A Syntactic Candidate Ranking Method for Answering Questions with a Main Content Verb

Abolfazl Keighobadi Lamjiri, Leila Kosseim, Thiruvengadam Radhakrishnan
Department of Computer Science and Software Engineering
Concordia University, Montréal, Canada
{a_keigho,kosseim,krishnan}@cse.concordia.ca

Abstract

We present a technique for ranking the candidate answers of questions that have a main content verb. This novel ranking method uses the question head (the most important noun phrase) as an anchor for selecting the target subtree in the parse tree of the candidate sentence. The semantic similarity of the action in the selected subtree to the action asked by the question is verified using WordNet::Similarity. For verifying the syntactic similarity of the target subtree to the question's parse tree, syntactic restrictions as well as word-based measures compute the unifiability of critical syntactic participants in the trees.

Results show a precision of 48% on the TREC 2003 to 2006 non-copulative questions. This confirms our hypothesis of the applicability of a basic syntactic mapping for question answering. Finally, in order to apply the web redundancy statistics into our linguistic method, we fed Aranea answers into our linguistic QA system. We obtained a slight accuracy improvement for the selected TREC question sets compared to using Aranea alone.

Keywords

Question Answering, Syntactic Mapping, Non-copula Verbs

1 Introduction

In this paper, we present a technique for ranking the candidate answers of questions that have a main content verb. Researchers in QA have classified questions based on various features; for example,

- their semantic type: arranged hierarchically in taxonomies (ex. comparison, definition, spatial or temporal, procedural, etc.) [3].
- their structure: factoid, that ask for names, dates, locations, quantities, etc. versus complex questions, that require syntactic, semantic or contextual processing, relation detection, etc. or ask for a list of answers, or any important information about a topic [13].

We categorize questions based on their main verb type into copulative (that have a 'to be' main verb) such as *Q66.2-"Who was the on-board commander of the submarine?"*, versus non-copulative (with main content verb) questions, such as *Q149.2-"The Daily Show parodies what other type of TV program"*[1]. The initial idea behind this categorization comes from our previous work in closed-domain [10]. As opposed to factoid questions, questions posed in a closed domain are longer and usually tend to be more open-ended and ask for properties, procedures or conditions [5]. As a result, they usually contain a main content verb, with critical syntactic subject and object relations. In our closed domain corpus [10], this type accounts for 70% of the questions. We showed that syntactic analysis is quite successful for measuring the relatedness of candidate answers to these non-copulative questions. In open domain, the distribution of questions is significantly different. For example, the TREC QA [22] data set contains only about 1/3 of non-copulative questions. Table 1 shows the distribution of questions in each category for the last four TREC question sets. In this paper, we show that our syntactic ranking technique is also applicable to open domain.

Question Type	#Questions	ratio
2003 copulative	171	59.0%
2003 non-copulative	119	41.0%
2004 copulative	149	64.8%
2004 non-copulative	81	35.2%
2005 copulative	230	62.7%
2005 non-copulative	137	37.3%
2006 copulative	264	65.5%
2006 non-copulative	139	34.5%
Total copulatives	814	63.1%
Total non-copulatives	476	36.9%

Table 1: *The number of copulative versus non-copulative questions in each TREC QA question set.*

Analysis of previous TREC results shows that all participating QA systems perform slightly better on copulative questions, practically showing that non-copulatives are more difficult to answer. To our knowledge, work on categorizing questions based on their main verb type has not been investigated before.

The syntactic scoring method we present here uses the question head as an anchor to select a subtree in the parse tree of the candidate sentence, where the exact answer will be found. The semantic similarity of the action in the selected subtree to the action asked in the question is then measured using WordNet::Similarity [17]. In order to verify the syntactic

[1] Although there are other copula verbs in English, such as 'look', 'feel', 'taste', 'smell', 'sound', etc., that can be used to connect the subject to an adjective, we only make a distinction between 'to be' versus non-'to be' verbs.

similarity of this subtree to the question parse tree, syntactic restrictions as well as lexical measures compute the unifiability of critical syntactic participants in the trees. Finally, the noun phrase that is of the expected answer type in the target subtree is extracted and returned from the best candidate sentence.

Although each technique has been investigated individually in different types of texts [8, 20], to our knowledge, this novel combination of syntactic and lexical criteria is unique and new to the field. Some linguistic QA systems such as LCC [15] transform the question and the set of candidate answer sentences into predicate logic form and return the value that satisfies the constraints on the answer variable. This approach transforms the text into word tokens and tries to apply the correct meaning of each token for understanding the text. Therefore, it requires a perfect syntactic parse tree of sentences as well as disambiguated word tokens. Additionally, reasoning over these predicates requires a high amount of world knowledge. As we will see, our approach however uses syntax while not being dependent on having a perfect parse tree. Finally, it can be used to syntactically verify the answers given by other QA systems.

2 Candidate Answer Extraction

In this section, we review the processes involved in the information retrieval phase for question answering. Section 3 will present our linguistic method for the candidate ranking phase.

2.1 Question Analyzer

The question analyzer module extracts the *expected answer type* and a ranked list of *question keywords* to be fed to the Lucene IR engine[2]. *Question keywords* will be used to retrieve the documents and passages relevant to the question, based on the assumption that relevant passages contain words in common with the question. The *expected answer type* is used as a constraint for extracting the answer. For example, the answer type for a question starting with 'Where' is a *location*. We use the existing work done in the Aranea QA system [14] to extract the expected answer type. Aranea is one of the top 5 QA systems in TREC 2002.

Important words are then marked as question keywords. Two factors contribute in deciding if a word is important: its part-of-speech and its number of modifiers. The question analyzer first filters out non-content words and keeps nouns, verbs, adjectives and adverbs. Part-of-speech tagging is done through the Gate infrastructure[3], which uses the Brill POS tagger [2], with the Penn Treebank tag set. So, specifically, we keep words with the following part-of-speech tags: NN, NNP, NNS, JJ, JJS, VB, VBG, VBN, VBD, VBZ, RB, RBS. The analyzer then processes *important* parse links provided by the Minipar parser [11] for identifying the question keywords. To do so, based on the type of the parse link seen, both the head and the tail or only the tail is kept:

- For a nominal complement of a preposition such as *"for convenience"* and determiner relation such as *"the network"* (shown as *'pcomp-n'*, *'det'* respectively, in the Minipar notations), only the tail (*'convenience'* and *'network'*) is marked as a question keyword.
- For an adjunct modifier link, a lexical modifier such as *"electric guitar"*, a conjunction, a subject or object links, a noun complement and a passive verb modifier of a noun such as *"the service provided..."* (shown as *'mod'*, *'lex-mod'*, *'conj'*, *'subj'*, *'obj'*, *'nn'*, and *'vrel'* in Minipar), both head and tail words are considered.

Finally, the number of modifiers of a word is computed and stored as a feature of that keyword. This feature will be used later in scoring these keywords. This strategy is based on the hypothesis that if a word has more modifiers, it acts as a central idea in the question and is therefore more important[4]. For example, in the question *Q76.4-"What is the title of his all-time best-selling record?"*, the noun 'record' has three modifiers ('his', 'all-time' and 'best-selling'). Question keywords are ranked based on the following heuristic function:

$$Score_{kw} = (\#modifiers + 1) \times Score_{POS}(kw)$$

where, $Score_{POS}$ is assigned as the following: proper nouns are favored (given a weight value of 6), then common nouns (2), verbs (1.25), adjectives (1), auxiliary verbs (0.7), and finally adverbs and determiners (0.5). These values were determined experimentally with our development set (the first 50 non-copulative questions from the 2005 set). The rationale behind these values is to boost proper nouns in the list, since they convey a unique meaning. Verbs on the other hand are more ambiguous [12] and can have more synonyms (alternatives for conveying the same meaning), so they are slightly pushed down the list. Adverbs usually relate to verbs, and not the keywords, so they receive the lowest rank. For example, the following ranking is computed for *Q98.4-"What organization has helped to revitalize Legion membership?"*:

```
Legion membership* revit* help* organiz*
```

Finally, the content words that are not involved in any syntactic relation of our interest, are added to the keyword list, with 0 as their number of modifiers. They will later be used as low-ranked keywords that can constrain the number of documents returned by IR.

2.2 Candidate Sentence Selection

The candidate answer extraction module processes the top n documents that are returned by Lucene and are found in the PRISE top document list[5]. Sentences that contain α percent of the keywords are recorded as candidate sentences to be ranked by our linguistic unifier (see Section 3). Since our rather strict unifier filters out bad answers later on, we chose a low threshold of $\alpha = 66\%$ to increase the recall at this stage.

[2] Available at `http://lucene.apache.org/`

[3] General Architecture for Text Engineering, available at `http://www.gate.ac.uk/`

[4] It is interesting to note that the PiQASso QA system [1] ranks question keywords based on their depth in the parse tree.

[5] This list is compiled by the TREC organization, running their IR engine on question keywords and the answer phrase.

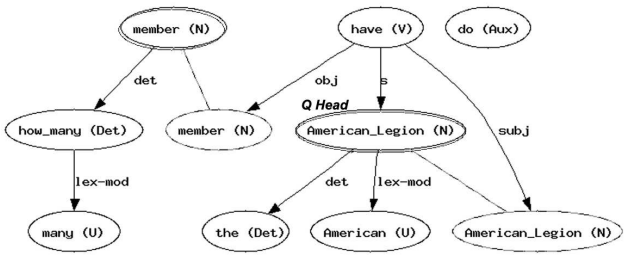

Fig. 1: *Parse structure for the question* "How many members does the American Legion have?"

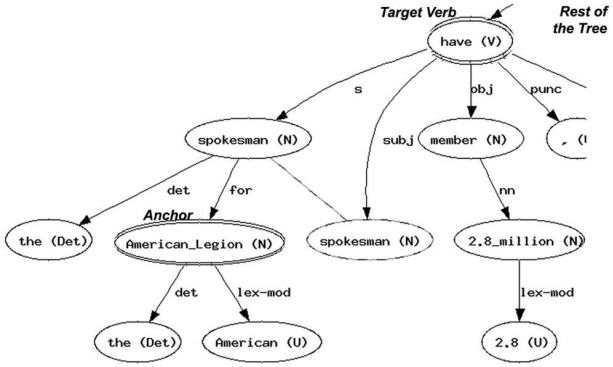

Fig. 2: *The parse structure for the sentence* "...said Phil Budahn, spokesman for the American Legion, which has 2.8 million members."

Experiments with different values of α for this boolean vector sentence selection show that varying this threshold only affects candidate ranking and answer extraction running time, but not the quality of the results. On average, the number of candidates retrieved decreases from 55 (with a standard deviation of 23) to 25 (std.dev. 16) when increasing the threshold from 35% to 75% for our development question set[6].

3 Syntactic Unification for Candidate Ranking

To rank the candidate sentences returned by the IR, we compute their syntactic similarity to the question. Pure linguistic criteria for measuring the similarity of parse trees impose very strict syntactic constraints that result in low recall. This problem has been observed by researchers in the field, such as [20]. On the other hand, statistical systems that learn and score syntactic links such as [9] and [21] are very lenient in considering the importance of primary roles (such as subject and object) over less important roles (such as a determiner or a modifier). The most interesting effort towards improving this syntactic measure is weighting the matching links according to their Inverse Document Frequency (IDF)[7]; rare link types have more information content than frequent relation types and hence, will contribute more when matching subtrees. However, this has not solved the recall problem and parse tree based techniques generally perform poorly compared to syntactically blind statistical methods.

Statistical approaches in QA inspired us to relax a strict syntactic mapping. We force critical syntactic roles to eliminate the candidates with no syntactic consistency with the question, and score the remaining links for the candidates that pass the first criterion.

3.1 Choosing the Target Subtree

Essentially, we believe that the best subtree in a candidate sentence is the one that has a similar verb to the question's main verb, and equivalent arguments. A strong verb similarity should co-occur with an essential entity similarity (question head) match in the candidate's parse tree. This suggests that a strong seed point is the root of the subtree in the candidate that contains the question's head noun phrase.

3.1.1 Finding the Question Head

To choose the question head, we rank all the noun phrases in the question and pick the one that contains the most valuable question keywords (with higher $Score_{kw}$ value, see Section 2.1). If this head phrase is found in the candidate sentence, it becomes an anchor to find the relevant verb. We then move up from this noun phrase in the candidate parse tree to reach the first parent verb. For example, in the question *Q98.5-"How many members does the American Legion have?"*, two noun phrases exist (the double lined nodes in Figure 1). The noun phrase *'American Legion'* is chosen as the question head, since it has the highest $Score_{kw}$ value. In the candidate sentence, *"...said Phil Budahn, spokesman for the American Legion, which has 2.8 million members."*, this anchor is found in the left subtree of the verb *'have'* (Figure 2). Moving up from this anchor skips the noun node *'spokesman'* and marks the verb node *'have'* as the root of the target subtree. This root will then be used as the seed point for starting the unification. In such long candidate sentences, using an anchor reduces the candidate verbs to the ones that include the question head (or a reference to it).

3.1.2 Semantic Similarity of Verbs

Since the main action specified in a non-copulative question is typically realized by a verb, our first step is to verify the semantic relatedness of the question's main verb to the candidate's target verb.

To do this, we use WordNet::Similarity [17]. This package provides six similarity measures which use information found in an *is-a*, *has-part*, *is-made-of*, and *is-an-attribute-of* relations in a hierarchy of concepts (or synsets) and quantify how much concept A is similar to concept B. Three measures are based on the information content (IC) of the least common subsumer (LCS) of concepts A and B. IC is a measure of the specificity of a concept, and the LCS of concepts A and B is the most specific concept that is an ancestor of both A and B. The *lin* and *jcn* [17] measures augment the IC of the LCS with the sum of the ICs of

[6] On an Athlon AMD 3500+ 64bit CPU with 4GB of RAM, the time needed to rank these candidates increases from 110 seconds to 200 seconds per question.

[7] Two links match if they have similar head, relation and tail values.

concepts A and B themselves. The *lin* measure scales the IC of the LCS by this sum, while *jcn* takes the difference of this sum and the IC of the LCS.

The other three similarity measures are based on path lengths between a pair of concepts: *lch* (Leacock and Chodorow), *wup* (Wu and Palmer), and *path*. Among these six measures, Leacock and Chodorow (lch) worked best for verbs in our development set. This measure basically finds the shortest path between two concepts, and scales that value by the maximum path length found in the *is-a* hierarchy in which they occur. We proceed with unification for the candidates that have a target verb with a similarity value more than 1.8 to the question's main verb.

Finally, we check whether the target verb relates the same entities as the question's main verb. A fuzzy statistical method evaluates the similarity of the two *subject* subtrees, and likewise for *object* or *modifier* subtrees, if any. We will look at this method in detail in the following section.

3.2 Unifying Two Subtrees

To unify two phrases (subject, object or modifier subtrees) marked by the linguistic method as the arguments of verbs, we apply a heuristic that uses two measures: the number of overlapping words based on a bag-of-words approach and the number of overlapping links.

These similarity scores are summed to produce the final score of a candidate sentence:

$$Similarity(Verb_q, Verb_T) + \Sigma_{i:Counterpart\ Subtree} Score(Q_i, T_i)$$

where, Q is the question, T, the target subtree, and

$$Score(Q_i, T_i) = \beta \times WordOverlap + (1 - \beta) \times LinkOverlap$$

is the unification score of two counterpart subtrees. The parameter β shows the relative importance of each feature: $\beta = \frac{1}{2}$ assigns equal importance to either feature, while $\beta = \frac{1}{3}$ (our configuration) considers the link-overlap to be twice as important as the bag-of-words feature. Note that the absolute value of the final score is not important since the scores are used only to rank the candidates.

Each subtree can be seen as a paraphrase, since its focus is an entity (noun) that possibly has some modifiers. For example, the noun phrase *'the American Legion'* in the question *"How many members does the American Legion have?"*(Figure 1) appears as *"spokesman for the American Legion"* in the answer sentence, depicted in Figure 2. Here, a score of 12.5 is returned by matching the words ('the', 'American' and 'Legion') and 2.0 for the matching links in the subject subtrees.

The reason we relax our linguistic constraints at this stage is that we are focusing on a sentence that conveys a similar event or state to the question; only a clue about similarity of its verb arguments is sufficient to conclude that its verb modifies the same entities as the question. Syntactic differences of verb arguments (subtrees) should not critically affect our judgment.

By analyzing a few unification cases, we realized that matching different types of links should have a variable contribution to the final unification score. Compare a modifier (*'mod'*) link matching in the candidate *"wireless network"* as opposed to a determiner (*'det'*) link in the candidate *"a network"* matching with the phrase *"... a wireless network ..."* in the question. The first case shows a stronger similarity since it narrows down the meaning of the noun (*'network'*). To account for this, we weight links differently: i.e., a lexical modifier link has the highest weight because it connects two proper nouns, while a determiner has the lowest score. Table 2 shows the classes of equivalent links we selected and the values we obtained experimentally for each class. These values can also be learned given a tagged set of equivalent, but syntactically different phrases, such as an appropriately selected subset of the *Equivalent sentence pairs with minor differences in content* from the Microsoft Research Paraphrase corpus[8].

Category	Minipar Relation	weight
Lexical modifier	lex-mod	1.0
Adjective/Nominal mod	mod,pnmod,pcomp-n,nn	0.5
(pre)Determiner	(pre)det	0.25
Possessives	gen	0.25

Table 2: *Weights of different syntactic links used in scoring the similarity of two phrases.*

For the previous example (Figure 2), the value of the *LinkOverlap* feature will therefore be 1.0+0.25 = 1.25 (for the lexical modifier and the determiner link).

3.2.1 Equivalent Copulative Structures

The above linguistic verb selection does not work well with sentences that have copulative verbs. Copulative questions such as *Q117.1-"What kind of plant is kudzu?"* or *Q144.5-"What is the divisions official song?"* have a particular structure: they convey a state and their argument structure is more flexible. Although these questions are syntactically similar, their answers come in different structures: *'ANS1'* in *"ANS1 is a member of the bean family which has..."* has a subjective role while *'ANS2'* in *"The divisions official song is ANS2"* comes in the predicate of a relative clause (*'pred'* subtree in Minipar). This phenomenon led us to allow toggling of subject and predicate arguments when unifying copulative structures.

3.2.2 Inter-type Syntactic Mapping

When the question or the answer sentence is copulative while the other one has a non-copulative main verb, they cannot be mapped to each other without syntactic modification or semantic reasonning. The answer to the non-copulative question *Q109.2-"How many countries does it operate in?"* about "Telefonica of Spain" is answered by the propositional attachment in the copulative sentence *"Telefonica is the largest supplier of telecommunications services in the Spanish and Portuguese speaking world with operations in 17 countries and over 62 million customers."*.

Multiple mapping cases can happen in this situation; for example, the answer to a copulative question might appear as a noun modifier or a propositional attachment in a non-copulative sentence. Manual modeling of all possible mapping cases is difficult and will

[8] Available at `http://research.microsoft.com/research/`

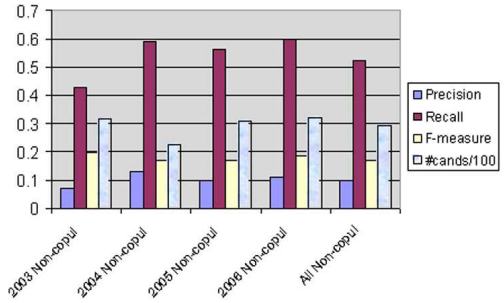

Fig. 3: *Precision and recall of the Lucene IR engine at the document level for the TREC non-copulative questions.*

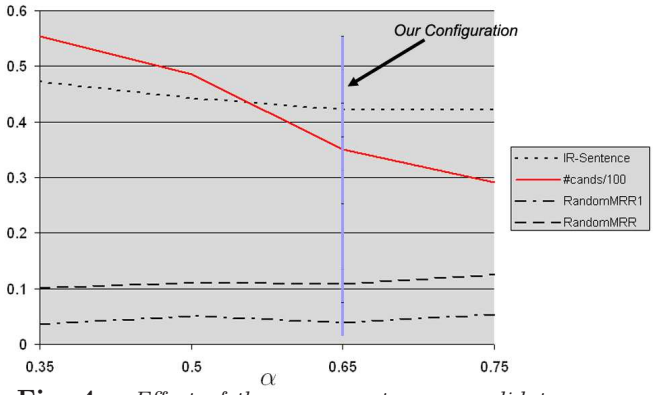

Fig. 4: *Effect of the α parameter on candidate answer extraction recall and the number of candidate sentences selected for the TREC question sets.*

not cover many cases. This should be done automatically and with a larger data set in order to significantly improve the results. For this reason, we leave this task as future work.

4 Evaluation

To evaluate our scoring approach, we first computed the accumulated error in extracting candidate answer sentences (the first phase). Since these are the sentences sent to the unifier (the ranking phase), they impose an upper bound on the final result.

4.1 Candidate Extraction Results

Figure 3 shows the precision and recall of the Lucene IR engine for the 426 non-copulative questions in the TREC 2003 to 2006 data sets (50 non-copulative questions from TREC 2005 were kept for development). Precision and recall at the document level (at level 35) are around 10% and 53% respectively[9] (except for the 2003 where most participants performed significantly more poorly). The '#cand/100' bar shows the number of candidates that are selected for each question (divided by 100). The more candidates extracted, the harder the ranking task.

Figure 4 shows the effect of the α parameter in sentence selection on precision and recall at the sentence level for our development set (see Section 2.1). We accept a slight error in precision by choosing $\alpha = 0.65$ for this parameter in favor of passing more candidates to the unifier and higher recall. On average, around 45 candidates are passed to the unifier for ranking. Figure 4 shows the effect of the parameter α on sentence selection recall and the baseline question answering accuracy (MRR). The IR-Sentence column in this Figure shows the percentage of questions that have at least one correct answer in their candidate set (candidate answer extraction recall). Sentence recall and the number of candidate sentences are the two factors that create an upper bound on the expected accuracy of our unifier. Note that sentence level recall (IR-Sentence) drops to around 44% from 53% at the document level (shown in Figure 3).

[9] Compared to the state of the art in IR for open domain QA systems (86.1% [7]), our IR method has space for improvement.

4.2 Candidate Ranking Accuracy

Candidate sentences should be ranked based on their syntactic and semantic similarity to the question. Before evaluating the performance of our ranking method, we looked at the baseline for this task. We considered the baseline to be the accuracy resulted by randomly selecting one candidate from the candidate set as the answer. The accuracy of a question answering system is reported as the Mean Reciprocal Ranking (MRR) score which is equal to the inverse of the position of the first correct answer in the list [22].

The theoretical baseline for this task is the precision of randomly selecting a candidate as the answer: $MRR = \frac{1}{x}\Sigma_{i=1}^{x}\frac{1}{i}$ where x is the number of candidate answers. On average, we have 2 correct answers in a set of $x = 44$ candidates; so we can assume one correct answer in a set of 22 candidates. This results in a theoretical baseline MRR of 16.7%. However, the experimental baseline ranking accuracy is 11.2%, because of high deviation in the size of candidate sets.

If only the top ranked candidate is considered in scoring (as applied recently in TREC QA evaluations), the probability of picking the correct answer is MRR1=$\frac{1}{22}$ = 4.5%, while experiments show MRR1=5.0%. Apparently, this baseline decreases as we relax the candidate selection threshold (lower α threshold).

Figure 5 shows the accuracy of our QA system compared to Aranea [14], a competitive open domain QA system. The column labeled 'Unifier Precision' shows the unifier's accuracy when the error in the IR's output is excluded from the final result. The results show a high performance for the candidate ranking algorithm especially for non-copulative questions (twice as high compared to copulatives). Low accuracy for the questions without a main content verb shows the important role of the main verb in our method.

The 'Aranea' column shows that it tends to work slightly better on copulative questions. Our analysis of the previous TREC submissions also shows that this is a trend in the field, indicating that non-copulative questions are typically harder to answer.

Most current open domain QA systems use redundancy from the Web and the corpus to rank their candidate answers. By combining such a list with the syntactically ranked candidate answer list re-

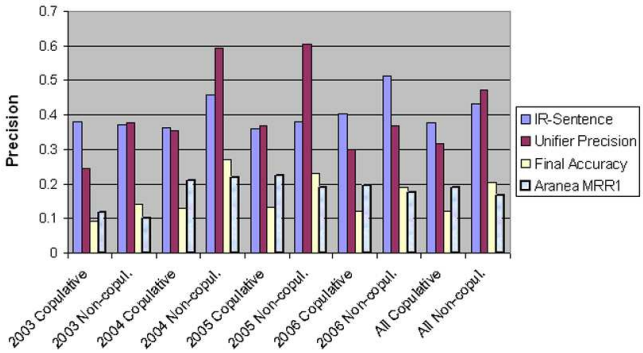

Fig. 5: *Precision (MRR1) of the unifier and the modified Aranea on TREC factoid questions.*

turned by our QA system, we have a chance to apply one's information to the other. To test this, we used the statistical Aranea QA system again. This time, the output of Aranea was used to improve our IR result (m documents from the result of $AranAnswers\ AND\ QuestionKeywords$ were added to the regular keyword document list) and provided the expected answer type.

4.3 Analysis

To better understand where the system goes wrong, we manually analyzed the errors in the 139 non-copulative questions from TREC 2006. As we mentioned earlier, lack of query expansion prevents our system to extract candidates that have different wordings from the question. Low IR recall causes more than 40% of the error by missing their candidate answers to start the unification with (see Figure 3).

The most frequent error source is when the answer to a non-copulative question is given in a copulative sentence (9 cases or 6.5%). As we mentioned earlier, many major syntactic differences can exist when mapping a copulative question on a non-copulative answer and vice versa. Manual modeling of multiple mapping cases is difficult and will not cover many cases. This should be done automatically and in large scale in order to significantly improve the results.

The *lcs* semantic similarity measure does not return a correct value for main verbs in eight the correct answer sentences. We do not specify the sense of verbs, while WordNet::Similarity has the capability of accepting the sense numbers in order to compute a more precise semantic relation between verbs.

Improving the semantic named entity tagger will help answering eight more questions by boosting their unification score. An 'Organization' tag for the sentence *"Crumb, who founded the volunteer Hospice of Clallam County..."* for example would help move this candidate for *Q120.3-"What organization did she found?"* about 'Rose Crumb' to the first rank. Three questions needed resolving the date of an event by adding the year the article is written in, to the answer. Answers found by our system for three questions were correct, but are not included in the judgement file. And finally, a bad parse tree for the correct candidate resulted in missing three more questions.

In conclusion, improving the inter-type syntactic mapping strategy, word sense disambiguation for improving the semantic similarity measure and using a more precise named entity tagger would improve our overall QA accuracy considerably.

5 Related Work

Most current TREC type question answering systems return the noun phrase in proximity of the question keywords seen in a candidate sentence that is of the expected answer type. Performing a statistical analysis in a set of web snippets afterwards indicates the most redundant answer to be most probably correct. The weakness of QA systems that incorporate parsing is that they rely on exact matching of relations, resulting in high precision while recall stays drastically low.

With one of the best performing QA systems in the TREC 2004 track, the university of Singapore QA [20] uses the Jaccard coefficient to test pairwise similarity of frames marked by the ASSERT predicate argument recognizer. This coefficient ignores stop-words and uses the bag-of-words feature for scoring. In addition to bag-of-words however, we keep stop words and find the common syntactic links (ex. *"his red car"* versus *"the car at his red door"*; we gave this syntactic feature twice as much contribution in the subtree (ARG) scoring by $(1 - \beta) = 2/3$).

Katz and Lin [8] have a ternary subject-verb-object scheme and use predicate logic; the constraint satisfaction to find an answer that satisfies the syntactic/semantic constraints is binary while we use a fuzzy scoring schema. Applicability of their comprehensive state-of-the-art method is shown successfully on five questions. Breaking the text into small grains in predicate-logic form is less feasible to apply in large scale and open-domain:

Salvo et al., in [4] introduce a hierarchical knowledge representation for Meaning Entailment: a sentence is entailed by a paragraph if its context graph can be unified with that of the paragraph. A cost function determines the goodness of a unification. Unified nodes must be at the same level in the hierarchy, and the cost of unifying nodes at higher levels dominates those of the lower levels. Nodes in both hierarchies are checked for subsumption in a top-down manner: The hierarchy level H_0 consists of verbs that unify if they are synonyms based on WordNet and their constituent phrases at H_1 level unify. Hierarchy set H_2 corresponds to word-level nodes. As it can be seen, syntax is used only at the topmost level H_0. As we will see later, subject and object relations are considered to be critical in our matching algorithm.

On the other hand, PiQASso [1] and AnswerFinder [16] compute the match between a question and a candidate answer using a metric which computes the overlap in their dependency relations. A similar work by Nyberg [6] introduces a light-weight fuzzy unification as an extension to their earlier work, JAVELIN; here, counterpart syntactic links and their head and tail tokens contribute to the final match score. Unlike PiQASso, syntactic links are weighted so that a matching 'subject' link has higher contribution than a 'determiner' link. For this linguistic work however, no evaluation result is provided.

In a popular statistical method to unify parse trees,

Raina et al. [19] learn weights for matching subtree at the source and destination nodes: matching of the modifier of two verb nodes may contribute less to the unification score than matching of their subjects. The algorithm given by Punyakanok basically selects the matching that results in the minimum *Edit Distance* in [18]. They experimentally determine weights and penalty values for *delete, insert* and *change* operations in a labeled tree-matching algorithm.

6 Conclusion and Future Work

In this paper we presented a method that imposes simple linguistic constraints to select only the candidates that refer to the same event that the question asks for. At the same time, candidate sentences are syntactically chunked. A heuristic measure then computes the similarity of each chunk in a candidate to its counterpart in the question. The similarity of the verb and its entities show high resemblance of that candidate to the question. The final answer is extracted and returned from the top ranked candidate.

We evaluated this algorithm on TREC 2003 to 2006 QA question sets and showed that our unification based scoring method achieves an accuracy of 47% for non-copulative factoid questions, while around one third of TREC questions are non-copulative.

Although we have a relatively low accuracy at the sentence extraction level, optimizing this phase will make the ranking task more difficult by extracting more candidates. Based on our closed domain experiments, however, we believe that using linguistic features brings robustness towards having a larger candidate set [10].

Special attention should be given to parsing the question; for example, converting the question to affirmative form or using more than one parser to detect incorrectly parsed questions. We are currently studying the combination of answer redundancy prevalent in open domain QA with our linguistic method to obtain higher performance in TREC questions. We categorized the questions based on the main verb type and showed that our QA method is specialized to answer questions that have a main content verb. Defining or learning more linguistic features for the question in order to categorize and feed questions based on the specialty of different QA systems might give an ultimate solution to tackle the question answering problem.

Acknowledgement

This research was financially supported by a grant from NSERC and Bell University Laboratories.

References

[1] G. Attardi, A. Cisternino, F. Formica, M. Simi, and A. Tommasi. PiQASso: Pisa Question Answering System. In *Proc. of the TREC-12 Conference*, pages 599–607, Gaithersburg, MD, 2001.

[2] E. Brill. A Simple Rule-based Part of Speech Tagger. In *Proc. of the Third ANLC*, pages 152 – 155, Italy, 1992.

[3] J. Burger and et al. Issues, Tasks and Program Structures to Roadmap Research in Question Answering. 2001.

[4] R. de Salvo Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. An Inference Model for Semantic Entailment in Natural Language. In *AAAI05*, pages 261–286, Illinois, USA, 2005.

[5] H. Doan-Nguyen and L. Kosseim. Using Terminology and a Concept Hierarchy for Restricted Domain Question Answering. In *Research on Computing Science, Special issue on Advances in Natural Language Processing*, pages 183–194, 2006.

[6] B. V. Durme, Y. Huang, A. Kupsc, and E. Nyberg. Towards light semantic processing for Question Answering. In *Proc. of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 54–61, NJ, USA, 2003.

[7] S. Harabagiu, A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. Question Answering with LCC's CHAUCER at TREC 2006. In *Proc. of the TREC 2006 Conference*, pages 283–292, Gaithersburg, MD, 2006.

[8] B. Katz and J. Lin. Selectively Using Relations to Improve Precision in Question Answering. In *Proc. of the EACL 2003 Workshop on NLP for Question Answering*, pages 50–60, Hungary, 2003.

[9] M. Kouylekov and H. Tanev. Document filtering and ranking using syntax and statistics for open domain question answering. In *Proc. of ESSLLI 2004 Workshop on Combining Shallow and Deep Processing for NLP*, pages 21–30, Nancy, France, 2004.

[10] A. K. Lamjiri, L. Kosseim, and T. Radhakrishnan. A Hybrid Unification Method for Question Answering in Closed Domains. In *Proc. of the 3rd International KRAQ'07 Workshop*, pages 36–42, Hyderabad, India, 2007.

[11] D. Lin. Principle-based Parsing without Overgeneration. In *Proc. of ACL-93*, pages 112–120, Ohio, USA, 1993.

[12] D. Lin. *Review of WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[13] J. Lin. The role of information retrieval in answering complex questions. In *COLING/ACL 2006 Poster Sessions*, pages 523–530, Sydney, Australia, July 2006.

[14] J. Lin and B. Katz. Question Answering from the Web Using Knowledge Annotation and Knowledge Mining Techniques. In *Proc. of CIKM'03*, pages 116 – 123, Louisiana, USA, 2003. ACM.

[15] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, and O. Bolohan. LCC Tools for Question Answering. In *Proc. of the TREC-11 Conference*, Gaithersburg, MD, 2002.

[16] D. Molla and M. Gardiner. AnswerFinder - Question Answering by combining lexical, syntactic and semantic information. In *Proc. of Australasian Language Technology Workshop (ALTW)*, pages 9–16, Sydney, 2004.

[17] T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proc. of he Nineteenth National Conference on Artificial Intelligence*, pages 1024–1025, San Jose, USA, 2004.

[18] V. Punyakanok, D. Roth, and W. tau Yih. Natural Language Inference via Dependency Tree Mapping: An Application to Question Answering. In *Computational Linguistics, Vol. 6, No. 9*, pages 1–10, 2004.

[19] R. Raina, A. Haghighi, C. Cox, J. Finkel, J. Michels, K. Toutanova, B. MacCartney, M.-C. de Marneffe, C. D. Manning, and A. Y. Ng. Robust Textual Inference using Diverse Knowledge Sources. In *Proc. of the First PASCAL Challenge*, pages 57–60, UK, 2005.

[20] R. Sun, J. Jiang, Y. F. Tan, H. Cui, T.-S. Chua, and M.-Y. Kan. Using Syntactic and Semantic Relation Analysis in Question Answering. In *Proc. of the TREC-13 Conference*, Gaithersburg, MD, 2004.

[21] H. Tanev, M. Kouylekov, B. Magnini, M. Negri, and K. Simov. Exploiting Linguistic Indices and Syntactic Structures for Multilingual Question Answering: ITC-irst at CLEF 2005. In *CLEF-2005 Working Notes*, pages 21–23, Vienna, Austria, 2005.

[22] E. Voorhees and D. Tice. The TREC-8 Question Answering Track Evaluation. In *Proc. of the TREC-8 Conference*, pages 83–106, Gaithersburg, MD, 1999.