}$ **Hanna**]'s worst fears were realized when [$_{role=patient}$ [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **William McKinley**]] was [$_{target}$ *assassinated*], but the country did rather well under TR's leadership anyway.

5. [$_{ne=person}$ **Roosevelt**] became president after [$_{role=patient}$ [$_{ne=person}$ **William McKinley**]] was [$_{target}$ *assassinated*] [$_{role=temporal}$ in [$_{ne=date}$ **1901**]] and served until [$_{ne=date}$ **1909**].

Correct Answer:

8. [$_{role=temporal}$ In [$_{ne=date}$ **1901**]], [$_{role=patient}$ [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **William McKinley**]] was [$_{target}$ *shot*] [$_{role=agent}$ by [$_{ne=person\_description}$ **anarchist**] [$_{ne=person}$ **Leon Czolgosz**]] [$_{role=location}$ at the [$_{ne=event}$ **Pan-American Exposition**] in [$_{ne=us\_city}$ **Buffalo**] , [$_{ne=us\_state}$ **N.Y.**]] [$_{ne=person}$ **McKinley**] died [$_{ne=date}$ **eight days later**].

Figure 1: An example where thematic role patterns help constraint the correct answer among competing candidates. All the named entities, but only roles pertaining to the *target* predicate are marked in the sentences.

- ▷ **Okapi scoring** — This is the Okapi BM25[14] score assigned by the information retrieval engine to the paragraph extracted in the very beginning.
- ▷ **N-gram** — Another feature that we use is based on the length of the longest n-gram (sequence of contiguous words) in the candidate answer sentences after removing the stopwords from both the question and the answer.
- ▷ **Case match** — Documents with words in the same case as in the question tend to be more relevant than those that have different case, so the former are given a relatively higher score. This is because capitalization helps disambiguate named entities from common words (at least in carefully written queries).

Confidence Annotation Once the likely answer candidates (paragraphs or sentences) are extracted for a question we need to estimate the likelihood of those being *good* answers. To do this, we have a scheme that annotates each with

# wrong (W)	# unsupported (U)	# inexact (X)	# right (R)	CW Score	NIL Precision	NIL Recall
411	7	3	79	0.226	0 / 9 = 0.000	0/46 = 0.00

Table 1: System A (Boulder) results; numbers are out of 500 questions.

some level of confidence.

We use a weighted linear combination of N-Best answer count, Named entity class of the answer, and the N-gram length features to calculate the degree of confidence.

2.3 Current Implementation

In this section we discuss the state of the current implementation for system A, which was used to produce the first run submitted to the TREC 2002 question answering track.

TREC-2002 Database The text database comprises non-stemmed, case-folded indexing of the TREC/AQUAINT corpus using a modified version of the `mg` (Managing Gigabytes) search engine [15] that incorporates the Okapi BM25 ranking scheme [14] and an expanded set of characters forming an indexed unit—so as to accommodate hyphenated words, URLs, emails etc. Each indexed document is a collection of segmented sentences forming a paragraph in the original corpus.

Answer Identification We currently use a rule-based question classifier that identifies the named entity type and thematic role of the expected answer to each question. For each query, the top N (2500, based on 85% recall threshold) ranked paragraphs are retrieved for processing. A set of documents are carried over from the list of documents retrieved by the IR engine, using gradually diluted boolean filters, until there are no more keywords to drop, or the cumulation of filtered documents exceeds a threshold of n (currently set to 10). We call this the *boolean peel-off* strategy. The answer named entity type is used to filter out documents that do not contain the required named entity type.

Answer Extraction For questions in which the answer is a specific named entity or a thematic role, if the top ranking sentence contains only one instance of that element, that instance is returned as the answer. In many cases, however, there is more than one element of the predicted type. In such cases, the system selects the element with the shortest average distance from each of the query words present in that snippet. There are penalties added for some punctuation symbols like commas, semi-colons, hyphens etc. In cases when the required answer is not a known named entity, the answer extractor tries to find a sequence of capitalized words that could be a probable named entity. In case the expected answer is not a named entity, and the thematic role cannot be identified without much ambiguity, then the system tries to find an apposition near the question words, and extracts that as the answer. An example would be definition questions, of the style "What is X?". Failing to get any of the above, the system replies that it does not have an answer to that particular question in the given corpus.

Confidence Annotation The prior probability of correct answers for a particular question type, along with the n-best count, are used to assign a confidence measure. The system then additionally tests whether the candidate at rank two has more counts in the top ten candidates than the candidate at rank one. If so, it is promoted to rank one.

2.4 Results

The results of System A on the TREC-2002 test set are presented in Table 1. These results are consistent with what we expected based on our TREC 9 and 10 development test set. Note that we answered "no answer" or NIL only if no answer was returned by our standard algorithm, which was rarely.

3 System B — The Columbia System

3.1 Overview of System Operation

Our second foundation system, developed at Columbia University, uses a hybrid architecture that evolved from an initial version that relied solely on the web as a source of answers. We focused on query expansion and candidate-answer scoring in order to perform search in parallel over open (the web) and closed (TREC) collections and then combine the results. For example, as described in more detail in Section 3.5, we experimented with a strategy that interleaves information learned from the web search to re-query the TREC document set.

In our initial, web-based system we adopted two working assumptions: First, given the size and redundancy of the web, a database of paraphrase patterns where more specific patterns are prioritized would be both necessary and effective for finding relevant documents (i.e., we aimed at precision rather than recall). Second, scores for candidate answers would be a composite function of attributes of the query formulation process, and of the space of candidate answers. For example, more specific queries have higher default scores, and an answer string retrieved from multiple documents is weighted higher than an answer string that appears in relatively few documents. Finally, we assumed that the question of how best to capitalize on these two assumptions would be primarily an empirical one. Thus questions that are superficially similar might require rather distinct queries. For example, questions that contain a noun denoting a leadership position, as in "Who was the first commander of FLEET X?" and "Who was the first coach of TEAM Y?" might both benefit from query expansion rules in which the related verbs appear (e.g., "<TERM> commanded FLEET X" and "<TERM> coached TEAM Y"), but this would not be the case for the structurally similar question "Who was the first general of FORCE Z?".

The remainder of this section documents four key modules of System B's architecture. There is a pipeline of three modules responsible for distinct phases of the query expansion process: paraphrasing of patterns derived from the question string (Section 3.2 below); modification of queries, e.g., by inserting terms likely to occur with the answer ([1]; Section 3.3); and term prioritization (Section 3.4). At each phase of the query expansion process we modify according to the results of that phase scoring weights that guide how likely that version of the query is to lead to the correct answer. Finally, after a candidate answer set has been assembled, these weights are assembled into a single score (Section 3.5). If, as in the case of our second run submitted to TREC, candidate answer sets for a given question come from distinct document collections, the source collection is considered in ranking potential answers (Section 3.5).

We conclude this section with a brief discussion of the results obtained on the data and questions of the 2002 TREC Q&A track.

3.2 Question Paraphrasing

In order to generate queries providing high precision coverage of the answer space for a given question, custom rules were developed providing a mapping from a given question type to a set of paraphrasing patterns which would generate alternative queries. For example, the question string "Where is the Hudson River located?" may result in the generation of queries including "Hudson River is located in", "Hudson River is located", "Hudson River is in the", "Hudson River is near", and "Hudson River in". Since we often do not have specific information about the question target, our paraphrasing rules allow changes on articles (definite, indefinite, or no article), number, and function words to maximize our coverage of the collection.

A two-level scoring scheme was implemented for these queries whereby each was scored based on the specificity of its query string as well as that of the paraphrasing pattern that generated it. Specificity here is defined by the length of the query string (or the length of the shortest possible generated query string in the case of the paraphrasing patterns); longer queries typically return fewer results than shorter (more general) queries. These query scores are used to aid the scoring and subsequent ranking of the returned results along with other factors that rate the results themselves (see Section 3.5).

3.3 Query Modification

Knowing the type of the question can be very helpful not only for defining the type of the answer but also for better targeting of the search of the documents which might contain a potential answer. This process can be viewed as connecting the answer and question spaces [3]. A question like "What is the length of the Amazon?" produces no useful results among the top 20 when submitted to Google, because among the two content words, "length" is not likely to appear directly in the answer, and "Amazon" is highly ambiguous (the river, the mythological female warrior, the online company, to name just the more common senses). "Length" is a very good *question* word, but needs to be mapped to corresponding *answer* words such as "miles" and "kilometers". If we expand the query by adding either *km* or *mi*, the first hits returned by Google are about the Amazon river and contain the answer.

In our TREC 2002 system we used query expansion only for questions that required answers consisting of a number plus a measurement unit. All the questions that required a number as an answer were categorized under the general type `NUMBER` by our question classifier. We built a further classifier that translated some of the question words to subtypes of `NUMBER`, namely `DISTANCE`, `WEIGHT`, `SPEED`, `TEMPERATURE`, `MONEY`, and `OTHER`. The classifier was constructed by training using RIPPER [6] to produce a set of rules. For each of the questions classified into the above subtypes, the classifier returns an automatically compiled list of words expressing the appropriate measurement units, which were added to the query.

3.4 Query Prioritization

Query prioritization scores queries so that those with highest predicted answer precision (i.e., number of documents retrieved by a query that contain the correct answer(s) divided by the total number of documents retrieved) will be attempted first during document retrieval. This is because, first, our paraphrasing and modification mechanisms can generate thousands of queries and it is impractical to submit all of those to the search engine, and, second, because we can tell that some query rewriting mechanisms are more likely to be accurate than others.

For example, given the question "How many soldiers died in World War II?", our system will generate many queries, using the paraphrasing (Section 3.2) and modification (Section 3.3) rules mentioned above. The generated queries include "World War II" (q_a) and "soldiers" (q_b). We want query prioritization to assign q_a a higher score since it clearly is more specific and relevant to this question, and thus is likely to have a higher predicted answer precision.

Unlike previous TREC systems [7, 12], which mainly applied heuristics for query prioritization, we empirically built our query prioritization algorithm using statistics collected over previous TREC question-answer pairs. We analyzed the relation between a query term (see Sections 3.2 and 3.3 for query term generation) and its answer precision. We considered various features of a query term, including the term's syntactic role (e.g., noun phrase, verb phrase, and head noun), morphological features (e.g., upper case for proper noun), and inverse document frequency (IDF). We found that IDF consistently reflected answer precision.

The query scoring algorithm for non-paraphrase-based queries relies on two functions:

- ▷ The term-scoring function T maps terms to term scores, where terms are strings of one or more words which are part of a query. The term score of a multiword term X is the product of the IDF values of the individual words forming the term. A minimum IDF value of 1.05 is used in this case to ensure that even common terms slightly boost the score of a multiword term. In addition, a suitably high value is used for words with unknown IDF values. We use IDF values computed over a large body of recent general news text.

- ▷ The query-scoring function Q maps a set of one or more terms (a query) to a query score. The query score for query Y is the product of the term scores of the query's component terms; given the definitionof term scores above, this means that a query's score is the product of the IDF values for all words in all phrases of the query.

As mentioned above, the query scores produced are used in document retrieval, answer selection, and answer ordering (for listing first the answers to questions where our system has the greatest confidence). For the latter purpose, scores must be normalized. This is because answer ordering requires us to compare answer confidence across answers to different questions; since part of our confidence is determined by the score of the query leading to the answer (see Section 3.5), we must be able to make a meaningful comparison between query scores for queries produced for two different questions. Therefore, scores for each set of queries produced for a given question are normalized using division by the highest query score for a query generated for that question. Thus, the highest scoring query produced for any question will have a score of 1, while lower scoring queries will have scores between 0 and 1.

It is important to note that our system scores any paraphrase-based query above any keyword-based query, irrespective of the functions mentioned above.[1] This is because the paraphrasing rules (Section 3.2) were hand-crafted and produced with an eye toward high precision. Thus, the scores produced by this module of the system were only used to differentiate between keyword-based queries.

3.5 Combining Evidence from Multiple Sources

Even when the answer must be found in a specific collection, as in the TREC evaluation, there are benefits in combining evidence from multiple collections: the answer can be found in a larger or broader collection, in which case the smaller collection can be searched again with that specific answer in mind; and, if no answer can be found in the smaller collection, the confidence in the answer from the larger collection can help determine whether "no answer present in the collection" (NIL) should be returned.

To this end, we have designed a general mechanism using *wrappers* that interface our system to different collections and/or search engines using a common API. We have implemented wrappers for the Google, Glimpse [13], and `mg` [15] search engines, and for TREC-11 we ran system B using a combination of Google on the web (a broad collection) and `mg` on the TREC/AQUAINT collection (the collection where the answer must be found).

Our system returns a list of answers extracted from each source (search engine and collection combination), which may include the same string multiple times. Each instance of an answer has an associated confidence score, which depends on the query used to retrieve the answer (see Sections 3.2 and 3.4), the confidence score returned by the search engine (not implemented for our TREC experiments), and the confidence of the answer extractor in selecting an appropriate phrase from the returned document. These instance scores are combined for each potential answer (i.e., across all instances where the same string is returned) using the following formula for computing the cumulative score after n instances of the same string have been processed

$$\text{cumulative\_score}_n = 1 - [(1 - \text{cumulative\_score}_{n-1}) \times (1 - \text{instance\_score}_n)]$$

which ensures that answers occurring multiple times are weighted higher, taking into account the evidence for them each time they are found.

An answer may be found in one or more sources. We employ the following algorithm for calculating a composite score based on the cumulative scores of each string proposed as a potential answer, from the web or the TREC collection. The algorithm distinguishes the following cases:

- ▹ No answers found in either collection. Since this probably represents a system failure (it is unlikely that a TREC question would not be answerable from both the web and the TREC collection), we return NIL with zero confidence.

- ▹ One or more answers found in the TREC collection, but no answers found from the Web. We return the best answer extracted from the TREC collection but depreciate its confidence by a constant factor because the coverage of the web would make it unlikely that no answers at all could be found there while the TREC collection would be able to provide one.

[1] Queries augmented with the query modification techniques discussed in Section 3.3 are considered keyword-based queries in this context.

- ▷ No answers found in the TREC collection, but one or more answers found from the Web. This is our *prima facie* case for a NIL answer; however, since often we got our answer from the web using a modified query, we re-query the TREC collection using the answer identified from the web. If that step succeeds, we report the answer with a combined confidence as in the next case below; otherwise we report NIL, with confidence equal to our confidence in the web answer.
- ▷ One or more answers found in each collection, and both top ranked answers are the same. We report that answer, with reinforced confidence according to the formula

$$\text{combined\_confidence} = 1 - [(1 - \text{confidence\_TREC}) \times (1 - \text{confidence\_web})]$$

- ▷ One or more answers found in each collection, but the top ranked answers are different. We report the TREC answer, but reduce its confidence by the formula

$$\text{combined\_confidence} = \text{confidence\_TREC} \times (1 - \text{confidence\_web})$$

3.6 Results

For the 500 questions in the TREC-11 question, System B produced 58 correct answers, 8 unsupported, 2 inexact, and 432 wrong. The system's unweighted precision is 11.6%, while its confidence-weighted score is 0.178, representing a significant boost over the unweighted precision. We produced a lot of NIL answers (195, or 39% of our total answers), which indicates that we were too conservative in failing to choose low-confidence answers found in the TREC collection. We did retrieve a third of the NIL answers present in the test set, but overall System B performed less well than System A which obtained a confidence-weighted score of 0.226 while producing very few NILs (1.8% of its total answers). On non-NIL answers, the two systems performance is closer (unweighted precision of 16.1% for system A and 14.1% for system B).

We attribute the lower performance of System B to two additional factors beyond the excessive production of NILs: First, we used a very simple extractor for selecting the answer out of the sentence where it was found, and the extractor failed to produce the correct phrase in a number of cases where we succeeded in finding the appropriate sentence. Second, our question classifier was not always successful in predicting the correct question type, producing no label for 77 (15.4%) of the questions. We performed significantly worse on those questions than in *who*, *when*, *where*, or *what* questions with well-extracted types.

4 Conclusion and Future Work

We have described two systems that handle questions with short, factual answers. These systems were developed in a very brief time frame, and are our first entry in the TREC Q&A track. They represent for us a starting point for an overall question answering architecture, to which we are adding additional capabilities. In the months since TREC, we have worked on developing detailed XML-based protocols for communication between modules in a common architecture. The system we are building by combining elements of the two systems discussed in this paper utilizes the best-performing components of each of the current systems. The modular architecture allows us to connect additional modules, and actual question answering is done in a distributed fashion, with most of question analysis done at Colorado and most of answer synthesis done at Columbia. Modules communicate with a central server via HTTP, while the architecture offers several communication interfaces at different levels of complexity (for example, one module may request only the high-level question type, while another may examine in detail the semantic parse). A web-based client provides a front end to the integrated system, allowing users in different locations to access the system.

Our research is moving towards questions with more complex answers, including opinions, events, definitions, and biographies. We recently completed a prototype module for processing definitional questions, and we are currently adding its functionality to our integrated system, so that for questions of the form "What is X?" we produce both a short, TREC-like, answer and an answer spanning several paragraphs. At the same time, we are continuing to tune individual components to enhance interoperability between the two sites; for example, we recently started re-focusing the semantic analysis (done at Colorado) on types of phrases that are likely to impact the processing of opinions and biographies (done at Columbia).

Acknowledgments

This work, as well as our broader research on question answering, is supported by ARDA AQUAINT contract MDA908-02-C-0008. We would like to thank Ralph Weischedel of BBN Systems and Technologies for giving us the named entity tagger (Identifinder), and Daniel Gildea for the FrameNet parser.

References

[1] Agichtein, E., Lawrence, S., and Gravano, L. "Learning Search Engine Specific Query Transformations for Question Answering". In *Proc. of the 10th International World-Wide Web Conference (WWW10)*, 2001.

[2] Baker, C., Fillmore, C., and Lowe, J. "The Berkeley FrameNet Project". In *Proceedings of the COLING-ACL*, Montreal, Canada, 1998.

[3] Berger, A., Caruana, R., Cohn, D., Freitag, D., and Mittal, V. "Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding". In *Proceedings of the 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July 2000.

[4] Bikel, D., Schwartz, R., and Weischedel, R.. "An Algorithm that Learns What's in a Name". *Machine Learning*, **34**:211–231, 1999.

[5] Brin, S. and Page, L. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *Computer Networks and ISDN Systems*, **30**(1–7):107–117, 1998.

[6] Cohen, W. W. "Fast Effective Rule Induction". In *Proceedings of the Twelfth International Machine Learning Conference (ML-95)*, 1995.

[7] Dumais, S., Banko, M., Brill, E., Lin, J., and Ng A. "Web question answering: Is more always better?". In *Proceedings of SIGIR-02*, pp. 291–298, 2002.

[8] Fellbaum, C., *editor*, "WordNet: An Electronic Lexical Database", MIT Press, 1998.

[9] Gildea, D. and Jurafsky, D. "Automatic Labeling of Semantic Roles". Technical Report TR-01-005, International Computer Science Institute, Berkeley, 2001.

[10] Harabagiu, S., Moldovan, D., *et al*. "Answering complex, list and context questions with LCC's Question-Answering Server". In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, pp. 355–361, Gaithersburg, Maryland, November 13-16, 2001.

[11] Hovy, E. and Hermjakob, U. "The Use of External Knowledge of Factoid QA". In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, pp. 644–652, Gaithersburg, Maryland, November 13-16, 2001.

[12] Lee, G. G., Seo, J., Lee, S., Jung, H., Cho, B. H., Lee, C., Kwak, B. K., Cha, J., Kim, D., An, J., Kim, H., and Kim, K. "SiteQ: Engineering High Performance QA System Using Lexico-Semantic Pattern Matching and Shallow NLP". In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland, November 13-16, 2001.

[13] Manber, U. and Wu, S. "Glimpse: A tool to search through entire file systems". In *Proceedings of the Winter USENIX Conference*, January 1994.

[14] Robertson, S. and Walker, S. "Okapi/Keenbow at TREC-8". In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pp. 151–162, Gaithersburg, Maryland, November 17-19, 1999.

[15] Witten, I., Moffat, A., and Bell, T. "Managing Gigabytes: Compressing and Indexing Documents and Images", Morgan Kaufmann Publishing, San Francisco, May 1999.