Lexical Paraphrasing for Document Retrieval and Node Identification

Ingrid Zukerman
School of Computer Science and Software Engineering
Monash University
Clayton, VICTORIA 3800
AUSTRALIA
ingrid@csse.monash.edu.au

Sarah George
School of Computer Science and Software Engineering
Monash University
Clayton, VICTORIA 3800
AUSTRALIA
sarahg@csse.monash.edu.au

Yingying Wen
School of Computer Science and Software Engineering
Monash University
Clayton, VICTORIA 3800
AUSTRALIA
ywen@csse.monash.edu.au

Abstract

We investigate lexical paraphrasing in the context of two distinct applications: document retrieval and node identification. Document retrieval – the first step in question answering – retrieves documents that contain answers to user queries. Node identification – performed in the context of a Bayesian argumentation system – matches users' Natural Language sentences to nodes in a Bayesian network. Lexical paraphrases are generated using syntactic, semantic and corpus-based information. Our evaluation shows that lexical paraphrasing improves retrieval performance for both applications.

1 Introduction

One of the difficulties users face when accessing a knowledge repository is that of expressing themselves in a way that will produce the desired outcome, e.g., retrieve relevant documents or make their dialogue contributions understood. The use of appropriate terminology is one aspect of this problem: if a user's vocabulary differs from that within the resource being accessed, the system may be unable to satisfy the user's requirements. In this paper, we investigate the application of lexical paraphrasing to two different information access applications: document retrieval and node identification. Document retrieval is performed in the context of the TREC Question Answering task, where the system retrieves documents that contain answers to users' queries. Node identification is performed in the context of a Natural Language (NL) interface to a Bayesian argumentation system (Zukerman and George, 2002). Here the system finds the nodes from a Bayesian network (BN) (Pearl, 1988) that best match a user's NL sentences.

Lexical paraphrases replace content words in a user's input with their synonyms. We use the following information sources to perform this task: syntactic – obtained from Brill's part-of-speech tagger (Brill, 1992); semantic – obtained from WordNet (Miller et al., 1990) and the Webster-1913 online dictionary; and statistical – obtained from our document collection. The statistical information is used to moderate the alternatives obtained from the semantic resources, by preferring query paraphrases that contain frequent word combinations.

Our evaluation assessed the effect of lexical paraphrasing on our two applications. Its impact on document retrieval was evaluated using subsets of queries from the TREC8, TREC9 and TREC10 collections, and its effect on node identification was evaluated using paraphrases generated by people for some nodes in our BN (Section 6.2).

In the next section we discuss related research. Section 3 describes our two applications. In Section 4, we consider the paraphrasing process, and in Section 5 the information retrieval procedures. Section 6 presents the results of our evaluation, followed by concluding remarks.

2 Related Research

The vocabulary mis-match between a user's queries and indexed documents is often addressed through

query expansion. Query expansion in turn is often preceded by *Word sense disambiguation (WSD)* in order to avoid retrieving irrelevant information. Mihalcea and Moldovan (1999) and Lytinen *et al.* (2000) used WordNet (Miller et al., 1990) to obtain the sense of a word. In contrast, Schütze and Pedersen (1995) and Lin (1998) used a corpus-based approach where they automatically constructed a thesaurus on the basis of contextual information. The results obtained by Schütze and Pedersen and by Lytinen *et al.* are encouraging. However, experimental results reported in (Gonzalo et al., 1998) indicate that the improvement in IR performance due to WSD is restricted to short queries, and that IR performance is very sensitive to disambiguation errors. Harabagiu *et al.* (2001) offered a different form of query expansion, where they used WordNet to propose synonyms for the words in a query, and applied heuristics to select which words to paraphrase.

Our approach to the vocabulary mis-match problem differs from WSD in that instead of returning the sense of the words in a sentence, we propose alternative lexical paraphrases. Like Langkilde and Knight (1998), when generating candidate paraphrases, we use WordNet to propose synonyms for the words in a sentence. However, they chose among these synonyms using the word-sense rankings offered by WordNet, while we make our selection using word-pair frequencies obtained from our corpus, and word-similarity information obtained from a thesaurus that is automatically constructed from the Webster Dictionary. It is worth noting that the retrieval performance obtained with paraphrases generated using our dictionary-based thesaurus compares favorably with that obtained using Lin's context-based thesaurus.

3 Applications

Our two applications are: document retrieval (`DocRet`) and node identification (`NodeID`).

The document retrieval application consists of retrieving documents that are likely to contain the answer to a user's query. This is the first step in our Question Answering project, whose aim is to use these documents to generate answers to queries.

The node identification application is the backbone of the NL interface to our *Bayesian Interactive Argumentation System (BIAS)* (Zukerman and George, 2002). Here the system matches NL sentences that make up a user's argument to nodes in a BN that represents the system's domain knowledge. The eventual goal is to understand the user's argument (in terms of the system's domain knowledge) and generate responses.

There are significant differences between these applications and their underlying domains:

Our `DocRet` repository consists of 131,896 articles from the LA Times portion of the NIST TREC collection (the other TREC repositories were omitted owing to disk space limitations). Full-text indexing was performed for these documents using lemmas (uninflected versions of words), rather than stems or complete words. Our `NodeID` repository is an 85-node BN which represents domain knowledge for a murder mystery. Most nodes are associated with one "canonical" sentence, and 8 nodes are associated with two or three sentences. Sample canonical sentences for the nodes in our domain are "Mr Green murdered Mr Body" and "The blue paint on the mailbox is one week old". The sentences in this domain were also lemmatized, but no indexing was performed owing to the small size of the BN.

The queries in the `DocRet` application have an average length of 7.22 words, and the articles in the LA Times repository have 242 words on average. In contrast, in the `NodeID` application, the user's sentences and the sentences associated with the nodes in the BN are of similar length (8.45 words on average for the user sentences, compared to 7.7 words for the BN sentences).

The `DocRet` task is one of *containment*, i.e., to retrieve documents that include the words in a query, while the `NodeID` task is one of *equivalence*, i.e., to retrieve the node that best matches the user's sentence. Further, for `DocRet` there may be several articles that answer a user's query, while for `NodeID` at most one node matches a user's sentence.[1]

4 Lexical Paraphrasing

In this section, we discuss the resources used by our paraphrasing mechanism and describe the paraphrasing process.

[1] The user may utter sentences that have no counterpart in the system's BN. Also, at present we are not dealing with sentences that include more than one node.

4.1 Resources

Our system uses syntactic, semantic and statistical information for paraphrase generation.

Syntactic information for each sentence was obtained from Brill's part-of-speech (PoS) tagger (Brill, 1992).

Semantic information was obtained from two sources: WordNet – a knowledge-intensive, hand-built on-line repository; and Webster – an on-line version of the Webster-1913 dictionary (`http://www.dict.org`). WordNet was used to generate lemmas for the corpus and the sentences, and to generate different types of synonyms for the words in the sentences. Webster was used to automatically build a thesaurus that includes similarity scores between lemmas, and to automatically construct a list of nominals corresponding to the verbs in the dictionary, and a list of verbs corresponding to the nouns in the dictionary. The thesaurus was used to assign similarity scores to the synonyms returned by WordNet. The nominalization and verbalization lists were used to generate additional synonyms. This was done by activating WordNet for the nouns corresponding to the verbs in the sentences and for the verbs corresponding to the nouns in the sentences.[2] The idea was that nominalizations and verbalizations will help paraphrase queries such as "who *killed* Lincoln?" into "who is the *murderer* of Lincoln?" (Harabagiu et al., 2001).

The thesaurus, nominal list and verb list were obtained by building a vector from the content lemmas in the definition of each word in the dictionary, and applying the cosine similarity measure to calculate a score for the similarity between the vector corresponding to each word and the vectors corresponding to the other words in the dictionary. The dictionary words that had the highest similarity score against the original word were retained. To build the thesaurus, only words with the same PoS as the original word were considered. To build the nominalization list, only nouns were considered for each verb in the dictionary, and to build the verbalization list, only verbs were considered for each noun. Further, for these lists, the retained noun (or verb) had to have the same stem as the original verb (or noun).

[2]It was necessary to build nominalization and verbalization lists because WordNet does not include this information.

Statistical information was obtained from the LA Times portion of the NIST Text Research Collection (`http://trec.nist.gov`). The statistical information was used to estimate the probability of the paraphrases generated for a sentence (Section 5.1). The statistical information was stored in a lemma dictionary (202,485 lemmas) and a lemma-pair dictionary (37,341,156 lemma pairs). Lemma pairs which appear only once constitute 64% of the pairs, and were omitted owing to disk space limitations.

4.2 Procedure

The following procedure was applied to paraphrase a sentence:

1. Tokenize, tag and lemmatize the sentence.
2. Generate replacement lemmas for each content lemma in the sentence.
3. Propose paraphrases for the sentence using different combinations of replacement lemmas, estimate the probability of each paraphrase, and rank the paraphrases according to their probabilities. Retain the lemmatized sentence plus the top K paraphrases.

Documents or nodes are then retrieved for the sentence and its paraphrases, the probability of each retrieved item is calculated, and the top N items are retained (Section 5).

4.2.1 Tagging and lemmatizing sentences

We used Brill's tagger (Brill, 1992) to obtain the PoS of a word. This PoS is used to constrain the number of synonyms generated for a word. Brill's tagger incorrectly tagged 16% of the queries in the `DocRet` application, which had a marginal detrimental effect on retrieval performance (Zukerman and Raskutti, 2002). After tagging, each sentence was lemmatized using WordNet.

4.2.2 Proposing replacements for each lemma

We used WordNet and the nominalization and verbalization lists built from Webster to propose replacements for the content lemmas in a sentence. These resources were used as follows:

1. For each word in the sentence, we determined its lemma(s) and the lemma(s) that verbalize it (if it is a noun) or nominalize it (if it is a verb).

2. We then used WordNet to propose different types of synonyms for the lemmas produced in the first step. These types of synonyms were: `synonyms`, `attributes`, `pertainyms` and `seealsos` (Miller et al., 1990). For example, according to WordNet, a `synonym` for "high" is "steep", an `attribute` is "height", and a `seealso` is "tall"; a `pertainym` for "chinese" is "China". `hypernyms` and `hyponyms` were optionally included depending on the approach used to represent the similarity between two lemmas (Section 4.2.4).

4.2.3 Paraphrasing sentences

Sentence paraphrases were generated by an iterative process which considers each content lemma in a sentence in turn, and proposes a replacement lemma from those collected from our information sources (Section 4.2.2). Sentences which did not have sufficient context were not paraphrased. These are sentences where all the words except one are closed-class words or stop words (frequently occurring words that are ignored when used as search terms).

4.2.4 Probability of a paraphrase

The probability that a paraphrase is an appropriate rendition of a sentence depends on two factors: (1) how similar is the paraphrase to the sentence, and (2) how common are the lemma combinations in the paraphrase. This may be expressed as follows:

$$\Pr(\text{Para}_i|\text{Sent}) = \frac{\Pr(\text{Sent}|\text{Para}_i) \times \Pr(\text{Para}_i)}{\Pr(\text{Sent})} \quad (1)$$

where Para_i is the ith paraphrase of a sentence. Since the probability of the denominator is constant for a given sentence, we obtain:

$$\Pr(\text{Para}_i|\text{Sent}) \propto \Pr(\text{Sent}|\text{Para}_i) \times \Pr(\text{Para}_i) \quad (2)$$

where

$$\Pr(\text{Sent}|\text{Para}_i) = \Pr(\text{Slem}_1, \ldots, \text{Slem}_L|\text{lem}_{i,1}, \ldots, \text{lem}_{i,L}) \quad (3)$$

$$\Pr(\text{Para}_i) = \Pr(\text{lem}_{i,1}, \ldots, \text{lem}_{i,L}) \quad (4)$$

where L is the number of content words in a sentence, $\Pr(\text{Slem}_j)$ is the probability of using Slem_j – the jth lemma in the sentence, and $\Pr(\text{lem}_{i,j})$ is the probability of using $\text{lem}_{i,j}$ – the jth lemma in the ith paraphrase of the sentence.

To calculate $\Pr(\text{Sent}|\text{Para}_i)$ in Eqn. 3 we assume (1) $\Pr(\text{Slem}_k|\text{lem}_{i,1}, \ldots, \text{lem}_{i,L})$ is independent of $\Pr(\text{Slem}_j|\text{lem}_{i,1}, \ldots, \text{lem}_{i,L})$ for $k, j = 1, \ldots, L$ and $k \neq j$, and
(2) given $\text{lem}_{i,k}$, Slem_k is independent of the other lemmas in the sentence paraphrase, i.e., $\Pr(\text{Slem}_k|\text{lem}_{i,1}, \ldots, \text{lem}_{i,L}) \cong \Pr(\text{Slem}_k|\text{lem}_{i,k})$. These assumptions yield

$$\Pr(\text{Sent}|\text{Para}_i) \cong \prod_{j=1}^{L} \Pr(\text{Slem}_j|\text{lem}_{i,j}) \quad (5)$$

Eqn. 4 may be rewritten using Bayes rule:

$$\Pr(\text{Para}_i) \cong \prod_{j=1}^{L} \Pr(\text{lem}_{i,j}|\text{ctxt}_{i,j}) \quad (6)$$

where $\text{ctxt}_{i,j}$ is the context for lemma j in the ith paraphrase.

Substituting Eqn. 5 and Eqn. 6 into Eqn. 2 yields

$$\Pr(\text{Para}_i|\text{Sent}) \propto \prod_{j=1}^{L} [\Pr(\text{Slem}_j|\text{lem}_{i,j}) \times \Pr(\text{lem}_{i,j}|\text{ctxt}_{i,j})] \quad (7)$$

$\boldsymbol{Pr}(\boldsymbol{Slem}_j|\boldsymbol{lem}_{i,j})$ may be interpreted as the probability of using Slem_j instead of $\text{lem}_{i,j}$. Intuitively, this probability depends on the similarity between the lemmas. We considered two approaches for representing this probability:

Baseline similarity:
$\Pr(\text{Slem}_j|\text{lem}_{i,j}) = 1$ if Slem_j is a WordNet synonym of $\text{lem}_{i,j}$, and 0 otherwise (where "synonym" encompasses different types of WordNet similarities).

Webster cosine similarity:
$\Pr(\text{Slem}_j|\text{lem}_{i,j}) = \text{SimScore}(\text{lem}_{i,j}, \text{Slem}_j)$ if Slem_j is a synonym of $\text{lem}_{i,j}$ according to the Webster thesaurus, and 0 otherwise (where SimScore is obtained by applying the cosine similarity measure, Section 4.1). This approach yields a reduced number of synonyms, which are at the intersection between the lemmas in the Webster thesaurus and those returned by WordNet. This enables us to additionally consider `hypernyms` and `hyponyms` returned by WordNet.[3]

[3] Experiments show that considering `hypernyms` and `hyponyms` under the baseline similarity measure exponentially increases the number of alternative paraphrases without improving retrieval performance.

$\boldsymbol{Pr}(\boldsymbol{lem}_{i,j}|\boldsymbol{ctxt}_{i,j})$ may be represented by $\Pr(\text{lem}_{i,j}|\text{lem}_{i,1},\ldots,\text{lem}_{i,j-1})$. We consider the baseline measure $\Pr(\text{lem}_{i,j}|\text{lem}_{i,1},\ldots,\text{lem}_{i,j-1})=1$, and also the following approximation:

$$\Pr(\text{lem}_{i,j}|\text{lem}_{i,1},\ldots,\text{lem}_{i,j-1}) \simeq \prod_{k=1}^{j-1} \Pr(\text{lem}_{i,k},\text{lem}_{i,j}) \quad (8)$$

where $\Pr(\text{lem}_{i,k},\text{lem}_{i,j})$ is obtained directly from the lemma-pair dictionary (Section 4.1). This approximation, although *ad hoc*, works well in practice, yielding a better performance than bi-gram approximations (Zukerman and Raskutti, 2002).

5 Retrieval Procedures

In this section, we describe the retrieval techniques used for our two applications, and present probabilistic formulations that incorporate sentence paraphrasing into our retrieval procedures.

5.1 Document retrieval

The retrieval procedure for the `DocRet` application relies on the vector-space model, which calculates the score of candidate documents given a list of terms in a query (Salton and McGill, 1983). Normally, this score is based on the TF.IDF measure (Term Frequency . Inverse Document Frequency). If we replace a query with its paraphrase, this score is represented by the following formula:

$$\textit{Score}(\text{Doc}|\text{Para}) = \sum_{j=1}^{L} \textit{tfidf}(\text{Doc},\text{lem}_j) \quad (9)$$

where L is the number of lemmas in the paraphrase.

By normalizing the scores of the documents, we obtain the probability that a document contains the answer to the paraphrase:

$$\Pr(\text{Doc}|\text{Para}) \propto \sum_{j=1}^{L} \textit{tfidf}(\text{Doc},\text{lem}_j) \quad (10)$$

We assume that given a paraphrase of a query, a document retrieved on the basis of the paraphrase is conditionally independent of the original query. This yields the following formula:

$$\Pr(\text{Doc}|\text{Query}) = \sum_{i=0}^{n} \Pr(\text{Doc}|\text{Para}_i) \times \Pr(\text{Para}_i|\text{Query}) \quad (11)$$

where n is the number of paraphrases. We also adopt the convention that the 0-th paraphrase is the original lemmatized query.

By substituting Eqn. 10 and Eqn. 7 for the first and second factors in Eqn. 11 we can calculate the probability of each document given a query.

5.2 Node identification

In order to find the node in the BN that best matches a user's sentence, we calculate for each node a score that reflects its similarity to the user's sentence. This score is composed of two elements: *SP-score* and *NP-score*. *SP-score* is obtained by paraphrasing the user's sentence, and calculating a similarity score between each paraphrase and the canonical sentence(s) for the node in question. *NP-score* is obtained by paraphrasing the canonical sentence(s) for the node, and calculating a similarity score between each paraphrase and the user's sentence. These elements are represented by the following formulas:

$$\textit{SP-score}(\text{Node}|\text{Sent}) = \textit{func}_{i=0,n}\{\text{SimScore}(\text{Para}_i(\text{Sent}),\text{Node})) \times \Pr(\text{Para}_i(\text{Sent})|\text{Sent})\} \quad (12)$$

$$\textit{NP-score}(\text{Node}|\text{Sent}) = \textit{func}_{i=0,n}\{\text{SimScore}(\text{Sent},\text{Para}_i(\text{Node})) \times \Pr(\text{Para}_i(\text{Node})|\text{Node})\} \quad (13)$$

where SimScore is calculated using the cosine similarity measure (which was also used to build the Webster thesaurus and the nominal and verb lists, Section 4.1); $\Pr(\text{Para}_i(\textit{item})|\textit{item})$ is estimated using Eqn. 7; n is the number of paraphrases generated for sentences and nodes; and *func* is either maximum or average. When *func*=maximum, the *SP-score* of a node is the score of the paraphrase of the user's sentence that best matches the node's canonical sentence, and the *NP-score* is the score of the paraphrase of the node which best matches the user's sentence. When *func*=average, the *SP-score* of a node is the weighted average of the scores of the paraphrases of the user's sentence, and the *NP-score* is the weighted average of the scores of the paraphrases of the node.

The probability that a node is intended by a user's sentence is the normalized value of

$$\Pr(\text{Node}|\text{Sent}) \propto \textit{func}\{\textit{SP-score}(\text{Node}|\text{Sent}),\textit{NP-score}(\text{Node}|\text{Sent})\} \quad (14)$$

where *func*=maximum retains the best of *SP-score* and *NP-score*, and *func*=average simply averages these scores.

We also conducted experiments using *SP-score* or *NP-score* alone, but the best results were obtained by combining these scores. In the future, we intend to experiment with a measure that compares the paraphrases of users' sentences against the paraphrases of nodes, instead of using *SP-* and *NP-scores*.

6 Evaluation

In this section we describe our evaluation metric and data sets. We then discuss our experiments, and analyze our results.

6.1 Evaluation metric

We use the *number of matches* measure to evaluate retrieval performance. In the `DocRet` application, this measure returns the number of queries for which the system retrieved at least one document that contains the answer to a query. In the `NodeID` application, this measure returns the number of sentences for which the intended node was among the returned nodes. The following example illustrates why this measure was chosen instead of the standard precision measure. Consider a situation in the `DocRet` application where 10 correct documents are retrieved for each of 2 queries and 0 correct documents for each of 3 queries, compared to a situation where 2 correct documents are retrieved for each of 5 queries. Average precision would yield a better score for the first situation, failing to address the question of interest for this application, namely how many queries have a chance of being answered, which is 2 in the first case and 5 in the second case.

6.2 Data

The data for the `DocRet` application consisted of 125 queries from the TREC8 collection, 404 queries from TREC9, and 231 queries from TREC10. These are queries whose answers reside in the LA Times portion of the TREC corpus.

The data for the `NodeID` application was generated by asking high-school and university students to modify the canonical sentences corresponding to 7 of the nodes in our BN. No constraints were placed on the allowed modifications, i.e., students were advised that they could make any changes to the sentences, provided their meaning was preserved. We received a total of 258 modified sentences from 21 students (an average of 37 sentences per node).

Table 1: Configurations of paraphrasing parameters

	P1	P2	P3
WN-CTXT	WordNet	baseline	`ctxt`
WNWEB-SIM	WordNet+Webster	`sim`	baseline
WNWEB-CTXT	WordNet+Webster	baseline	`ctxt`
WNWEB-SIMCTXT	WordNet+Webster	`sim`	`ctxt`

6.3 Experiments

Our evaluation determines the effect of paraphrasing on retrieval performance, as well as the number of paraphrases that yields the best performance. For both applications, we submitted to the retrieval engine increasing sets of paraphrases as follows: first the lemmatized query or sentence alone (Set 0), next we added to the query or sentence up to 2 paraphrases (Set 2), then up to 5 paraphrases (Set 5), up to 12 paraphrases (Set 12), and up to a maximum of 19 paraphrases (Set 19).[4]

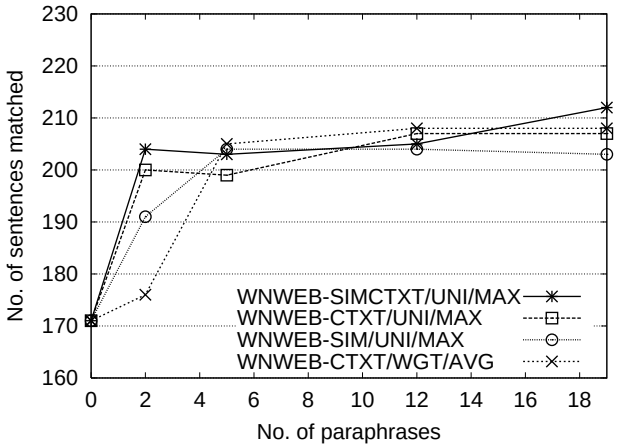
For the `DocRet` application, the number of retrieved documents was kept constant at 200, as suggested in (Moldovan et al., 2002). Since for the `NodeID` application there is only one correct node, it is critical to return this node most of the time. Hence, we considered 1 or 3 retrieved nodes.

We experimented with different combinations of the operating parameters of the paraphrasing process as follows: (P1) WordNet alone or WordNet+Webster; (P2) $\Pr(\text{Slem}_j|\text{lem}_{i,j})$ – baseline measure or Webster similarity score (`sim`); and (P3) $\Pr(\text{lem}_{i,j}|\text{ctxt}_{i,j})$ – baseline measure or the approximation in Eqn. 8 (`ctxt`). The combination of these parameters yielded the four configurations in Table 1 (the remaining options are not viable).

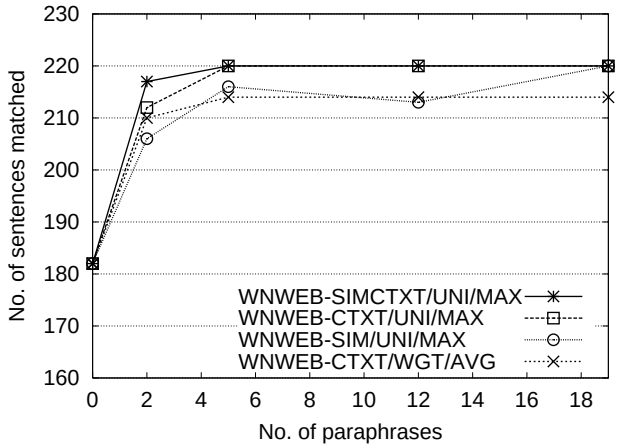
In addition, we considered the following options for the retrieval process: (R1) Weighting – uniform or weighted; and (R2) *func* – maximum or average (only for the `NodeID` application, Section 5.2).[5] The uniform policy in R1 assigns the same weight (=1)

[4] These numbers represent the *maximum* number of paraphrases – fewer paraphrases were generated if there weren't enough synonyms. Also, previous experiments show that Sets 0, 2, 5, 12 and 19 are significant in terms of retrieval performance, and that there seems to be no advantage in generating more than 19 paraphrases.

[5] *func*=maximum is not suitable for the `DocRet` application, as a `DocRet` document does not contain different versions of one sentence, as is the case for the `NodeID` application.



(a) 1 retrieved node



(b) 3 retrieved nodes

Figure 1: Effect of number of paraphrases on performance for the `NodeID` application (258 sentences)

to the paraphrases during retrieval, instead of using the weights obtained from the paraphrasing process (weighted policy). The rationale for the uniform policy is that while a sorting criterion is required to rank the paraphrases, once a ranking is obtained, the top-K paraphrases may be equally useful.

The policies in R1 yielded four additional configurations, one for each entry in Table 1. These configurations were used to generate and rank the paraphrases, but the top-K paraphrases were weighed uniformly during retrieval (we distinguish the weighted policy from the uniform policy by appending the suffix WGT to the weighed options and the suffix UNI to the uniform options). Finally, the policies in R2 yielded another eight configurations for the `NodeID` application only (each of the above options with *func*=maximum or *func*=average). These configurations are denoted by appending the suffix MAX or AVG to the above designations, e.g., WNWEB-SIMCTXT/UNI/AVG.

6.4 Results

Paraphrasing improved retrieval performance under the vast majority of the paraphrasing and retrieval options for both applications. Performance generally improved as the number of paraphrases increased, with the largest improvement being obtained for 2-5 paraphrases. However, it is worth noting that when retrieval was performed using the uniform R1 policy and *func*=average, the performance was sometimes unstable. That is, the number of matched sentences or answerable queries sometimes

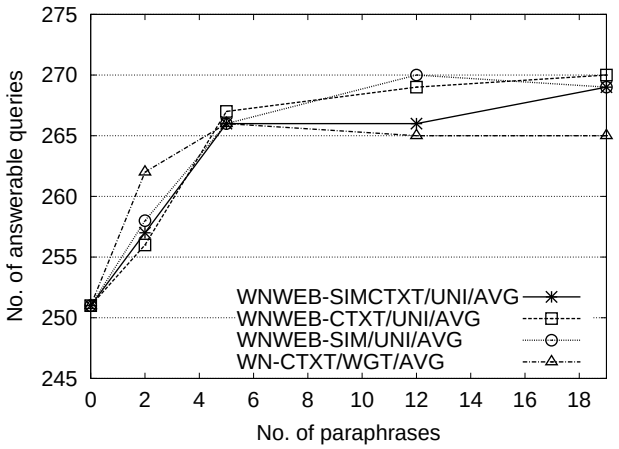


Figure 2: Effect of number of paraphrases on performance for the `DocRet` application (TREC9, 404 queries, 200 retrieved documents)

fluctuated as the number of paraphrases increased. This may be explained by the observation that low probability paraphrases (i.e., paraphrases with a low Webster similarity score to the original sentence, or with a low context-probability) have the same contribution to the score of a node or a retrieved document as high-probability paraphrases.

Figures 1(a), 1(b) and 2 depict the performance of the four best configurations among those discussed in Section 6.3. Figures 1(a) and 1(b) show the performance obtained for the `NodeID` application for 1 retrieved node and 3 retrieved nodes respectively. The retrieval performance obtained for TREC9 queries in the `DocRet` application is

shown in Figure 2 (paraphrasing also improves performance for TREC8 and TREC10, but the improvements are more modest than for TREC9). As can be seen from these Figures, paraphrasing has a greater impact on the `NodeID` application than on the `DocRet` application. Specifically, the best configuration for `NodeID` (WNWEB-SIMCTXT/UNI/MAX with 19 paraphrases) improved the number of matched sentences from 65.9% to 82.2% for 1 retrieved node, and from 70.5% to 85.3% for 3 retrieved nodes. In contrast, the best configuration for `DocRet` (WNWEB-CTXT/UNI/AVG with 19 paraphrases) improved the number of answerable queries from 62.1% to 66.8%.

Finally, note that the top-4 configurations for our two applications differ (but with respect to one aspect only). This indicates that while paraphrasing generally improves retrieval performance, the characteristics of the application determine both the impact of paraphrasing and the paraphrasing and retrieval configuration that works best.

7 Conclusion

We have investigated the effect of lexical paraphrasing on two applications: document retrieval and node identification. Paraphrasing was performed using syntactic, semantic and statistical information. Our results show that paraphrasing improves retrieval performance. However, the improvement is sensitive to the application at hand and to the paraphrasing and retrieval operating parameters.

Acknowledgments

This research was supported in part by grants A49927212 and DP0209565 from the Australian Research Council.

References

Eric Brill. 1992. A simple rule-based part of speech tagger. In *ANLP-92 – Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy.

Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan Cigarran. 1998. Indexing with WordNet synsets can improve text retrieval. In *Proceedings of the COLING-ACL'98 Workshop on Usage of WordNet in Natural Language Processing Systems*, pages 38–44, Montreal, Canada.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2001. The role of lexico-semantic feedback in open domain textual question-answering. In *ACL01 – Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 274–281, Toulouse, France.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *COLING-ACL'98 – Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics*, pages 704–710, Montreal, Canada.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-ACL'98 – Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics*, pages 768–774, Montreal, Canada.

Steven Lytinen, Noriko Tomuro, and Tom Repede. 2000. The use of WordNet sense tagging in FAQfinder. In *Proceedings of the AAAI00 Workshop on AI and Web Search*, Austin, Texas.

Rada Mihalcea and Dan Moldovan. 1999. A method for word sense disambiguation of unrestricted text. In *ACL99 – Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland.

George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244.

Dan Moldovan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu. 2002. Performance issues and error analysis in an open domain question answering system. In *ACL02 – Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Philadelphia, Pennsylvania.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, San Mateo, California.

G. Salton and M.J. McGill. 1983. *An Introduction to Modern Information Retrieval*. McGraw Hill.

Hinrich Schütze and Jan O. Pedersen. 1995. Information retrieval based on word senses. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, Nevada.

Ingrid Zukerman and Sarah George. 2002. Towards a noise-tolerant, representation-independent mechanism for argument interpretation. In *COLING 2002 – Proceedings of the 19th International Conference on Computational Linguistics*, pages 1170–1176, Taipei, Taiwan.

Ingrid Zukerman and Bhavani Raskutti. 2002. Lexical query paraphrasing for document retrieval. In *COLING'02 – Proceedings of the 19th International Conference on Computational Linguistics*, pages 1177–1183, Taipei, Taiwan.