Question Answering with Lexical Chains Propagating Verb Arguments

Adrian Novischi **Dan Moldovan**
Language Computer Corp.
1701 N. Collins Blvd, Richardson, TX, 75080
{adrian,moldovan}@languagecomputer.com

Abstract

This paper describes an algorithm for propagating verb arguments along lexical chains consisting of WordNet relations. The algorithm creates verb argument structures using VerbNet syntactic patterns. In order to increase the coverage, a larger set of verb senses were automatically associated with the existing patterns from VerbNet. The algorithm is used in an in-house Question Answering system for re-ranking the set of candidate answers. Tests on factoid questions from TREC 2004 indicate that the algorithm improved the system performance by 2.4%.

1 Introduction

In Question Answering the correct answer can be formulated with different but related words than the question. Connecting the words in the question with the words in the candidate answer is not enough to recognize the correct answer. For example the following question from TREC 2004 (Voorhees, 2004):

Q: (boxer Floyd Patterson) Who did he beat to win the title?

has the following wrong answer:

WA: He saw Ingemar Johanson knock down Floyd Patterson seven times there in winning the heavyweight title.

Although the above sentence contains the words *Floyd*, *Patterson*, *win*, *title*, and the verb *beat* can be connected to the verb *knock_down* using lexical chains from WordNet, this sentence does not answer the question because the verb arguments are in the wrong position. The proposed answer describes Floyd Patterson as being the object/patient of the *beating* event while in the question he is the subject/agent of the similar event. Therefore the selection of the correct answer from a list of candidate answers requires the check of additional constraints including the match of verb arguments.

Previous approaches to answer ranking, used syntactic partial matching, syntactic and semantic relations and logic forms for selecting the correct answer from a set of candidate answers. Tanev et al. (Tanev et al., 2004) used an algorithm for partial matching of syntactic structures. For lexical variations they used a dependency based thesaurus of similar words (Lin, 1998). Hang et al. (Cui et al., 2004) used an algorithm to compute the similarity between dependency relation paths from a parse tree to rank the candidate answers. In TREC 2005, Ahn et al. (Ahn et al., 2005) used Discourse Representation Structures (DRS) resembling logic forms and semantic relations to represent questions and answers and then computed a score "indicating how well DRSs match each other". Moldovan and Rus (Moldovan and Rus, 2001) transformed the question and the candidate answers into logic forms and used a logic prover to determine if the candidate answer logic form (ALF) entails the question logic form(QLF). Continuing this work Moldovan et al. (Moldovan et al., 2003) built a logic prover for Question Answering. The logic prover uses a relaxation module that is used iteratively if the proof fails at the price of decreasing the score of the proof. This logic prover was improved with temporal context detection (Moldovan et al., 2005).

All these approaches superficially addressed verb lexical variations. Similar meanings can be expressed using different verbs that use the same arguments in different positions. For example the sentence:

John bought a cowboy hat for $50
can be reformulated as:
John paid $50 for a cowboy hat.
The verb *buy* entails the verb *pay* however the arguments *a cowboy hat* and *$50* have different position around the verb.

This paper describes the approach for propagating the arguments from one verb to another using lexical chains derived using WordNet (Miller, 1995). The algorithm uses verb argument structures created from VerbNet syntactic patterns (Kipper et al., 2000b).

Section 2 presents VerbNet syntactic patterns and the machine learning approach used to increase the coverage of verb senses. Section 3 describes the algorithms for propagating verb arguments. Section 4 presents the results and the final section 5 draws the conclusions.

2 VerbNet Syntactic Patterns

The algorithm for propagating verb arguments uses structures for representing them. Several choices were considered for retrieving verbs' argument structure. Verb syntactic patterns from WordNet (called frames) could not be used because some tokens in the patterns (like "PP" or "CLAUSE") cannot be mapped to arguments. FrameNet (Baker et al., 1998) and PropBank (Kingsbury and Palmer, 2002) contain verb syntactic patterns, but they do not have a mapping to WordNet. Finally VerbNet (Kipper et al., 2000b) represents a verb lexicon with syntactic and semantic information. This resource has a mapping to WordNet and therefore was considered the most suitable for propagating predicate arguments along lexical chains.

2.1 VerbNet description

VerbNet is based on classes of verbs. Each verb entry points to a set of classes and each class represents a sense of a verb. The classes are organized hierarchically. Each class contains a set of syntactic patterns corresponding to licensed constructions. Each syntactic pattern is an ordered list of tokens and each token represents a group of words. The tokens contain various information and constraints about the word or the group of words they represent. The name of the token can represent the thematic role of an argument, the verb itself, prepositions, adjectives, adverbs or plain words.
VerbNet uses 29 thematic roles (presented in table 1). VerbNet has a static aspect and a dynamic aspect. The static aspect refers to the organization of verb entries. The dynamic aspect refers to the lexicalized trees associated with syntactic patterns. A detailed description of VerbNet dynamic aspect can be found in (Kipper et al., 2000a).

Table 1: VerbNet thematic roles

Thematic Roles		
Topic	Experiencer	Stimulus
Cause	Actor	Actor1
Actor2	Agent	Asset
Attribute	Benefactor	Benefi ciary
Destination	Instrument	Location
Material	Patient	Patient1
Patient2	Predicate	Product
Recipient	Source	Theme
Theme1	Theme2	Time
Extent	Value	

The algorithm for propagating predicate arguments uses the syntactic patterns associated with each sensekey. Each class contains a set of WordNet verb sensekeys and a set of syntactic patterns. Therefore, syntactic patterns can be associated with verb sensekey from the same class. Since sensekeys represent word senses in WordNet, each verb synset can be associated with a set of VerbNet syntactic patterns. VerbNet syntactic patterns allow predicate arguments to be propagated along lexical chains. However, not all verb senses in WordNet are listed in VerbNet classes. For the remaining verb sensekeys that are not listed in VerbNet, syntactic patterns were assigned automatically using machine learning as described in the following section.

2.2 Associating syntactic patterns with new verb senses

In order to propagate predicate arguments along lexical chains, ideally every verb in every synonym set has to have a set of syntactic patterns. Only a part of verb senses are listed in VerbNet classes. WordNet 2.0 has 24,632 verb sensekeys, but only 4,983 sensekeys are listed in VerbNet classes. For the rest, syntactic patterns were assigned automatically. In order to assign these syntactic patterns to the verb senses not listed in VerbNet, training examples were needed, both positive and negative. The learning took place for one syntactic pattern at a time. A syntactic pattern can be listed in more than one class. All verb senses associated with a syntactic pattern can be considered positive examples of verbs having that syntactic pattern. For generating negative examples, the following assumption was used: if a verb sense listed in a VerbNet class is not associated with a given syntactic pattern, then that verb sense represents a negative example for that pattern. 352 syntactic patterns were found in all VerbNet classes. A training example was generated for each pair of syntactic patterns and verb sensekeys, resulting in a total number of 1,754,016 training examples. These training examples were used to infer rules that would classify if a verb sense key can be associated with a given syntactic pattern. Training examples were created by using the following features: verb synset semantic category, verb synset position in the IS-A hierarchy, the fact that the verb synset is related to other synsets with CAUSATION relation, the semantic classes of all noun synsets derivationally related with the given verb synset and the WordNet syntactic pattern ids. A machine learning algorithm based on C5.0 (Quinlan, 1998) was run on these training examples. Table 2 presents the performance of the learning algorithm using a 10-fold cross validation for several patterns. A number of 20,759 pairs of verb senses with their syntactic patterns were added to the existing 35,618 pairs in VerbNet. In order to improve the performance of the question answering system, around 100 patterns were manually associated with some verb senses.

Table 2: Performance of learning verb senses for several syntactic patterns

Id	Pattern	Performance
0	<Agent> <VERB> <Theme>	74.2%
1	<Experiencer> <VERB> <Cause>	98.6%
2	<Experiencer> <VERB> <Oblique> for <Cause>	98.7%
3	<Experiencer> <VERB> <Cause> in <Oblique>	98.7%
4	<Agent> <VERB> <Recipient>	94.7%
5	<Agent> <VERB> <Patient>	85.6%
6	<Patient> <VERB> <ADV>	85.1%
...	...	...
348	<Agent> <VERB> <Patient> at <Cause>	99.8%
349	<Agent> <VERB> in <Theme>	99.8%
350	<Agent> <VERB> <Source> <ADJ>	99.5%
351	<Agent> <VERB> at <Source>	99.3%

3 Propagating Verb Arguments

Given the argument structure of a verb in a sentence and a lexical chain between this verb and another, the algorithm for propagating verb arguments transforms this structure step by step, for each relation in the lexical chain. During each step the head of the structure changes its value and the arguments can change their position. The arguments change their position in a way that preserves the original meaning as much as possible. The argument structures mirror the syntactic patterns that a verb with a given sense can have. An argument structure contains the type of the pattern, the head and an array of tokens. Each token represents an argument with a thematic role or an adjective, an adverb, a preposition or just a regular word. The head and the arguments with thematic roles are represented by concepts. A concept is created from a word found in text. If the word is found in WordNet, the concept structure contains its surface form, its lemma, its part of speech and its WordNet sense. If the word is not found in WordNet, its concept structure contains only the word and the part of speech. The value of the field for an argument is represented by the concept that is the head of the phrase representing the argument. Because a synset may contain more than one verb and each verb can have different types of syntactic patterns, propagation of verb arguments along a single relation can result in more than one structure. The output of the algorithm as well as the output of the propagation of each relation in the lexical chain is the set of argument structures with the head being a verb from the set of synonyms of the target synset. For a given relation in the lexical chain, each structure coming from the previous step is transformed into a set of new structures. The relations used and the process of argument propagation is described below.

3.1 Relations used

A restricted number of WordNet relations were used for creating lexical chains. Lexical chains between verbs were used for propagating verb arguments, and lexical chains between nouns were used to link semantically related arguments expressed with different words.

Between verb synsets the following relations were used: HYPERNYM, TROPONYM, ENTAILMENT and CAUSATION. These relations were selected because they reveal patterns about how they propagate predicate arguments.

The HYPERNYMY relation links one specific verb synset to one that is more general. Most of the time, the arguments have the same thematic roles for the two verbs. Sometimes the hypernym synset has a syntactic pattern that has more thematic roles than the syntactic pattern of the start synset. In this case the pattern of the hypernym is not considered for propagation.

The HYPONYMY relation is the reverse of HYPERNYMY and links one verb synset to a more specific one. Inference to a more specific verb requires abduction. Most of the time, the arguments have the same thematic roles for the two verbs. Usually the hyponym of the verb synset is more specific and have less syntactic patterns than the original synset. This is why a syntactic pattern of a verb can be linked with the syntactic pattern of its hyponym that has more thematic roles. These additional thematic roles in the syntactic pattern of the hyponym will receive the value ANY-CONCEPT when verb arguments are propagated along this relation.

ENTAILMENT relation links two verb synsets that express two different events that are related: the first entails the second. This is different than HYPERNYMY or HYPONYMY that links verbs that express the same event with more or less details. Most of the time the subject of these two sentences has the same thematic role. If the thematic role of subjects is different, then the syntactic pattern of the target verb is not considered for propagation. The same happens if the start pattern contains less arguments than the target pattern. Additional arguments can change the meaning of the target pattern.

A relation that is the reverse of the ENTAILMENT is not coded in WordNet but, it is used for a better connectivity. Given one sentence S_1 with a verb V_1 that is entailed by a verb V_2, the sentence S_1 can be reformulated using the verb V_2, and thus creating sentence S_2. Sentence S_1 does not imply sentence S_2 but makes it plausible. Most of the time, the subject of these two sentences has the same thematic role. If the thematic role of subjects is different, then the pattern of the target verb synset is not considered for propagation. The same happens if the start pattern has less arguments than the target pattern. Additional arguments can change the meaning of the target pattern.

The CAUSATION relation puts certain restrictions on the syntactic patterns of the two verb synsets. The first restriction applies to the syntactic pattern of the start synset: its subject must be an *Agent* or an *Instrument* and its object must be a *Patient*. The second restriction applies to the syntactic pattern of the destination synset: its subject must be a *Patient*. If the two syntactic patterns obey these restrictions then an instance of the destination synset pattern is created and its arguments will receive the value of the argument with the same thematic role in the pattern belonging to start synset.

The reverse of the CAUSATION relation is not codified in WordNet database but it is used in lexical chains to increase the connectivity between synsets. Similar to causation relation, the reverse causation imposes two restrictions on the patterns belonging to the start and destination synset. First restriction applies to the syntactic pattern of the start synset: its subject must have the thematic role of *Patient*. The second restriction applies to the syntactic pattern of the destination synset: its subject must be an *Agent* or an *Instrument* and its object must be a *Patient*. If the two syntactic patterns obey these restrictions then an instance of the destination synset pattern is created and its arguments will receive the value of the argument with the same thematic role in the pattern belonging to start synset.

When deriving lexical chains for linking words from questions and correct answers in TREC 2004, it was observed that many chains contain a pair of DERIVATION relations. Since a pair of DERIVATION relations can link either two noun synsets or two verb synsets, the pair was concatenated into a new relation called SIM_DERIV. The number of SIM-DERIV relations is presented in table 3. For example the verb synsets *emanate#2* and *emit#1* are not synonyms (not listed in the same synset) but they are linked by a SIM-DERIV relation (both have a DERIVATION relation to the noun synset *(n-emission#1, emanation#2)* - nominalizations of the two verbs are listed in the same synset). There are no restrictions between pairs of patterns that participate in argument propagation. The arguments in the syntactic pattern instance of the destination synset take their values from the arguments with the same thematic roles from the syntactic pattern instance of the start synset.

Table 3: The SIM-DERIV relations generated for nouns and verb .

Relation	**Source**	**Target**	**Number**
SIM-DERIV	noun	noun	45,178
SIM-DERIV	verb	verb	15,926

The VERBGROUP and SEE-ALSO relations were not included in the experiment because it is not clear how they propagate arguments.

A restricted set of instances of DERIVATION relation was used to link verbs to nouns that describe their action. When arguments are propagated from verb to noun, the noun synset will receive a set of syntactic patterns instances similar to the semantic instances of the verb. When arguments are propagated from noun to verb, a new created structure for the verb sense takes the values for its arguments from the arguments with similar thematic roles in the noun structure.

Between the heads of two argument structures there can exist lexical chains of size 0, meaning that the heads of the two structures are in the same synset. However, the type of the start structure can be different than the type of the target structure. In this case, the arguments still have to be propagated from one structure to another. The arguments in the target structure will take the values of the arguments with the same thematic role in the start structure or the value ANY-CONCEPT if these arguments cannot be found.

Relations between nouns were not used by the algorithm but they are used after the algorithm is applied, to link the arguments from a resulted structure to the arguments with the same semantic roles in the target structure. If such a link exists, then the arguments are considered to match. From the existing WordNet relations between noun synsets only HYPERNYM and HYPONYM were used.

3.2 Assigning weights to the relations

Two synsets can be connected by a large number of lexical chains. For efficiency, the algorithm runs only on a restricted number of lexical chains. In order to select the most likely lexical chains, they were ordered decreasingly by their weight. The weight of a lexical chain is computed using the following formula inspired by (Moldovan and Novischi, 2002):

$$W = W_1 * A_{1,2} * MG_1 * W_2 * A_{2,3} * MG_2 * ... * A_{n-1,n} * MG_{n-1} * W_n$$

where n represents the number of relations in the lexical chain. The formula uses the weights W_i ($i = 1..n$) of the relations along the chain (presented in table 4) and coefficients for pairs of relations $A_{i,i+1}$ (some of them presented in table 5, the rest having a weight of 1.0). This formula resulted from the observation that the relations are not equal (some relations like HYPERNYMY are stronger than other relations) and that the order of relations in the lexical chain influences its fitness (the order of relations is approximated by the weight given to pairs of relations). The formula uses the "measure of generality" of a concept defined as:

$$MG_C = \frac{500}{500+N_{RGLOSS}}$$

where N_{RGLOSS} represents the number of occurrences of a given concept in WordNet glosses.

Table 4: The weight assigned to each relation

Relation	Weight
HYPERNYM	0.8
HYPONYM	0.7
DERIVATION	0.6
ENTAILMENT	0.7
R-ENTAILMENT	0.6
CAUSATION	0.7
R-CAUSATION	0.6

Table 5: Some of the weights assigned to pair of relations

Relation 1	Relation 2	Coefficient Weight
HYPERNYM	HYPONYM	1.25
HYPERNYM	ENTAILMENT	1.25
HYPERNYM	R-ENTAILMENT	0.8
HYPERNYM	CAUSATION	1.25
HYPERNYM	R-CAUSATION	1.25
HYPONYM	HYPERNYM	0.8
HYPONYM	ENTAILMENT	1.25
HYPONYM	R-ENTAILMENT	0.8
HYPONYM	CAUSATION	1.25
HYPONYM	R-CAUSATION	0.8
ENTAILMENT	HYPERNYM	1.25
ENTAILMENT	HYPONYM	0.8
ENTAILMENT	CAUSATION	1.25
ENTAILMENT	R-CAUSATION	0.8
R-ENTAILMENT	HYPERNYM	0.8
R-ENTAILMENT	HYPONYM	0.8
R-ENTAILMENT	CAUSATION	0.8
R-ENTAILMENT	R-CAUSATION	1.25
CAUSATION	HYPERNYM	1.25
CAUSATION	HYPONYM	0.8
CAUSATION	ENTAILMENT	1.25
CAUSATION	R-ENTAILMENT	0.8

3.3 Example

In the test set from the QA track in TREC 2004 we found the following question with correct answer:

Q 28.2: (Abercrombie & Fitch) When was it established?
A: ... Abercrombie & Fitch began life in 1982 ...

The verb *establish* in the question has sense 2 in WordNet 2.0 and the verb *begin* in the answer has also sense 2. The following lexical chain can be found between these two verbs:

(v-begin#2,start#4)
R-CAUSATION
(v-begin#3,lead_off#2,start#2,commence#2)
SIM-DERIV
(v-establish#2,found#1)

From the question, an argument structure is created for the verb *establish#2* using the following pattern:
<Agent> establish#2 <Patient>
where the argument with the thematic role of *Agent* has the value ANY-CONCEPT, and the *Patient* argument has the value *Abercrombie & Fitch*.

From the answer, an argument structure is created for verb *begin#2* using the pattern:
<Patient> begin#2 <Theme>
where the *Patient* argument has the value *Abercrombie & Fitch* and the *Theme* argument has the value *n-life#2*. This structure is propagated along the lexical chain, each relation at a time. First for the R-CAUSATION relation links the verb *begin#2* having the pattern:
<Patient> <Verb> <Theme>
with the verb *begin#3* that has the pattern:
<Agent> begin#3 <Patient>
The *Patient* keeps its value *Abercrombie &Fitch* event though it is changing its syntactic role from subject of the verb *begin#2* to the object of the verb *begin#3*. The *Theme* argument is lost along this relation, instead the new argument with the thematic role of *Agent* receives the special value ANY-CONCEPT.

The second relation in the chain, SIM-DERIV links two verbs that have the same syntactic pattern:
<Agent> <Verb> <Patient>
Therefore a new structure is created for the verb *establish#2* using this pattern and its arguments take their values from the similar arguments in the argument structure for verb *begin#3*. This new structure exactly matches the argument structure from the question therefore the answer is ranked the highest in the set of candidate answer. Figure 1 illustrates the argument propagation process for this example.

4 Experiments and Results

The algorithm for propagating verb arguments was used to improve performance of an in-house Question Answering system (Moldovan et al., 2004). This improvement comes from a better matching between a question and the sentences containing the correct answer. Integration of this algorithm into the Question Answering system requires 3 steps: (1) creation of structures containing verb arguments for the questions and its possible answers, (2) derivation of lexical chains between the two structures and propagation of the arguments along lexical chains, (3) measuring the similarity between the propagated structures and the structures from the question and re-ranking of the candidate answers based on similarity scores. Structures containing predicate arguments are created for all the verbs in the question and all verbs in each possible answer. The QA system takes care of coreference resolution.

Argument structures are created for verbs in both active and passive voice. If the verb is in passive voice, then its arguments are normalized to active voice. The subject phrase of the verb in passive voice represents its object and the noun phrase inside prepositional phrase with preposition "by" becomes its subject. Special attention is given to di-transitive verbs. If in passive voice, the subject phrase can represent either the direct object or indirect object. The distinction is made in the following way: if the verb in passive voice has a direct object then the subject represents the indirect object (beneficiary), otherwise the subject represents direct object. All the other arguments are treated in the same way as in the active voice case.

After the structures are created from a candidate answer and a question, lexical chains are created between their heads. Because lexical chains link two word senses, the heads need to be disambiguated. Before searching for lexical chains, the heads could be already partially disambiguated, because only a restricted number of senses of the head verb can have the VerbNet syntactic pattern matching the input text. An additional semantic disambiguation can take place before deriving lexical chains. The verbs from the answer and question can also be disambiguated by selecting the best lexical chain between them. This was the approach used in our experiment.

The algorithm propagating verb arguments was tested on a set of 106 pairs of phrases with similar meaning for which argument structures could be built. These phrases were selected from pairs of questions and their correct answers from the

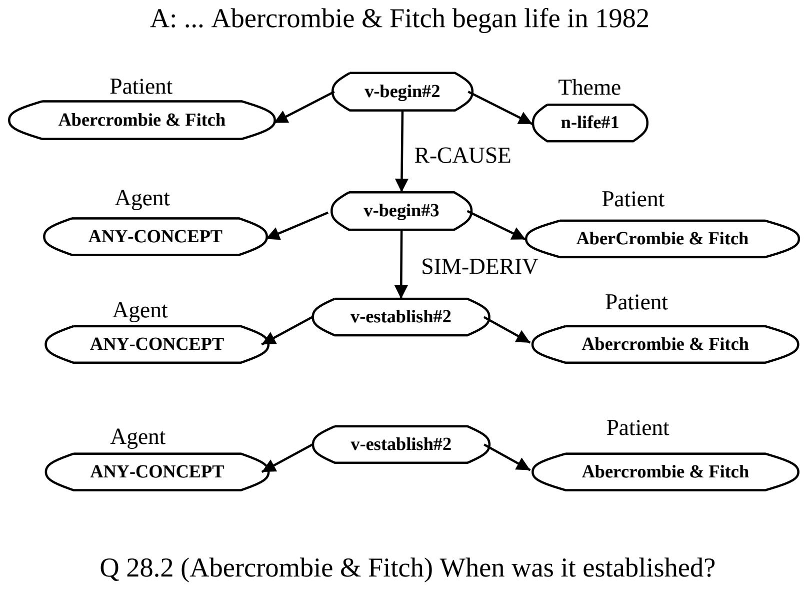

Figure 1: Example of lexical chain that propagates syntactic constraints from answer to question.

set of factoid questions in TREC 2004 and also from the pairs of scenarios and hypotheses from first edition of PASCAL RTE Challenge (Dagan et al., 2005). Table 6 shows algorithm performance. The columns in the table correspond to the following cases:
a) how many cases the algorithm propagated all the arguments;
b) how many cases the algorithm propagated one argument;
c) home many cases the algorithm did not propagate any argument;
using top 5, 20, 50 lexical chains.

The purpose of the algorithm for propagating predicate arguments is to measure the similarity between the sentences for which the argument structures have been built. This similarity can be computed by comparing the target argument structure with the propagated argument structure. The similarity score is computed in the following way: if N represents the number of arguments in a pattern, each argument matched is defined to have a contribution of $1/(N+1)$, except for the subject that has a contribution if matched of 2/(N+1). The propagated pattern is compared with the target pattern and the score is computed by summing up the contributions of all matched arguments.

The set of factoid questions in TREC 2004 has 230 questions. Lexical chains containing the restricted set of relations that propagate verb arguments were found for 33 questions, linking verbs in those questions to verbs in their correct answer. This is the maximum number of questions on which the algorithm for propagating syntactic constraints can have an impact without using other knowledge. The algorithm for propagating verb argument could be applied on 15 of these questions. Table 7 shows the improvement of the Question Answering system when the first 20 or 50 answers returned by factoid strategy are re-ranked according to similarity scores between argument structures. The performance of the question answering system was measured using Mean Reciprocal Rank (MRR).

Table 7: The impact of the algorithm for propagating predicate arguments over the question answering system

Number of answers	Performance
Top 20	1.9%
Top 50	2.4%

5 Conclusion

This paper describes the approach of propagating verb arguments along lexical chains with WordNet relations using VerbNet frames. Since VerbNet frames are not associated with all verb senses from WordNet, some verb senses were added automatically to the existing VerbNet frames. The algorithm was used to improve the performance of the answer's ranking stage in Question Answering system. Only a restricted set of WordNet semantic

Table 6: The performance of the algorithm for propagating predicate arguments with semantic constraints

	Arguments propagated	Top 5 chains	Top 10 chains	Top 20 chains
a	all arguments	23(21.6%)	28(26.4%)	32(30.2%)
b	at least one argument	73(68.8%)%	81(76.4%)	89(83.9%)
c	no arguments	32(30.2%)	25(23.6%)	17(16.0%)

relations were used to propagate predicate arguments. Lexical chains were also derived between the arguments for a better match. On the set of factoid questions from TREC 2004, it was found that for 33(14.3%) questions, the words in the question and the related words in the answer could be linked using lexical chains containing only the relations from the restricted set that propagate verb arguments. Overall, the algorithm for propagating verb arguments improved the system performance with 2.4%

References

Kisuh Ahn, Johan Bos, James R. Curran, Dave Kor, Malvina Nissim, and Bonnie Webber. 2005. Question Answering with QED at TREC-2005. In *Proceedings of TREC 2005*.

Collin F. Baker, Charles J. Fillmore, , and John B Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the COLING-ACL*, Montreal, Canada.

Hang Cui, Keya Li, Renxu Sun, Tat-Seng Chua, and Min-Yen-Kan. 2004. National University of Singapore at the TREC-13 Question Answering Main Task. In *Proceedings of the 13th Text Retrieval Conference (TREC-2004)*, Gaithersburg, Maryland, USA, November 16-19.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. Recognising Textual Entailment Challenge, http://www.pascal-network.org/Challenges/RTE, March.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Spain.

K. Kipper, H. Dang, W. Schuler, and M. Palmer. 2000a. Building a class-based verb lexicon using tags. In *Proceedings of Fifth TAG+ Workshop*.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000b. Class-based construction of a verb lexicon. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 691–696. AAAI Press / The MIT Press.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL-98*, Montreal, Canada, August.

G. Miller. 1995. WordNet: a lexical database. *Communications of the ACM*, 38(11):39–41, November.

Dan Moldovan and Adrian Novischi. 2002. Lexical chains for question answering. In *Proceedings of COLING 2002*, pages 674–680.

Dan I. Moldovan and Vasile Rus. 2001. Logic Form Transformation of WordNet and its Applicability to Question Answering. In *Proceedings of the ACL 2001*, Toulouse, France, July.

Dan I. Moldovan, Christine Clark, Sanda M. Harabagiu, and Steven J. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL 2003*, Edmonton, Canada, May-June.

Dan Moldovan, Sanda Harabagiu, Christine Clark, and Mitchell Bowden. 2004. PowerAnswer 2: Experiments and Analysis over TREC 2004. In *Proceedings of Text Retrieval Conference 2004*.

Dan Moldovan, Christine Clark, and Sanda Harabagiu. 2005. Temporal Context Representation and Reasoning. In *Proceedings of IJCAI-2005*, pages 1099–1104, Edinburgh, Scotland, July-August.

R. Quinlan. 1998. C5.0: An Informal Tutorial, RuleQuest.

H. Tanev, M. Kouylekov, and B. Magnini. 2004. Combining linguistic processing and web mining for question qnswering: Itc-irst at trec 2004. In *Proceedings of the 13th Text Retrieval Conference (TREC-2004)*, pages 429–438, Gaithersburg, Maryland, USA, November 16-19.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 Question Answering Track. In *Proceedings of the 13th Text Retrieval Conference (TREC-2004)*, pages 83–105, Gaithersburg, Maryland, USA, November 16-19.