A Question Answering System Developed as a Project in a Natural Language Processing Course*

W. Wang, J. Auer, R. Parasuraman, I. Zubarev, D. Brandyberry, and M. P. Harper
Purdue University
West Lafayette, IN 47907
{wang28,jauer,pram,dbrandyb,harper}@ecn.purdue.edu and zubarevi@cs.purdue.edu

Abstract

This paper describes the Question Answering System constructed during a one semester graduate-level course on Natural Language Processing (NLP). We hypothesized that by using a combination of syntactic and semantic features and machine learning techniques, we could improve the accuracy of question answering on the test set of the Remedia corpus over the reported levels. The approach, although novel, was not entirely successful in the time frame of the course.

1 Introduction

This paper describes a preliminary reading comprehension system constructed as a semester-long project for a natural language processing course. This was the first exposure to this material for all but one student, and so much of the semester was spent learning about and constructing the tools that would be needed to attack this comprehensive problem. The course was structured around the project of building a question answering system following the HumSent evaluation as used by the Deep Read system (Hirschman et al., 1999). The Deep Read reading comprehension prototype system (Hirschman et al., 1999) achieves a level of 36% of the answers correct using a bag-of-words approach together with limited linguistic processing. Since the average number of sentences per passage is 19.41, this performance is much better than chance (i.e., 5%). We hypothesized that by using a combination of syntactic and semantic features and machine learning techniques, we could improve the accuracy of question answering on the test set of the Remedia corpus over these reported levels.

2 System Description

The overall architecture of our system is depicted in Figure 1. The story sentences and its five questions (who, what, where, when, and why) are first preprocessed and tagged by the Brill part-of-speech

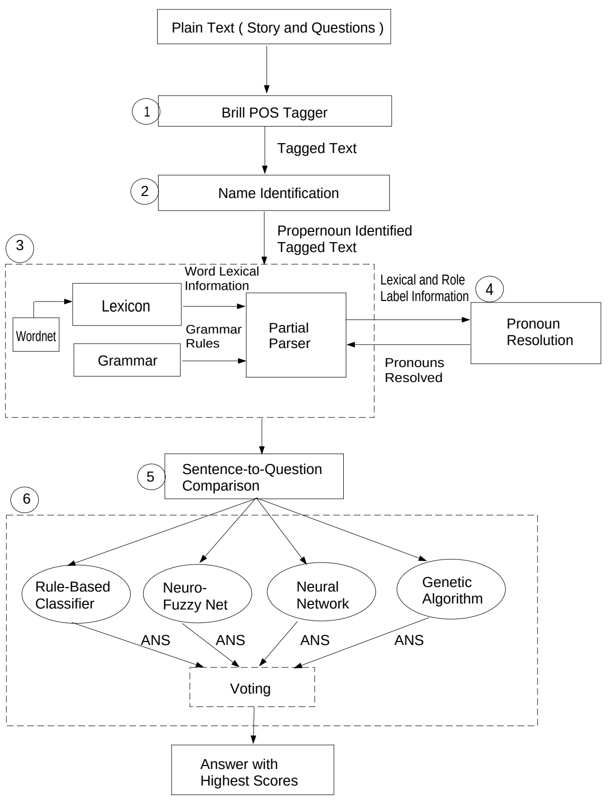


Figure 1: The architecture for our question answering system.

(POS) tagger distributed with the Deep Read system. This tagged text is then passed to the Name Identification Module, which updates the tags of named entities with semantic information and gender when appropriate. The Partial Parser Module then takes this updated text and breaks it into phrases while attempting to lexically disambiguate the text. The Pronoun Resolution Module is consulted by the parser in order to resolve pronouns before passing partially parsed sentences and questions to the Sentence-to-Question Comparison Module. The Comparison Module determines how strongly the phrases of a sentence are related to those of a question, and this information is passed to several

* We would like to thank the Deep Read group for giving us access to their test bed.

modules which attempt to learn which features of the comparison are the most important for identifying whether a sentence is a strong answer candidate. We intended to set up a voting scheme among various modules; however, this part of the work has not been completed (as indicated by the dashed lines).

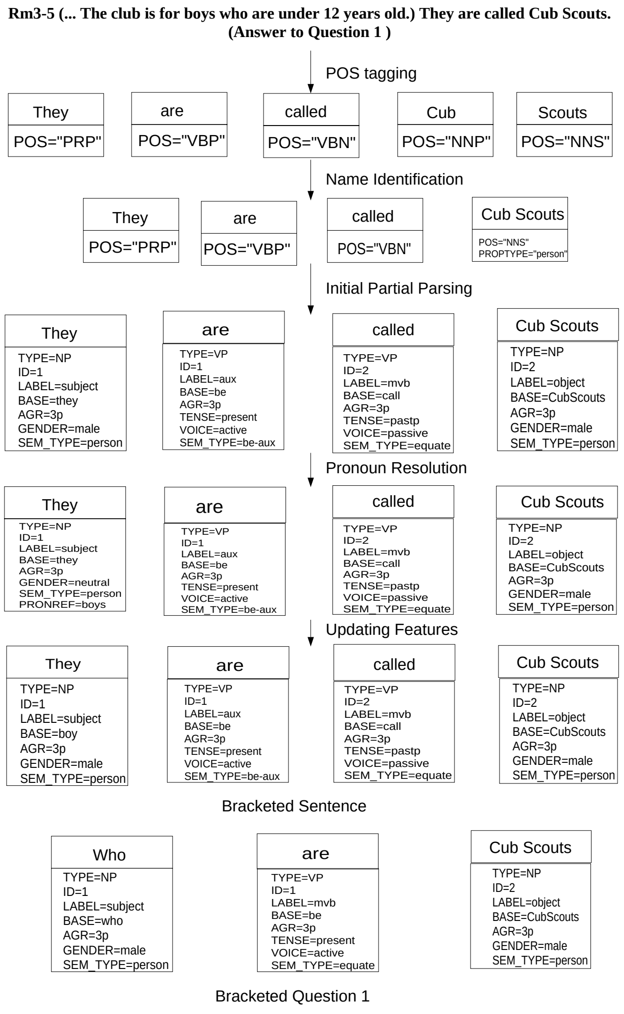
Our system, like Deep Read, uses as the development set 28 stories from grade 2 and 27 from grade 5, each with five short answer questions (who, what, when, where, and why), and 60 stories with questions from grades 3 and 4 for testing [1]. We will refer to the development and testing data as the Remedia corpus. The following example shows the information added to a plain text sentence as it progresses through each module of the system we have created. Each module is described in more detail in the following sections.

2.1 Name Identification Module

The Name Identification Module expects as input a file that has been tagged by the Brill tagger distributed with the Deep Read system. The most important named entities in the Remedia corpus are the names of people and the names of places. To distinguish between these two types, we created dictionaries for names of people and names of places. The first and last name dictionaries were derived from the files at `http://www.census.gov/genealogy/names/`. First names had an associated gender feature; names that were either male or female included gender frequency. Place names were extracted from atlases and other references, and included names of countries, major cities and capital cities, major attractions and parks, continents, etc. WordNet was also consulted because of its coverage of place names. There are 5,165 first name entries, 88,798 last name entries, and 1,086 place name entries in the dictionaries used by this module.

The module looks up possible names to decide whether a word is a person's name or a location. If it cannot find the word in the dictionaries, it then looks at the POS tags provided in the input file to determine whether or not it is a propernoun. Heuristics (e.g., looking at titles like *Mr.* or word endings like *ville*) are then applied to decide the semantic type of the propernoun, and if the type cannot be determined, the module returns both person and location as its type. The accuracy of the Name Identification Module on the testing set was 79.6%. The accuracy adjusted to take into account incorrect tagging was 83.6%.

[1] There were differences between the Deep Read electronic version of the passages and the Remedia published passages. We used the electronic passages.



Rm3-5 (... The club is for boys who are under 12 years old.) They are called Cub Scouts. (Answer to Question 1)
POS tagging
They | POS="PRP"
are | POS="VBP"
called | POS="VBN"
Cub | POS="NNP"
Scouts | POS="NNS"
Name Identification
They | POS="PRP"
are | POS="VBP"
called | POS="VBN"
Cub Scouts | POS="NNS" PROPTYPE="person"
Initial Partial Parsing
They | TYPE=NP ID=1 LABEL=subject BASE=they AGR=3p GENDER=male SEM_TYPE=person
are | TYPE=VP ID=1 LABEL=aux BASE=be AGR=3p TENSE=present VOICE=active SEM_TYPE=be-aux
called | TYPE=VP ID=2 LABEL=mvb BASE=call AGR=3p TENSE=pastp VOICE=passive SEM_TYPE=equate
Cub Scouts | TYPE=NP ID=2 LABEL=object BASE=CubScouts AGR=3p GENDER=male SEM_TYPE=person
Pronoun Resolution
They | TYPE=NP ID=1 LABEL=subject BASE=they AGR=3p GENDER=neutral SEM_TYPE=person PRONREF=boys
are | TYPE=VP ID=1 LABEL=aux BASE=be AGR=3p TENSE=present VOICE=active SEM_TYPE=be-aux
called | TYPE=VP ID=2 LABEL=mvb BASE=call AGR=3p TENSE=pastp VOICE=passive SEM_TYPE=equate
Cub Scouts | TYPE=NP ID=2 LABEL=object BASE=CubScouts AGR=3p GENDER=male SEM_TYPE=person
Updating Features
They | TYPE=NP ID=1 LABEL=subject BASE=boy AGR=3p GENDER=male SEM_TYPE=person
are | TYPE=VP ID=1 LABEL=aux BASE=be AGR=3p TENSE=present VOICE=active SEM_TYPE=be-aux
called | TYPE=VP ID=2 LABEL=mvb BASE=call AGR=3p TENSE=pastp VOICE=passive SEM_TYPE=equate
Cub Scouts | TYPE=NP ID=2 LABEL=object BASE=CubScouts AGR=3p GENDER=male SEM_TYPE=person
Bracketed Sentence
Who | TYPE=NP ID=1 LABEL=subject BASE=who AGR=3p GENDER=male SEM_TYPE=person
are | TYPE=VP ID=1 LABEL=mvb BASE=be AGR=3p TENSE=present VOICE=active SEM_TYPE=equate
Cub Scouts | TYPE=NP ID=2 LABEL=object BASE=CubScouts AGR=3p GENDER=male SEM_TYPE=person
Bracketed Question 1

Figure 2: Processing an example sentence for matching with a question in our system.

2.2 Partial Parser Module

The Partial Parser Module follows sequentially after the Name Identification Module. The input is the set of story sentences and questions, such that the words in each are tagged with POS tags and the names are marked with type and gender information. Initially pronouns have not been resolved; the partial parser provides segmented text with rich lexical information and role labels directly to the Pronoun Resolution Module. After pronoun resolution, the segmented text with resolved pronouns is returned to the partial parser for the parser to update the feature values corresponding to the pronouns. Finally, the partial parser provides bracketed text to the Comparison Module, which extracts features that will be used to construct modules for answering questions.

The Partial Parser Module utilizes information in a lexicon and a grammar to provide the partial parses. The lexicon and the parser will be detailed in the next two subsections.

2.2.1 The Lexicon

There were two methods we used to construct the lexicon: **open lexicon**, which includes all words from the development set along with all determiners, pronouns, prepositions, particles, and conjunctions (these words are essential to achieving good sentence segmentation), and **closed lexicon**, which includes all of the development and testing words[2]. We constructed the closed lexicon with the benefit of the development corpus only (i.e., we did not consult the test materials to design the entries). To improve coverage in the case of the open lexicon, we constructed a module for obtaining features for words that do not appear in the development set (unknown words) that interfaces with WordNet to determine a word's base/stem, semantic type, and synonyms. When an unknown word has multiple senses, we have opted to choose the first sense because WordNet orders senses by frequency of use. Ignoring numbers, there are 1,999 unique words in the development set of the Remedia corpus, and 2,067 in the testing data, of which 1,008 do not appear in the development set. Overall, there are 3,007 unique words across both training and testing.

One of our hypotheses was that by creating a lexicon with a rich set of features, we would improve the accuracy of question answering. The entries in the lexicon were constructed using the conventions adopted for the Parsec parser (Harper and Helzerman, 1995; Harper et al., 1995; Harper et al., 2000). Each word entry contains information about its root word (if there is one), its lexical category (or categories) along with a corresponding set of allowable features and their corresponding values. Lexical categories include noun, verb, pronoun, propernoun, adjective, adverb, preposition, particle, conjunction, determiner, cardinal, ordinal, predeterminer, noun modifier, and month. Feature types used in the lexicon include **`subcat`**, **`gender`**, **`agr`**, **`case`**, **`vtype`** (e.g., progressive), **`mood`**, **`gap`**, **`inverted`**, **`voice`**, **`behavior`** (e.g., mass), **`type`** (e.g., interrogative, relative), **`semtype`**, and **`conjtype`** (e.g., noun-type, verb-type, etc.). We hypothesized that **`semtype`** should play a significant role in improving question answering performance, but the choice of semantic granularity is a difficult problem. We chose to keep the number of semantic values relatively small. By using the lexicographers' files in WordNet to group the semantic values, we selected 25 possible semantic values for the nouns and 15 for the verbs. A script was created to semi-automate the construction of the lexicon from information extracted from previously existing dictionaries and from WordNet.

[2]Initially, we created the closed lexicon because this list of words was in the Deep Read materials. Once we spotted that the list contained words not in the development material, we kept it as an alternative to see how important full lexical knowledge would be for answering questions.

2.2.2 The Partial Parser

The parser segments each sentence into either a noun phrase (NP), a verb phrase (VP), or a prepositional phrase (PP), each with various feature sets. NPs have the feature types: **`Base`** (the root word of the head word of the NP), **`AGR`** (number/person information), **`SemType`** (the **`semtype`** of the root form in the lexicon, e.g., person, object, event, artifact, organization), **`Label`** (the role type of the word in the sentence, e.g., subject), and **`Gender`**. Verb phrases (VPs) have the feature types: **`Base`**, **`AGR`**, **`SemType`** (the **`semtype`** of the root form in the lexicon, e.g., contact, act, possession), **`Tense`** (e.g., present, past), and **`Voice`**. Prepositional phrases (PPs) have the feature types: **`Prep`** (the root form of the preposition word), **`SemType`** (the **`semtype`** of the root form in the lexicon, e.g., at-loc, at-time), **`Need`** (the object of the preposition), and **`NeedSemType`** (the **`semtype`** of the object of the preposition). Feature values are assigned using the lexicon, Pronoun Resolution Module, and grammar rules.

We implemented a bottom-up partial parser to segment each sentence into syntactic subparts. The grammar used in the bottom-up parser is shown below:

1. NP $\rightarrow$ DET ADJ+ NOUN+
2. NP $\rightarrow$ DET NOUN
3. NP $\rightarrow$ ADJ PROPERNOUN+
4. VP $\rightarrow$ (AUX-VERB) MAIN-VERB
5. PP $\rightarrow$ ADV
6. PP $\rightarrow$ ADJ (PRED)
7. PP $\rightarrow$ PREP NP

At the outset, the parser checks whether there are any punctuation marks in the sentence, with commas and periods being the most helpful. A comma is used in two ways in the Remedia corpus: it acts as a signal for the conjunction of a group of nouns or propernouns, or it acts as punctuation signalling an auxiliary phrase (usually a PP) or sentence. In the NP conjunction case, the parser groups the conjoined nouns or propernouns together as a plural NP. In the second case, the sentence is partially parsed. The partial parser operates in a bottom-up fashion taking as input a POS-tagged and name-identified sentence and matching it to the right-hand side of the grammar rules. Starting from the beginning of the sentence or auxiliary phrase (or sentence), the parser looks for the POS tags of the words, transforming the POS tags into corresponding lexical categories and tries to match the RHS of the rules. Phrases are maintained on an agenda until they are finalized.

NPs often require merging since some consecutive NPs form a single multi-word token (i.e., multi-word names and conjunctions). An NP that results from merging two tokens into a single multi-word token has its `Base` as the rootword of the combined token, and `AGR` and `SemType` features are updated according to the information retrieved from the lexicon based on the multi-word token. In the case of an NP conjunction, the `Base` is the union of the `Base` of each NP, `AGR` is set to $3p$, and `SemType` is assigned as that of the head word of the merged NP. The rule for finding the head word of an NP is: find the **FIRST** consecutive noun (propernoun) group in the NP, then the **LAST** noun (propernoun) in this group is defined as the head word of the NP.

The partial parser performs word-sense disambiguation as it parses. Words such as *Washington* have multiple `semtype` values in the lexicon for one lexical category. The following are rules for word-sense disambiguation used by the parser:

- **NP** plus **VP** rules for word-sense disambiguation:
  If there are verbs such as *name*, *call*, or *be*, which have the `semtype` of equate, then the **NP**s that precede and follow the **VP** have the same `semtype`.
  If a noun is the object of a verb, then the `subcat` feature value of the verb can be used to disambiguate its word sense (e.g., *take* generally has the `subcat` of obj+time).
- **PP** rules for word-sense disambiguation:
  For some nouns (propernouns) which are the object of a preposition, the intersection of the `semtype` value sets of the preposition word and its object determines their `semtype`.
- **NP**s in the date line of each passage are all either dates or places with the typical order being place then time. For example, in *(WASHINGTON, June, 1989)*, *Washington* is assigned `semtype` of location rather than person.

To process unknown words (the 1,008 words in the testing set that don't appear in the development set) in the case of the open lexicon, WordNet is used to assign the `semtype` feature for nouns and verbs, the `AGR` feature for verbs can be obtained in part from the POS tag, and `AGR` for unknown noun words can be determined when they are used as the subject of a sentence. For the closed lexicon, the only unknown words are numbers. If a number is a four-digit number starting with 16 to 19 or is followed by *A.D* or *B.C.* then generally it is a year, so its `semtype` is defined as time. Other numbers tend to be modifiers or predicates and have the `semtype` of num.

2.3 Pronoun Resolution Module

A pronoun resolution module was developed using the rules given in Allen's text (Allen, 1995) along with other rules described in the work of Hobbs (Hobbs, 1979). The module takes as input the feature-augmented and segmented text provided by the partial parser. Hence, the words are marked with lexical (including gender) and semantic feature information, and the phrase structure is also available. After the input file is provided by the Partial Parser Module, the Pronoun Resolution Module searches for the pronouns by looking through the NPs identified by the partial parser. Candidate antecedents are identified and a comparison of the features is made between the pronoun and the possible antecedent. The phrase that passes the most rule filters is chosen as the antecedent. First and second person pronouns are handled by using default values (i.e., writer and reader). If the system fails to arrive at an antecedent, the pronoun is marked as non-referential, which is often the case for pronouns like *it* or *they*. Some of the most useful rules are listed below:

- Reflexives must refer to an antecedent in the same sentence. For simplicity, we chose the closest noun preceding the pronoun in the sentence with matching `Gender`, `AGR`, and `SemType`.
- Two NPs that co-refer must agree in `AGR`, `Gender`, and `SemType` (e.g., person, location). Since, in many cases the gender cannot be determined, this information was used only when available.
- A subject was preferred over the object when the pronoun occurred as the subject in a sentence.
- When *it* occurs in the beginning of a paragraph, it is considered non-referential.
- We prefer a global entity (the first named entity in a paragraph) when there is a feature match. In the absence of such, we prefer the closest propernoun preceding the pronoun with a feature match. If that fails, we prefer the closest preceding noun or pronoun with a feature match.

The accuracy of our pronoun resolution module on the training corpus was 79.5% for grade 2 and 79.4% for grade 5. On testing, it was 81.33% for grade 3 and 80.39% for grade 4. The overall accuracy of this module on both the testing and training corpus was 80.17%. This was an improvement over the baseline Deep Read coreference module which achieved a 51.61% accuracy on training and a 50.91% accuracy on testing, giving an overall accuracy of 51.26%. This accuracy was determined based on Professor Harper's manual pronoun resolution of both the training and testing set (the perfect coreference information was not included in the distribution of the Deep Read system).

2.4 Sentence-to-Question Comparison Module

The Sentence-to-Question Comparison Module takes as input a set of tagged stories, for which phrase types and features have been identified. The semantic and syntactic information is coded as shown in Figure 2 (using XML tags). A mechanism to quantify a qualitative comparison of questions and sentences has been developed. The comparison provides data about how questions compare to their answers and how questions compare to non-answers. The classification of answers and non-answers is implemented by using feature comparison vectors of phrase-to-phrase comparisons in questions and potential answer sentences.

A comparison is made using phrase-to-phrase comparisons between each sentence and each question in a passage. In particular, NP-to-NP, VP-to-VP, PP-to-PP, and NP-to-PP comparisons are made between each sentence and each of the five questions. These comparisons are stored for each sentence in the following arrays. Note that in these arrays Q varies from 1 to 5, signifying the question that the sentence matches. F varies over the features for the phrase match.

CN[Q][F]	Comparison of NP features (F = |{`Base`, `AGR`, and `SemType`}|) between question Q and the sentence.
CV[Q][F]	Comparison of VP features (F = |{`Base`, `AGR`, `SemType`, `Tense`}|) between question Q and the sentence.
CP[Q][F]	Comparison of PP features (F = |{`NeedBase`, `Prep`, `PPSemType`, `NeedSemType`}|) between question Q and the sentence.
CPN[Q][F]	Comparison of PP features in sentence to NP features in question Q. Here F=2, comparing `NeedBase` and `Base`, and `NeedSemType` and `SemType`.

Values for these comparison matrices were calculated for each sentence by comparing the features of each phrase type in the sentence to features of the indicated phrase types in each of the five questions. The individual matrix values describe the comparison of the best match between a sentence and a question for NP-to-NP (the three feature match scores for the best matching NP pair of the sentence and question Q are stored in CN[Q]), VP-to-VP (stored in CV[Q]), PP-to-PP (stored in CP[Q]), and PP-to-NP (store in CPN[Q]). Selecting the phrase comparison vector for a phrase type that best matches a sentence phrase to a question phrase was chosen as a heuristic to avoid placing more importance on a sentence only because it contains more information. Comparisons between features were calculated using the following equations. The first is used when comparing features such as `Base`, `NeedBase`, and `Prep`, where a partial match must be quantified. The second is used when comparing features such as `SemType`, `AGR`, and `Tense` where only exact matches make sense.

$$c = \begin{cases} 1 & \text{if } \mathrm{Str}_1 = \mathrm{Str}_2 \\ \frac{\text{min length}(\mathrm{Str}_1,\mathrm{Str}_2)}{\text{max length}(\mathrm{Str}_1,\mathrm{Str}_2)} & \text{length}(\mathrm{Str}_1) \neq \text{length}(\mathrm{Str}_2) \\ & \quad \wedge(\mathrm{Str}_1 \in \mathrm{Str}_2 \vee \mathrm{Str}_2 \in \mathrm{Str}_1) \\ 0 & \text{if } \mathrm{Str}_1 \neq \mathrm{Str}_2 \end{cases}$$

$$c = \begin{cases} 1 & \text{if } \mathrm{Str}_1 = \mathrm{Str}_2 \\ 0 & \text{if } \mathrm{Str}_1 \neq \mathrm{Str}_2 \end{cases}$$

The matrices for the development set were provided to the algorithms in the Answer Module for training the component answer classifiers. The matrices for the testing set were also passed to the algorithms for testing. Additionally, specific information about the feature values for each sentence was passed to the Answer Module.

2.5 Answer Modules

Several methods were developed in parallel in an attempt to learn the features that were central to identifying the sentence from a story that correctly answer a question. These methods are described in the following subsections. Due to time constraints, the evaluations of these Answer Modules were carried out with a closed lexicon and perfect pronoun resolution.

2.5.1 A Neuro-Fuzzy Network Classifier

An Adaptive Network-based Fuzzy Inference System (ANFIS) (Jang, 1993) from the Matlab Fuzzy Logic Toolbox was used as one method to resolve the story questions. A separate network was trained for each question type in an attempt to make the networks learn relationships between phrases that classify answer sentences and non-answer sentences differently. ANFIS has the ability to learn complex relationships between its input variables. It was expected that by learning the relationships in the training set, the resolution of questions could be performed on the testing set.

For ANFIS, the set of sentence-question pairs was divided into five groups according to question type. Currently the implementation of ANFIS on Matlab is restricted to 4 inputs. Hence, we needed to devise a way to aggregate the feature comparison information for each comparison vector. The comparison vectors for each phrase-to-phrase comparison were reduced to a single number for each comparison pair (i.e., NP-NP, VP-VP, PP-PP, NP-PP). This reduction was performed by multiplying the vector values by a normalized weighting constant for the feature values (e.g., NP-comparison = (`Base` weight)*(`Base` comparison value) + (`AGR` weight)*(`AGR` comparison value) + (`SemType` weight)*(`SemType` comparison value), with the weights summing to 1). In most cases that a match is found, the comparison values are 1 (exact match). So weights were chosen that allowed the ANFIS to tell something about the match characteristics (e.g., if the `AGR` weight is 0.15 and the `SemType` weight is 0.1, and the NP-comparison value was 0.25, it can be concluded that the NP that matched best between in the sentence-question pair had the same `AGR` and `SemType` features). The aggregation weights were chosen so that all combinations of exact matches on features would have unique values and the magnitude of the weights were chosen based on the belief that the higher weighted features contribute more useful information. The weights, ordered to correspond to the features in the table on the previous page are: (.55, .15, .3) for CN, (.55, .1, .22, .13) for CV, (.55, .15, .2, .1) for CP, and (.55, .45) for CPN.

ANFIS was trained using the update on the development set provided by the Sentence-to-Question Comparison Module as described above. During testing, the data, provided by the Comparison Module and updated as described above, is used as input to ANFIS. The output is a confidence value that describes the likelihood of a sentence being a answer. Every sentence is compared with every question in ANFIS, and then within question, the sentences are ranked by the likelihood that they are a question's answer.

The accuracy of the best classifier produced with ANFIS was quite poor. In the grade 3 set, we achieved an accuracy of 13.33% on who questions, 6.67% on what questions, 0% on where questions, 6.67% on when questions, and 3.33% on why questions. In the grade 4 set, we achieved an accuracy of 3.54% on who questions, 10.34% on what questions, 10.34% on where questions, 0% on when questions, and 6.9% on why questions. Although the best ranking sentence produced poor accuracy results on the testing set, with some additional knowledge the top-ranking incorrect answers may be able to be eliminated. The plots in Figure 3 display the number of times the answer sentence was assigned a particular rank by ANFIS. The rank of the correct sentence tends to be in the top 10 fairly often for most question types. This rank tendency is most noticeable for who, what and when questions, but it is also present for where questions. The rank distribution for why questions appears to be random, which is consistent with our belief that they require a deeper analysis than would be possible with simple feature comparisons.

2.5.2 A Neural Network Classifier

Like ANFIS, this module uses a neural network, but it has a different topology and uses an extended feature set. The nn (Neureka) neural network simulation system (Mat, 1998) was used to create a multi-layer (one hidden layer) back-propagation network. A single training/testing instance was generated from each story sentence. The network contains an input layer with two groups of features. The sentence/question feature vectors that compare a sentence to each of the five story questions comprise the first group. Sentence features that are independent of the questions, i.e., contains a location, contains a time/date, and contains a human, comprise the second group. The hidden layer contains a number of nodes that was experimentally varied to achieve best performance. The output layer contains five nodes, each of which has a binary output value which indicates whether or not the sentence is the answer to the corresponding question (i.e., question 1 through 5).

Several training trials were performed to determine the optimum parameters for the network. We trained using various subsets of the full input feature set since some features could be detrimental to creating a good classifier. However, in the end, the full set of features performed better than or equivalently to the various subsets. Increasing the number of hidden nodes can often improve the accuracy of the network because it can learn more complex relationships; however, this did not help much in the current domain, and so the number of hidden nodes was set to 16. For this domain, there are many more sentences that are not the answer to a question than that are. An effort was made to artificially change this distribution by replicating the answer sentences in the training set; however, no additional accuracy was gained by this experimentation. Finally, we created a neural network for each question type as in ANFIS; however, these small networks had lower accuracy than the single network approach.

The overall test set accuracy of the best neural network classifier was 14%. In the grade 3 set, we achieved an accuracy of 30% on who questions, 0% on what questions, 23.3% on when questions, 13.3% on where questions, and 3.3% on why questions. In the grade 4 set, we achieved an accuracy of 17.2% on who questions, 10.3% on what questions, 23.6% on when questions, 10.3% on where questions, and 3.4% on why questions.

2.5.3 A Rule-based Classifier based on C5.0

We attempted to learn rules for filtering out sentences that are not good candidates as answers to questions using C5.0 (Rul, 1999). First we extracted information from the sentence-to-question correspondence data ignoring the comparison values to make the input C5.0-compatible, and produced five different files (one for each question type). These files were then fed to C5.0; however, the program did not produce a useful tree. The problem may have been that most sentences in the passages are negative instances of answers to questions.

2.5.4 GAS

GAS (Jelasity and Dombi, 1998) is a steady genetic algorithm with subpopulation support. It is capable of optimizing functions with a high number of local optima. The initial parameters were set theoretically. In the current matching problem, because the number of local optima can be high due to the coarse level of sentence information (there can be several sentence candidates with very close scores), this algorithm is preferred over other common genetic algorithms. This algorithm was trained on the training set, but due to the high noise level in the

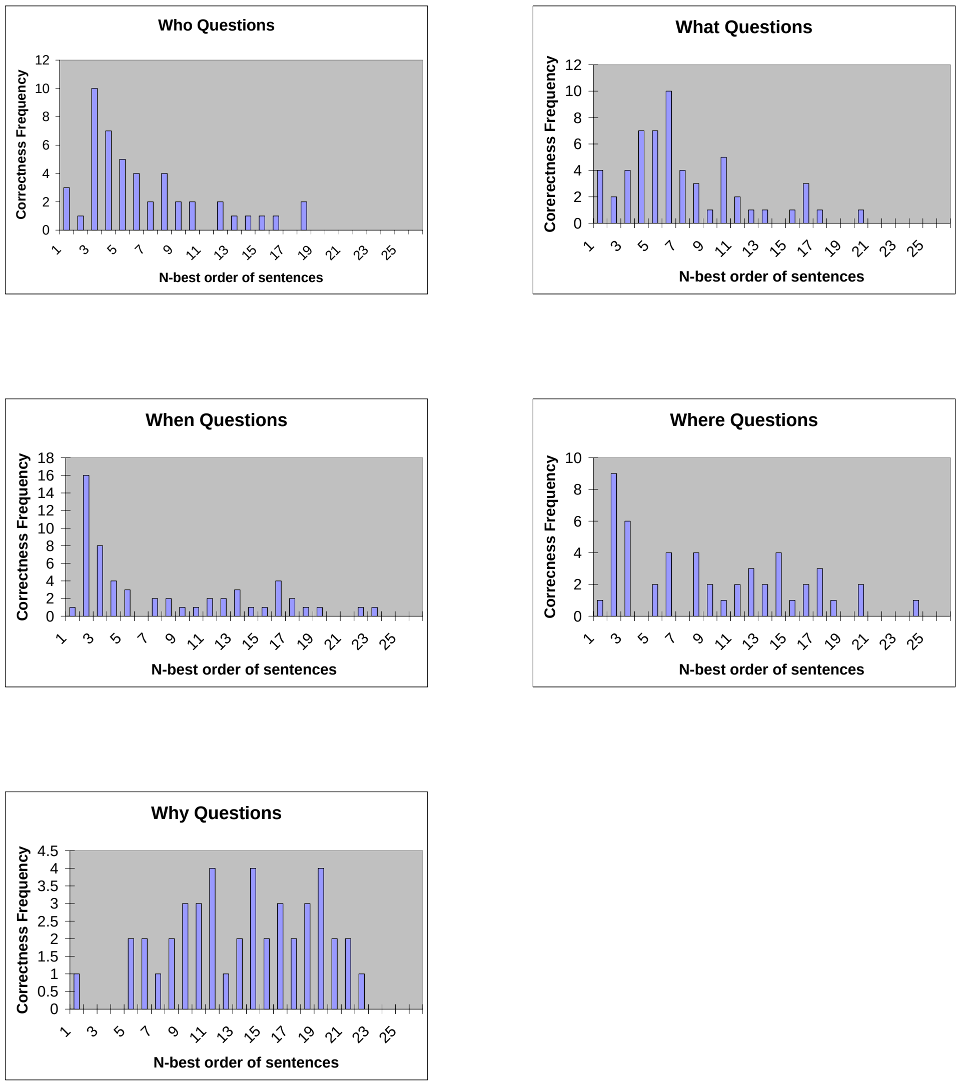


Figure 3: Correct answers ordered by ANFIS preference.

training data, the algorithm fails to produce a winning population based on the mean square minimization function.

3 A Closer Look at the Features

After observing the question answer accuracy results of the above classifiers, we concluded that the features we extracted for the classifiers are affected by noise. The fact that we take into consideration only the top matching phrase-to-phrase matches on a specific set of features may have contributed to this noisiness. To analyze the noise source of features, given that `SemType` was hypothesized to be essential for answer candidate discrimination, we examined those `SemType` values that occurred most frequently and calculated statistics on how often the values occurred in story sentences that are answers versus non-answers to the questions. We observed the following phenomena:

1. For who questions, the `SemType` value person plays an important role in identifying answer sentences, since 83.64% answers have person as its NP `SemType` value, and 21.82% have it as its PP `NeedSemType` value. However, 66.83% of the non-answer sentences also have person as its NP `SemType` and 15.85% as its PP `NeedSemType`. Phrases with person `SemType` appear in most sentences, whether they are answers or not, and this

weakens its ability to act as an effective filter.

2. For what questions, the SemType value person appears as the NP SemType of most answer and non-answer sentences. The next most dominant feature is the SemType value object, which appears in the NP for 29.41% of the answer sentences and PP NeedSemType for 15.68% of the answer sentences. Most of the other SemType values such as time contribute trivially to distinguishing answers from non-answers, as might be expected.
3. For when questions, person appears dominant among NP SemType values; however, time features appear to be second most dominant since 19.30% of the answer sentences have time as their NP SemType, and 26.32% have at-time as their PP SemType. Note that the PP NeedSemType and VP SemType appear to be less capable of guiding the selection of the correct answer.
4. For where questions, location features are important with 24.07% answer sentences having location as their NP SemType value, and 20.37% having at-loc as their PP SemType. However, the distribution of values for VP SemType and PP NeedSemType shows no interesting patterns.

The current training strategy weights the NP-NP, VP-VP, PP-PP, and NP-PP comparisons equivalently. The above observations suggest that training classifiers based on these equally weighted comparisons may have prevented the detection of a clear class boundary, resulting in poor classification performance. Since different phrase types do not appear to contribute in the same way across different question types, it may be better to generate a rule base as a prefilter to assign more weight to certain phrases or discard others before inputting the feature vector into the classifier for training.

4 Future Directions

As a next step, we will try to tame our feature set. One possibility is to use a rule-based classifier that is less impacted by the serious imbalance between negative and positive instances than C5.0 in order to learn more effective feature sets for answer candidate discrimination corresponding to different question types. We could then use the classifier as a preprocessing filter to discard those less relevant comparison vector elements before inputting them into the classifiers, instead of inputting comparison results based on the complete feature sets. This should help to reduce noise generated by irrelevant features. Also, we will perform additional data analysis on the classification results to gain further insight into the noise sources.

The classifiers we developed covered a wide range of approaches. To optimize the classification performance, we would like to implement a voting module to process the answer candidates from different classifiers. The confidence rankings of the classifiers would be determined from their corresponding answer selection accuracy in the training set, and will be used horizontally over the classifiers to provide a weighted confidence measure for each sentence, giving a final ordered list, where the head of the list is the proposed answer sentence. We propose to use a voting neural network to train the confidence weights on different classifiers based on different question types, since we also want to explore the relationship of classifier performance with question types. We believe this voting scheme will optimize the bagging of different classifiers and improve the hypothesis accuracy.

References

J. Allen. 1995. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Menlo Park, CA.

M. P. Harper and R. A. Helzerman. 1995. Managing multiple knowledge sources in constraint-based parsing spoken language. *Fundamenta Informaticae*, 23(2,3,4):303–353.

M. P. Harper, R. A. Helzerman, C. B. Zoltowski, B. L. Yeo, Y. Chan, T. Stewart, and B. L. Pellom. 1995. Implementation issues in the development of the parsec parser. *SOFTWARE - Practice and Experience*, 25:831–862.

M. P. Harper, C. M. White, W. Wang, M. T. Johnson, and R. A. Helzerman. 2000. Effectiveness of corpus-induced dependency grammars for post-processing speech. In *Proceedings of the 1st Annual Meeting of the North American Association for Computational Linguistics.*

L. Hirschman, M. Light, E. Breck, and J.D. Burger. 1999. Deep Read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332.

J. R. Hobbs. 1979. Coherence and coreference. *Cognitive Science*, 1:67–90.

J-SR Jang. 1993. ANFIS: Adaptive-Network-based Fuzzy Inference System. *IEEE Transactions on System, Man, and Cybernetics*, 23(3):665–685.

M. Jelasity and J. Dombi. 1998. GAS, a concept on modeling species in genetic algorithms. *Artificial Intelligence*, 99(1):1–19.

The MathWorks, Inc., 1998. *Neural Network Toolbox, v3.0.1.*

Rulequest Research, 1999. *Data Mining Tools See5 and C5.0.* http://www.rulequest.com/see5-info.html.