USER FRIENDLY IGUANAS

Matt Myers & Henry Longmore

ASK IGUANA

A question answering system
06 December 2001

Organization

Our system consists of a central module that calls sub-modules to complete the task of reading stories and answering questions (see Appendix A, Figure 1). The basic flow for our system is as follows: read in the story and questions, loop through each question and count words that match from the question in each sentence, call the appropriate answer module for each type of question, and output the sentence containing the answer.

Specifics of System Components

The story is "read" by the **readStory** module and parsed into separate sentences that are stored in a global array; the questions for each story are stored in a similar manner. The next thing that runs is the named entity recognizer (**NErecognizer)** which finds proper names of people and places (also some generic locations) and stores them in global data structures for use by the separate answer modules. The main module then loops through each question of the story. For each question, a call is made to **countWords** which compares the words in the question to each sentence in the story and records the number of matching words in a global array (countScores). The counter doesn't just match every word, it uses a stop-word file to remove common words including articles, pronouns, conjunctions, etc. Then to account for variations in number of nouns (plural vs. singular) and tense of verbs, the words used to compare are run through the **morphalizer** which is really just a stemmer that applies morphological rules to derive variations of words by adding or subtracting prefixes or suffixes. An irregular verb file is also used in this process to find variations of verbs that don't follow the regular rules. Once the general data is collected, a call is made to one of the answer modules, which returns the answer sentence.

Description of Answer Modules

answerWho Uses a list of named people entities populated by the Named Entity recognizer, combined with the results of question-to-sentence matching (countScores), to determine a likely answer sentence. A list of sentences to consider is initialized with sentences having a match count greater than or equal to some threshold, the match count used as the point value assigned. In the version submitted, the match count threshold is one. Next, for every item in the named people entities list, two points are added to that sentence's score. If the sentence is not already among those to be considered, it is added to that list. At this point, the maximum score is found, and the list of sentences to consider is narrowed down to include only those sentences with the maximum score. If only one sentence remains after this narrowing, it is selected as the answer sentence. Otherwise, the count scores of the remaining sentences are compared, and the one with the highest score is selected as the answer sentence.

answerWhat Uses question-to-sentence matching (countScores) for the initial guess. This is what is output for most of the questions. However, if the question indicates the answer should come from a certain semantic class, then points are added to sentences that contain instances of the given class. The semantic classes considered are locations, dates, and alternate names. If the question contains a word such as "called", or "also known as" then sentences that contain an "alternate

name" word are given a higher score. The location class is triggered when the question contains a word like "city" or "country" and a call is made to the answerWhere module (see below) and the result from that is given as the output. When the question asks what happened on some date, the module neutralizes the score for the dateline, because the dateline will most likely match, and is never the answer for this class of question. Thus, the sentences are searched for key words that indicate a generic time reference like "today" or "last night." We considered using another class of "what" questions that asked, "What is the name of...?" However, there was no increase in performance when calls were made to the answerWho module, and in some cases performance decreased, so this part was dropped.

answerWhen Uses date and time patterns to identify sentences that contain answers to a "when" type question. Scores are given depending on what type of pattern the question contains, the score from countScores is also added in. Some of the patterns match a number followed closely by a time referent (e.g. months, days, hours). Sentences that contain the preposition "on" or "in" followed by a month, or day of the week are scored. Any four consecutive digits are considered to be a year. Other time words are also matched, e.g. "midnight", "now", "o'clock." Many times a question wants to know when some event started, or ended, so this module also looks for key words indicating this kind of time reference, such as "since", "started" or, "finished." If the question contains the word "this," this is a clue that the answer is in the dateline. This works for questions like "When did this story take place?" or "When did this [event]" in the story take place? The scores from question-to-sentence matching are factored in as well, and the highest ranked sentence is given as the answer.

answerWhere Works in *exactly* the same manner as the "answerWho" module, except that it uses named place entities instead of named people entities. This is the only significant difference between these two modules. From a design standpoint, the two modules could be merged, using a parameter to indicate which type of named entities to use. However, we like the idea of using a separate file for each type of question, as it provides the most flexibility in trying to enhance the performance on any one type of question.

answerWhy Why questions are difficult to get right. The best sentence from countScores was rarely the right answer, so we looked for proximity of the correct answer to the best matching sentence, but found no regular pattern. Subsequently this module ignores countScores, except to break ties in it's own scoring method. Sentences are scored by how many weighted key words they contain. The key words are roughly broken into the following categories (given in order of point value, highest first):

Reasons: "because", "reason", "so"
Thought words: "suppose", "determine", "want"
Behavior: "must", "should", "might"
Explanations: "mean", "describe", "tell"
Amounts: "enough", "less", "each"
Other: "when", "way", "which"

The scores for this are additive, if a sentence contains more than one of these key words, the score is the sum of all the words. This method performs better than an exclusive scoring method where only the single highest value word in a sentence is scored.

Dictionaries and Other Resources

The number of external resources we used were limited. We had a list of stopwords, a rulefile, and a list of irregular verbs. The stopword list was assembled by using the lists found from the top ten results from a search on Yahoo! for the words "stop word list" (not quoted). Our list was pieced together from lists of varying size—the largest with 100 words, the smallest with only 10 words. Though the top ten sites were examined, only some had useful lists. The following five URLs were used to assist us in creating our stopword list:
http://www.versuslaw.com/Support/stopwordprint.asp,
http://www.access.gpo.gov/su_docs/locators/cgp/stopword.html,
http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words,
http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words,
http://dvl.dtic.mil/stop_list.pdf.
We removed words from the largest list, and added words from other lists. In particular, we found it helpful to remove prepositions. Our final version of the stopword list contains 69 words total, only one of which is a preposition: of. Of was removed at one point, but this decreased the performance significantly, so it was put back into the list. The stopword list is simply a file with one word per line, that word being a word to remove from the text.

The rulefile for the morphological analyzer we turned into a fancy stemmer was taken from assignment 1 of CS 5340, Fall 2001. It was shortened for this project. The list of irregular verbs was obtained from the web page found at http://www.gsu.edu/~wwwesl/egw/verbs.htm. It is a large list of base verb, past tense, and past participle. The original file format was html, so it was saved as plain text to remove the markup. Each line of the file then contained (after some editing) a base verb, followed by the past tense of that verb, followed by the past participle of that verb. There were many identical entries, such as "catch caught caught", which were shortened to "catch caught", and "bet bet bet", which were removed entirely. There were also some entries with more than three verbs listed, due to the past or past participle having more than one accepted form. The total number of lines in the final list of irregular verbs was 148; the total number of words was 371. These three files are included in Appendix A, Figures 2-4.

Contributions from User Friendly Team Members

The initial design of our system was done by both of us. Henry set out to make a stopword list and a date/time pattern list, which was absorbed into the answerWhen module. Matt began work on parsing the story files, removing stopwords from sentences, and matching words from the questions to the sentences, choosing the sentence with the most matches as the answer. These were all pieced together for the preliminary evaluation (excepting the answerWhen module, which was not yet created). After the preliminary evaluation, Henry began working on a Named Entity recognizer, while Matt worked on adding morphological analysis to the word matching. The morphological analysis devolved from the analyzer specified in assignment 1 to a fancy stemmer. Meanwhile, Henry's work on the Named Entity recognition was going well, the patterns used to find Named Entities began to include non-named entity patterns, such as a pattern to match "…in a cave." Finally, Matt began working on separate modules to answer when, what, and why questions, while

Henry worked on separate modules to answer who and where questions using the Named Entity recognizer. Lastly, Henry created the readme file and submitted the project for the final evaluation.

NLP Techniques

Our final system used few actual NLP techniques. We used some word matching with stopword removal and some stemming (which, forgive the tautology, had its roots in morphological analysis). We also had Named Entity recognition for who and where and sometimes why questions, which used some bigrams and trigrams, and in a sense, semantic classes based on prepositions. The date and time keywords for when questions also implemented semantic classes in a sense, as did the keywords for what and why questions.

Named Entity Recognition

Our named entity recognition was based mainly on recognizing patterns of one or more capitalized words in a row, a technique not extensible to texts obtained from a speech-processing engine due to a lack of capital letters. We added some context, making some patterns bigrams or trigrams, by observing which prepositions or other words tended to precede locations. Most patterns were general, while others were a bit specific. For example, the patterns "to (cap words)" and "from (cap words)" were used, as were the more specific patterns "home town" and "place in (cap words) called (cap words)". Eventually, the list of patterns was corrupted with the addition of non-capitalized patterns such as "in a (non-cap words)" and "above (non-cap words)". The pattern matching was initially done with stopwords removed; this proved to make it difficult to match answers like "He lived in a cave." Thus, the final Named Entity recognizer does not use stopword removal. Patterns for people were a bit harder to come up with than places, so there are only ten patterns for people, while there are twenty-seven patterns for places, one pattern for titles, and one pattern for unknown named entities. The people patterns take advantage of title abbreviations, such as Dr., Mr., Mrs., etc., and words like boy, girl, father, mother, sister, brother, etc. Each category of named entities had a list devoted to it, so people went into a named people entities list, places into a named places entities list, and unknown entities into an unknown entities list. All named entities at the beginning of sentences were placed into the unknown entities list. The named entity recognition did not make use of any external lists, such as planets, countries, states, cities, et cetera. The named entity patterns can be found in Appendix A, Figure 5.

Performance

Our preliminary evaluation system used simple word matching with stopword removal. The results we obtained with this system were decent, considering the simple methods we were using. On the training set, we achieved an accuracy of 20.7%, getting 29 out of 140 questions correct. On test set one, we achieved 25.9% accuracy, with 35 correct out of 135. To get a better idea of what was going on, we made some changes to the scoring program that was released after the preliminary evaluation. The result of these changes was a chart showing the correctness of our answer for each individual question, as well as totals for the story and totals for the type of question (see Appendix

B). While trying to improve our system's performance, Henry kept a log of various changes made and the resulting performance. This log is included as Appendix C.

Our final system performed well on what, when, and where questions. Where was the leader on the test set one during the preliminary evaluation with 40.74%; when lead on the training set during the preliminary evaluation with 32.14%. On the final evaluation, where lead test set one with 70.37%, and when lead on test set two and the training set with 46.67% and 64.29% respectively. Several factors may have contributed to the lower performance of our system on where questions on test set two. One may be that some patterns were quite specific, and only applied to specific stories in test set one or the training set. Another factor may be that some of the general patterns were simply not found in test set two. Further, the nature of the where question answers changed somewhat, with a higher number of answer sentences having no named entities, rendering the named entity patterns based on capitalization useless. There were more questions in test set two answerable only by the story title than in the previous test sets. Our system completely discarded the story title. Lastly, there were 35 unanswerable questions in test set two, mostly why questions, but quite a few were where questions.

After receiving the results of our system's final evaluation, we decided to play around with the system, to see what might change. If we removed the block of code in the answerWhere module that adds matched sentences to the list, our system misses six more, getting only 87 correct instead of 93, on test set two. If we (separately) remove this same block from the answerWho module, our system misses two more than the one evaluated, with 91 correct. If non-capitalized patterns are limited to a minimum of one word and a maximum of three (excluding prepositions, etc.) our system misses one more, with 92 correct. Adding a few patterns, or modify some, we get slightly worse performance on the training set (40%) and on test set one (42.14%), but 9 more correct on test set two (33.44%). Half the patterns added were actually a tweak of existing patterns, such as allowing articles to be optional, or allowing for any article rather than a specific one ("in (a|the|an)" instead of "in a"). The other half were new additions: "go from (place)", "go to (place)", "through (place)", "near (a|the|an) (place)", "around (a|the|an) (place)". All improvement (and degradation) of performance was in the where questions. Adding some additional who patterns did not affect the results.

We also wanted to see what our final system's performance would be with only word matching enabled—was all our other work worth the effort. The results our system obtained with only word matching are as follows: on the training set, 40/140 (28.57%), on test set one, 52/135 (38.52%), test set two, 85/300 (28.33%). Other than the training set, our additional modules provide only a relatively small amount of improvement.

Finally, we had a simple defect in the code that printed the answers. The answer was first checked for punctuation before printing. If none was found, a period was printed at the end of the sentence. In checking for punctuation, we neglected to check for parentheses in the event that the dateline was guesses and did not end with a "-". We also did not check for spaces, in the event that excess space was found at the end of the dateline. Also, there were inconsistencies in how the answer was tagged in the answer files. Sometimes, the "-" at the end of the dateline was not included between the answer tags, while other times it was. One of our responses was marked incorrectly as a result. Fixing the code defect and accounting for the one marked incorrect that was indeed correct gave

96/300 (32%) on test set two. If we then remove the unanswerable questions from the total possible, the result is 96/265 (36.23%). Now if we add the additional named entity patterns, the performance is increased to 104/265 (39.25%).

Successes, regrets, lessons learned

What worked, what didn't

Question-to-sentence matching gave a good mountain upon which to throw a few boulders. The NE recognizer didn't do well when there were few or no named entities. It always gave the dateline in this case. To prevent this, we used a threshold value, such that if the number of named entities was lower than the threshold, only the score from word matching was used. This sometimes eliminated the dateline from consideration when it was indeed the answer. Our approach was not very scientific, in that patterns were handcrafted, scores were assigned randomly - add 2 here, 3 there. This worked, but we believe it could have been better with a more scientific approach.

Proposed Enhancements for Ask Iguana Again

If we were asked to design another question answering system, we would keep the same basic system as a foundation. We would then use more patterns and variations of patterns in the NE recognizer. We would use Machine Learning techniques, particularly for learning patterns for the NE recognizer. We would weight various patterns in the NE recognizer, e.g. weight non-capitalized patterns lower than capitalized patterns, or specific patterns lower than non-specific. We would find a way to use "unknown" named entities, instead of leaving them to sit in a list but to no end. We also would find a way to use the actual named entities found instead of just the numbers of the sentences with named entities. We would try parsing to get parts of speech, and attempt to provide forward/backward coreference resolution for some pronouns.

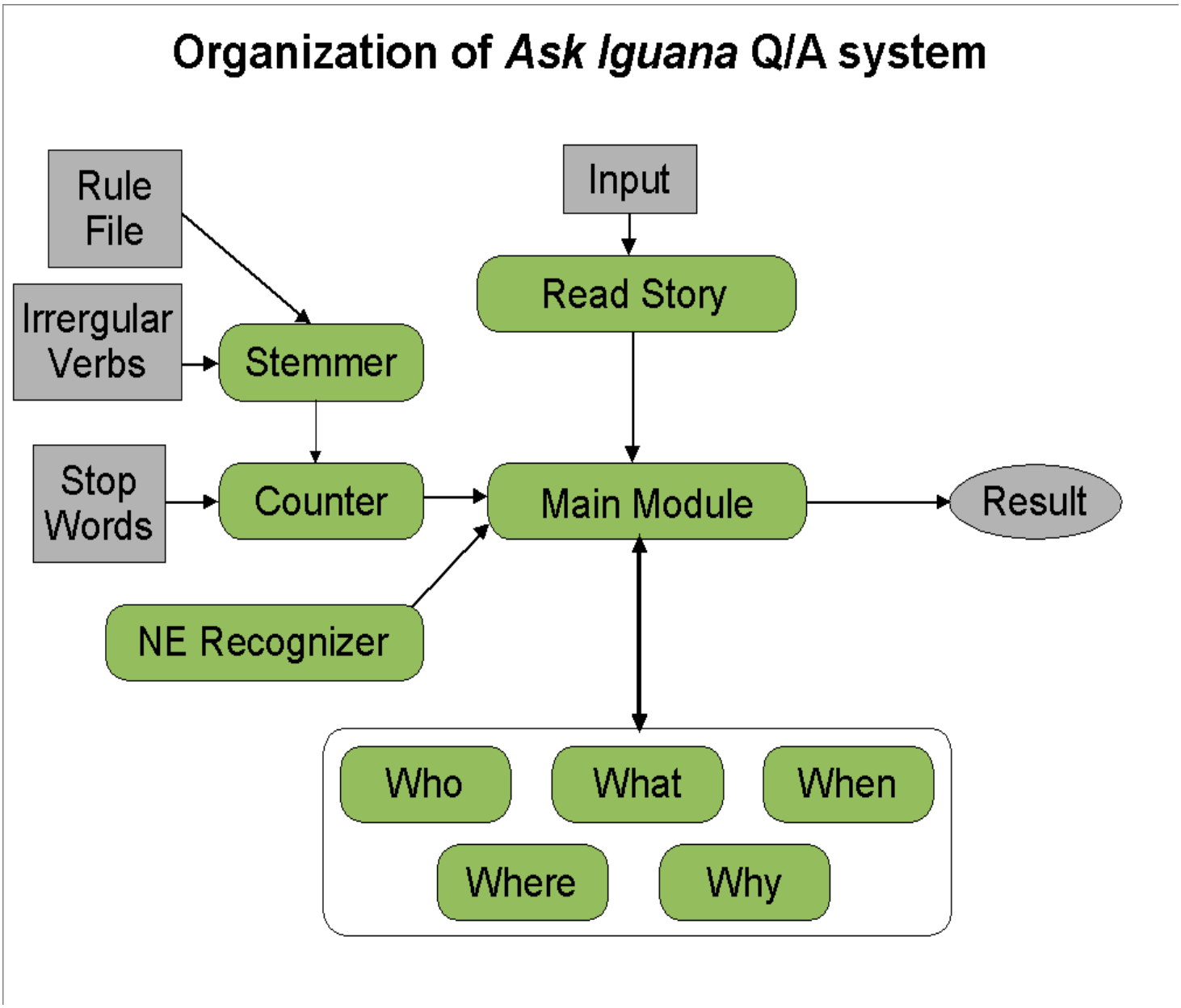
Thoughts on the experience

We both enjoyed this project. Here are some more specific thoughts from each of us.

Henry: This project hasn't really changed my view of real world NLP - I believed it was hard from stats shown in class, now I know it is. I enjoyed it immensely, but wish I had been able to devote more time to the project. My advise to all future students is to not take EE/CS 3710 the same semester as a class they have a high level of interest in…3710 will seem like the worst task master ever invented.

Matt: Getting Q/A right is harder than I thought, I figured it would be easy to get 40% on a blind test (with known question types) by adding rules and keywords. Perl rocks!

Appendix A – Figure 1



Appendix A – Figure 2

Morphology Rules

suffix ly - adjective -> adverb .
suffix ily y adjective -> adverb .
suffix ed - verb -> verb TENSE past .
suffix ed e verb -> verb TENSE past .
suffix nned n verb -> verb TENSE past .
suffix ied y verb -> verb TENSE past .
prefix re - verb -> verb .
prefix pre - verb -> verb .
suffix ness - adjective -> noun .
suffix iness y adjective -> noun .
suffix s - noun -> noun NUMBER plural .
suffix es - noun -> noun NUMBER plural .
suffix er - verb -> noun .
suffix er e verb -> noun .
suffix ier y adjective -> adjective .
suffix est - adjective -> adjective .
suffix iest y adjective -> adjective .
suffix y - noun -> adjective .
suffix y e noun -> adjective .
suffix ing - verb -> verb TENSE pres_participle .
suffix ing e verb -> verb TENSE pres_participle .
suffix nning n verb -> verb TENSE pres_participle .
suffix less - noun -> adjective .
suffix ful - noun -> adjective .
suffix ic - noun -> adjective .
suffix tion e verb -> noun .
suffix pping p verb -> verb TENSE pres_participle .
prefix en - adjective -> adjective .
suffix - ly adjective -> adverb .
suffix y ily adjective -> adverb .
suffix - ed verb -> verb TENSE past .
suffix e ed verb -> verb TENSE past .
suffix n nned verb -> verb TENSE past .
suffix y ied verb -> verb TENSE past .
prefix - re verb -> verb .
prefix - pre verb -> verb .
suffix - ness adjective -> noun .
suffix y iness adjective -> noun .
suffix - s noun -> noun NUMBER plural .
suffix - es noun -> noun NUMBER plural .
suffix - er verb -> noun .
suffix e er verb -> noun .
suffix y ier adjective -> adjective .
suffix - est adjective -> adjective .
suffix y iest adjective -> adjective .
suffix - y noun -> adjective .
suffix e y noun -> adjective .
suffix - ing verb -> verb TENSE pres_participle .
suffix e ing verb -> verb TENSE pres_participle .
suffix n nning verb -> verb TENSE pres_participle .
suffix - less noun -> adjective .
suffix - ful noun -> adjective .
suffix - ic noun -> adjective .
suffix e tion verb -> noun .
suffix p pping verb -> verb TENSE pres_participle .
prefix - en adjective -> adjective .

Appendix A – Figure 3

Stopwords (not as formatted in the file)

a also am an and any as
be but
can can't
he him himself his her hers herself
I if it its itself
me Mr. Mrs. Ms. My myself
no nor not
of or our ours ourselves
she so
than that the their theirs them these they this those
us
very
we well what when where which who whom whose why will
yes yet you your yours yourself yourselves

Appendix A – Figure 4

Irregular Verbs (not as formatted in the file)

awake awoke awoken	be was were been	bear bore born	beat beaten
become became	begin began begun	bend bent	bind bound
bite bit bitten	bleed bled	blow blew blown	break broke broken
breed bred	bring brought	build built	burn burned burnt
buy bought	catch caught	choose chose chosen	cling clung
come came	creep crept	deal dealt	dig dug
dive dived dove	do did done	draw drew drawn	dream dreamed dreamt
drive drove driven	drink drank drunk	eat ate eaten	fall fell fallen
feed fed	feel felt	fight fought	find found
flee fled	fling flung	fly flew flown	forbid forbade forbidden
forget forgot forgotten	forgive forgave forgiven	forsake forsook forsaken	freeze froze frozen
get got gotten	give gave given	go went gone	grind ground
grow grew grown	hang hung	have had	hear heard
hide hid hidden	hold held	keep kept	kneel knelt
know knew known	lay laid	lead led	leap leaped leapt
learn learned learnt	leave left	lend lent	lie lay lain
light lit lighted	lose lost	make made	mean meant
meet met	misspell misspelled misspelt	mistake mistook mistaken	mow mowed mown
overcome overcame	overdo overdid overdone	overthrow overthrew overthrown	owe owed
pay paid	plead pleaded	prove proved proven	ride rode ridden
ring rang rung	rise rose risen	run ran	saw sawed sawn
say said	see saw seen	seek sought	sell sold
send sent	sew sewed sewn	shake shook shaken	shave shaved shaven
shear sheared shorn	shine shone	shoe shoed shod	shoot shot
show showed shown	shrink shrank shrunk	sing sang sung	sink sank sunk
sit sat	sleep slept	slay slew slain	slide slid
sling slung	sow sowed sown	speak spoke spoken	speed sped
spend spent	spill spilled spilt	spin spun	spit spat
spring sprang sprung	stand stood	steal stole stolen	stick stuck
sting stung	stink stank stunk	stride strode stridden	strike struck
string strung	strive strove striven	swear swore sworn	sweep swept
swell swelled swollen	swim swam swum	swing swung	take took taken
teach taught	tear tore torn	tell told	think thought
thrive thrived throve	throw threw thrown	tread trod trodden	understand understood
uphold upheld	wake woke woken	wear wore worn	weave wove woven
wed wedded	weep wept	wind wound	win won
withhold withheld	withstand withstood	wring wrung	write wrote written

Appendix A – Figure 5

Named Entity patterns

"place called\\s+($cap_pattern+)", # "place called Blah", it's a place!
"home called\\s+($cap_pattern+)", # "home called Blah", it's a place!
"at\\s((the)?\\s+($cap_pattern+))", # it follows at (optional the is NEW after FINAL EVAL)
"place\\s+in\\s+($cap_pattern+)called\\s+($cap_pattern+)", # place in Canada called Newfoundland
"in\\s+($cap_pattern+)", # if it follows in
"up\\s+($cap_pattern+)", # if it follows up
"to\\s+($cap_pattern+)", # if it follows to
"left\\s+($cap_pattern+)", # if it follows left (left Peru)
"northern\\s+($cap_pattern+)", # directional qualities mean
"southern\\s+($cap_pattern+)", # the NE must be a place
"eastern\\s+($cap_pattern+)", #
"western\\s+($cap_pattern+)", #
"from\\s+($cap_pattern+)", # it follows from (could misfire -- The gift was from John.)
"in\\s+((a|the|an)\\s+($notcap_pattern+))", # in a cave (the & an are NEW after FINAL EVAL)
"far\\s+as\\s+($cap_pattern+)", # as far as Turkey
"place\\s+in\\s+($cap_pattern+)", # place in Canada
"middle\\s+of\\s+((the)?\\s+($cap_pattern+))", # Pacific Ocean (optional ‘the’ NEW)
"home\\s+town", # their home town (Boston)
"city\\s+of\\s+($cap_pattern+)", # city of Boston
"(($cap_pattern+)[Ii]slands\\s+of\\s+($cap_pattern+))", # The islands of Hawaii
"(($cap_pattern+)[Ii]slands)", # in Cuba and the Phillipine Islands
"place{1,1}d?\\s+near\\s+($cap_pattern+)", # placed near Ellis Island (?)
"place{1,1}d?\\s+near\\s+($notcap_pattern+)", # placed near their brains
"((above|over)\\s+($cap_pattern+))", # He flew above Jackie (over added post FINAL EVAL)
"((above|over)\\s+($notcap_pattern+))", # He flew above the monkey (?) (same as above)
"on\\s+a\\s+($notcap_pattern+)", # on a belt
"to\\s+(($notcap_pattern+)in\\s+($notcap_pattern+)?cit{1,1}(y|ies))", # 26 to homes in cities
"bottom\\s+of\\s+($notcap_pattern+)", # 27 bottom of the creek
"go\\s+from\\s+(($notcap_pattern+)to($notcap_pattern+))", # 28 NEW after FINAL EVAL
"go\\s+to\\s+($notcap_pattern+)", # 29 NEW after FINAL EVAL
"through\\s+($notcap_pattern+)", # 30 NEW after FINAL EVAL
"near\\s+((a|the|an)\\s+($notcap_pattern))", # 31 NEW after FINAL EVAL
"around\\s+((a|the|an)\\s+($notcap_pattern)in($notcap_pattern+))", # 32 NEW after FINAL EVAL
"$cap_pattern\\s+the\\s+($cap_pattern+)", # Match "Cap the Cap"
"$cap_pattern+", # Match one or more consecutive Caps

Appendix B – Detailed Results

The following pages show the results of our preliminary evaluation system on the training test set and test set one, followed by the results of our final evaluation system on all three test sets (training set, test set one, test set two).

The results of test set two have been split on two pages.

[File]	[who]	[what]	[when]	[where]	[why]	[right]
rm2-1.txt	[_]	[x]	[x]	[_]	[_]	2
rm2-10.txt	[_]	[x]	[x]	[_]	[_]	2
rm2-11.txt	[_]	[x]	[_]	[_]	[_]	1
rm2-12.txt	[x]	[_]	[x]	[_]	[_]	2
rm2-13.txt	[x]	[x]	[_]	[_]	[_]	2
rm2-14.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-15.txt	[_]	[x]	[_]	[_]	[_]	1
rm2-16.txt	[_]	[_]	[_]	[x]	[x]	2
rm2-17.txt	[_]	[_]	[x]	[_]	[_]	1
rm2-18.txt	[_]	[_]	[x]	[_]	[_]	1
rm2-19.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-2.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-20.txt	[x]	[_]	[_]	[x]	[_]	2
rm2-21.txt	[_]	[_]	[x]	[_]	[x]	2
rm2-22.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-23.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-24.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-25.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-26.txt	[_]	[x]	[_]	[_]	[x]	2
rm2-29.txt	[x]	[_]	[_]	[x]	[_]	2
rm2-3.txt	[_]	[_]	[x]	[_]	[_]	1
rm2-30.txt	[x]	[_]	[_]	[_]	[_]	1
rm2-4.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-5.txt	[_]	[x]	[_]	[_]	[_]	1
rm2-6.txt	[_]	[_]	[x]	[x]	[_]	2
rm2-7.txt	[_]	[_]	[x]	[_]	[_]	1
rm2-8.txt	[_]	[x]	[_]	[_]	[_]	1
rm2-9.txt	[_]	[_]	[_]	[_]	[_]	0
[Totals]:	5/28	8/28	9/28	4/28	3/28	29/140
	17.86%	28.57%	32.14%	14.29%	10.71%	(20.71%)

[File]	[who]	[what]	[when]	[where]	[why]	[right]
rm5-1.txt	[_]	[_]	[x]	[_]	[_]	1
rm5-10.txt	[_]	[_]	[_]	[_]	[_]	0
rm5-11.txt	[_]	[_]	[_]	[_]	[_]	0
rm5-12.txt	[_]	[_]	[x]	[_]	[_]	1
rm5-13.txt	[_]	[_]	[x]	[_]	[_]	1
rm5-14.txt	[_]	[_]	[_]	[x]	[_]	1
rm5-15.txt	[_]	[x]	[_]	[_]	[_]	1
rm5-16.txt	[_]	[_]	[x]	[_]	[_]	1
rm5-17.txt	[_]	[_]	[_]	[x]	[x]	2
rm5-18.txt	[_]	[_]	[_]	[_]	[_]	0
rm5-19.txt	[_]	[_]	[_]	[_]	[_]	0
rm5-2.txt	[x]	[_]	[_]	[_]	[_]	1
rm5-20.txt	[_]	[x]	[_]	[_]	[_]	1
rm5-21.txt	[x]	[_]	[x]	[_]	[_]	2
rm5-22.txt	[_]	[x]	[_]	[_]	[_]	1
rm5-23.txt	[_]	[x]	[x]	[x]	[_]	3
rm5-24.txt	[_]	[_]	[_]	[_]	[_]	0
rm5-25.txt	[_]	[_]	[x]	[x]	[_]	2
rm5-26.txt	[x]	[_]	[_]	[x]	[x]	3
rm5-27.txt	[_]	[_]	[_]	[x]	[_]	1
rm5-3.txt	[_]	[_]	[_]	[x]	[_]	1
rm5-4.txt	[x]	[_]	[x]	[x]	[_]	3
rm5-5.txt	[x]	[_]	[_]	[_]	[_]	1
rm5-6.txt	[_]	[x]	[_]	[_]	[_]	1
rm5-7.txt	[x]	[_]	[x]	[x]	[_]	3
rm5-8.txt	[_]	[_]	[_]	[x]	[_]	1
rm5-9.txt	[x]	[x]	[_]	[x]	[_]	3
[Totals]:	7/27	6/27	9/27	11/27	2/27	35/135
	25.93%	22.22%	33.33%	40.74%	7.41%	(25.92%)

[File]	[who]	[what]	[when]	[where]	[why]	[right]
rm2-1.txt	[_]	[x]	[x]	[x]	[x]	4
rm2-2.txt	[_]	[_]	[x]	[x]	[_]	2
rm2-3.txt	[_]	[_]	[x]	[x]	[x]	3
rm2-4.txt	[_]	[_]	[_]	[_]	[x]	1
rm2-5.txt	[x]	[x]	[x]	[_]	[_]	3
rm2-6.txt	[x]	[_]	[x]	[x]	[_]	3
rm2-7.txt	[_]	[_]	[x]	[_]	[x]	2
rm2-8.txt	[_]	[_]	[x]	[_]	[_]	1
rm2-9.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-10.txt	[_]	[x]	[x]	[x]	[_]	3
rm2-11.txt	[_]	[x]	[_]	[_]	[x]	2
rm2-12.txt	[x]	[_]	[x]	[_]	[_]	2
rm2-13.txt	[x]	[x]	[x]	[_]	[_]	3
rm2-14.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-15.txt	[x]	[x]	[_]	[x]	[_]	3
rm2-16.txt	[x]	[x]	[x]	[x]	[_]	4
rm2-17.txt	[_]	[x]	[x]	[x]	[_]	3
rm2-18.txt	[_]	[_]	[x]	[x]	[_]	2
rm2-19.txt	[_]	[_]	[_]	[_]	[x]	1
rm2-20.txt	[_]	[x]	[_]	[x]	[x]	3
rm2-21.txt	[_]	[_]	[x]	[x]	[_]	2
rm2-22.txt	[x]	[_]	[_]	[_]	[x]	2
rm2-23.txt	[_]	[_]	[_]	[_]	[_]	0
rm2-24.txt	[_]	[x]	[x]	[x]	[_]	3
rm2-25.txt	[_]	[x]	[_]	[_]	[_]	1
rm2-26.txt	[_]	[x]	[x]	[_]	[_]	2
rm2-29.txt	[x]	[_]	[x]	[x]	[_]	3
rm2-30.txt	[_]	[x]	[x]	[_]	[_]	2
[Totals]:	8/28	13/28	18/28	13/28	8/28	60/140
	28.57%	46.43%	64.29%	46.43%	28.57%	(42.85%)

[File]	[who]	[what]	[when]	[where]	[why]	[right]
rm5-1.txt	[_]	[x]	[x]	[x]	[x]	4
rm5-10.txt	[_]	[_]	[x]	[_]	[_]	1
rm5-11.txt	[_]	[x]	[_]	[x]	[_]	2
rm5-12.txt	[_]	[_]	[x]	[_]	[_]	1
rm5-13.txt	[_]	[_]	[x]	[x]	[_]	2
rm5-14.txt	[_]	[_]	[_]	[x]	[_]	1
rm5-15.txt	[_]	[x]	[x]	[_]	[_]	2
rm5-16.txt	[_]	[x]	[x]	[x]	[x]	4
rm5-17.txt	[_]	[_]	[_]	[x]	[x]	2
rm5-18.txt	[x]	[x]	[x]	[x]	[x]	5
rm5-19.txt	[_]	[_]	[x]	[x]	[_]	2
rm5-2.txt	[x]	[_]	[_]	[_]	[_]	1
rm5-20.txt	[_]	[_]	[_]	[_]	[_]	0
rm5-21.txt	[x]	[_]	[x]	[x]	[x]	4
rm5-22.txt	[_]	[x]	[x]	[_]	[_]	2
rm5-23.txt	[_]	[_]	[x]	[x]	[_]	2
rm5-24.txt	[_]	[_]	[_]	[_]	[_]	0
rm5-25.txt	[x]	[_]	[x]	[x]	[_]	3
rm5-26.txt	[_]	[_]	[_]	[x]	[_]	1
rm5-27.txt	[x]	[_]	[_]	[x]	[_]	2
rm5-3.txt	[x]	[x]	[_]	[x]	[x]	4
rm5-4.txt	[_]	[x]	[x]	[x]	[x]	4
rm5-5.txt	[x]	[_]	[_]	[x]	[x]	3
rm5-6.txt	[_]	[x]	[x]	[_]	[_]	2
rm5-7.txt	[x]	[x]	[x]	[x]	[_]	4
rm5-8.txt	[_]	[_]	[_]	[x]	[_]	1
rm5-9.txt	[x]	[x]	[_]	[x]	[_]	3
[Totals]:	9/27	11/27	15/27	19/27	8/27	62/135
	33.33%	40.74%	55.56%	70.37%	29.63%	(45.92%)

[File]	[who]	[what]	[when]	[where]	[why]	[right]
rm3-1.txt	[x]	[x]	[x]	[x]	[_]	4
rm3-10.txt	[x]	[_]	[_]	[x]	[_]	2
rm3-11.txt	[_]	[_]	[x]	[_]	[_]	1
rm3-12.txt	[_]	[_]	[x]	[_]	[_]	1
rm3-13.txt	[_]	[_]	[_]	[_]	[_]	0
rm3-14.txt	[_]	[x]	[_]	[_]	[_]	1
rm3-15.txt	[x]	[x]	[_]	[x]	[_]	3
rm3-16.txt	[_]	[x]	[x]	[_]	[_]	2
rm3-17.txt	[x]	[x]	[_]	[_]	[_]	2
rm3-18.txt	[_]	[_]	[x]	[x]	[_]	2
rm3-19.txt	[x]	[_]	[x]	[_]	[_]	2
rm3-2.txt	[_]	[x]	[_]	[x]	[_]	2
rm3-20.txt	[x]	[_]	[x]	[x]	[_]	3
rm3-21.txt	[x]	[x]	[x]	[x]	[_]	4
rm3-22.txt	[_]	[x]	[_]	[_]	[_]	1
rm3-23.txt	[x]	[_]	[_]	[x]	[_]	2
rm3-24.txt	[_]	[_]	[x]	[x]	[_]	2
rm3-25.txt	[_]	[_]	[x]	[_]	[_]	1
rm3-26.txt	[x]	[_]	[x]	[_]	[_]	2
rm3-27.txt	[_]	[x]	[x]	[_]	[x]	3
rm3-28.txt	[_]	[_]	[_]	[_]	[_]	0
rm3-29.txt	[x]	[x]	[_]	[_]	[x]	3
rm3-3.txt	[_]	[_]	[x]	[x]	[_]	2
rm3-30.txt	[_]	[x]	[x]	[x]	[_]	3
rm3-4.txt	[x]	[x]	[_]	[_]	[_]	2
rm3-5.txt	[_]	[_]	[_]	[_]	[_]	0
rm3-6.txt	[x]	[_]	[_]	[x]	[_]	2
rm3-7.txt	[_]	[_]	[_]	[_]	[_]	0
rm3-8.txt	[_]	[x]	[x]	[x]	[_]	3
rm3-9.txt	[_]	[x]	[_]	[_]	[_]	1

[File]	[who]	[what]	[when]	[where]	[why]	[right]
rm4-1.txt	[x]	[_]	[_]	[_]	[_]	1
rm4-10.txt	[_]	[_]	[x]	[_]	[_]	1
rm4-11.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-12.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-13.txt	[_]	[x]	[x]	[_]	[x]	3
rm4-14.txt	[_]	[x]	[_]	[x]	[x]	3
rm4-15.txt	[_]	[_]	[x]	[_]	[_]	1
rm4-16.txt	[x]	[_]	[_]	[_]	[_]	1
rm4-17.txt	[_]	[x]	[x]	[x]	[_]	3
rm4-18.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-19.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-2.txt	[_]	[x]	[_]	[_]	[x]	2
rm4-20.txt	[_]	[_]	[_]	[x]	[_]	1
rm4-21.txt	[_]	[x]	[x]	[x]	[_]	3
rm4-22.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-23.txt	[_]	[_]	[x]	[_]	[_]	1
rm4-24.txt	[_]	[x]	[_]	[x]	[_]	2
rm4-25.txt	[_]	[_]	[x]	[_]	[_]	1
rm4-26.txt	[_]	[_]	[_]	[_]	[x]	1
rm4-27.txt	[_]	[_]	[x]	[_]	[_]	1
rm4-28.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-29.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-3.txt	[_]	[x]	[_]	[_]	[_]	1
rm4-30.txt	[_]	[_]	[x]	[_]	[x]	2
rm4-4.txt	[_]	[_]	[x]	[x]	[_]	2
rm4-5.txt	[_]	[_]	[x]	[_]	[_]	1
rm4-6.txt	[x]	[x]	[_]	[x]	[_]	3
rm4-7.txt	[_]	[_]	[x]	[_]	[_]	1
rm4-8.txt	[_]	[_]	[_]	[_]	[_]	0
rm4-9.txt	[x]	[_]	[x]	[_]	[_]	2

[Totals]:	16/60	22/60	28/60	20/60	7/60	93/300
	26.67%	36.67%	46.67%	33.33%	11.67%	(31%)

Appendix C

Henry's log of changes

We have ask_iguana fine-tuned to get 45.9% on test_set1. Some of the fine-tuning may be too specific to this test set, but perhaps not. We shall see.

In doing this fine tuning, I: stopped using the sentences with stop words removed (to avoid having to remove too much from the stop list yet still match specific patterns, like 'in a cave', and not others, like 'in a well-known ...' (matched in a well, making well a place)). I removed the preposition 'for' from the stoplist. Without answer_where plugged in, this gets 51 correct (improvement of 1 -- story 5-9 Q 1 gets answered, no others missed as a result). With answer_where plugged in, this gets 55 correct (improvement of 2 from when 'for' was on the stoplist). I tried removing 'of', but that worsened the performance, so I put it back. It seems 'of' is less indicative of a person or place or thing than other prepositions. I also added a resolution in the case of ties (the @temp_answer array has more than one element)--select the last element. This got two more correct (from 56 (see below) to 58), and increased the % correct from 41.48 to 42.96.

I commented out a rather specific regular expression (The islands of Hawaii), and fixed a bug in another (middle of the Pacific Ocean -- but reg. expr. was looking for 'middle of thePacific Ocean'). This only changed which wrong answer was given. I added the specific pattern back in, and edited one pattern to include [Ii] as the first letter for Island. This adds one more correct to the list, bringing the number of correct to 59, and the % to 43.704.

I have added a 'hack' that I think will be pretty general...it is my opinion/estimation that few, if any, answer sentences will be questions themselves. So if the answer we've selected is a question, we go with the default--the dateline. This increased the number correct from 55 to 56.

I added another 'hack' by adding the word pair disappoint, disappointment to the list of irregular verbs in lieu of actual morphological analysis. If we were to spend more time on this, real morphological analysis would be better...or maybe we could add stemming instead? This did not work. Perhaps stemming would? Matt would probably know better why it didn't work.

Another addition, given more time, might be a partial parser to use for coreference resolution (every time a they is encountered, search backwards for another they, or for a group). However, this would only be helpful in some cases...in many cases, it is the sentence with the coreference that contains the answer.

The who patterns aren't as effective as the where patterns, but then there aren't as many, either. They are, however, a little more general than the where patterns, so they may achieve equally good results on the unknown test set (2).

With the What & When & Why modules plugged in, the system achieves 46.6% on famine (but only about 45.9% on linux on Matt's machine). It also performs well on the training set, achieving between 40 and 42.5% depending on which segments of code are commented out ("bottom of" pattern, tie resolution for where and who q's).

This tie resolution technique is more general than picking the last sentence, but it misses 2 more questions on test set 1 (one who and one where), and gets 1 more right on the training set (one where). Because it is more general, I expect it has the best chances of performing well on test set 2 (compared to choosing the last sentence).

I wish we were using the unknown NE's, but they wouldn't be too useful right now. They need to have stopwords removed from them.

On the training set and test set 1, we do well on the when and where questions, and pretty good on the what questions, but not too well on the who and why questions. The who questions could be improved my making better use of the unknown named entities (if they had stopwords removed). The why questions are just plain hard. A lot of them have no answer, or the answer requires coreference resolution and perhaps some semantics or world knowledge.

I tried using the unknown NE's in the tie resolution of who and where q's. The results were slightly worse. The unknown NE's still have all capitalized words at the beginning of a sentence that are not on the stoplist, which is not optimal. Also, perhaps a better thing to do would be to search for unknown NE's in the place and who and title NE's, and try to "unify" them in a sense (e.g. We find that Mr. Robbins is a person, and we have a Christopher Robbins in the unknown NE's--make the latter part of the person list).