Using Template-Grammars for Shake & Bake Paraphrasing

Michael Carl, Ecaterina Rascu and Paul Schmidt

Institut für Angewandte Informationsforschung
Saarbrücken, Germany
email:{carl,kati,paul}@iai.uni-sb.de

Abstract

In this paper we propose an approach to corpus-based generation in a machine translation framework that is similar to shake & bake (Whitelock, 1992). A bag of words is mapped against an automatically induced TL template grammar and a sentence is generated by recursively applying rules that are extracted from the template grammar. A test version of the template grammar is enriched with further lexical and grammatical variation patterns. We show how we induce a template grammar and how it is enriched with additional paraphrasing knowledge. We suggest a framework for weighing and training the template grammars and show that the enriched template grammars produce better paraphrases.

1. Introduction

Paraphrasing is frequently used as an instrument to improve the output of various NLP applications. On the one hand, it is used to cope with the highly variable character of natural language in applications like multi-document summarisation (Barzilay et al., 1999) where phrases reporting on the same fact have to be found in input documents. Moreover, in controlled language machine translation (Mitamura and Nyberg, 2001) replacing a source language (SL) item with a paraphrase that is better suited to the requirements of the MT engine boosts the performance of the system, whereas in information retrieval or question answering (Jacquemin et al., 1997; Rinaldi et al., 2003) query expansion by means of paraphrases leads to increased retrieval efficiency. On the other hand, paraphrasing is used to control the generation process within NLP applications. It may help produce the best-suited utterance in a given situation (Iordanskaja et al., 1991; Robin, 1994; Dras, 1999).

In this paper we describe an approach to corpus-based generation within an MT framework in which paraphrasing is used as a verifying instrument and a means to improve the performance of the system.

The approach developed in this paper is a contribution to METIS-II. In the EU-project METIS-II, a follow-up to METIS-I[1] (Dologlou et al., 2003), the aim is to investigate the possibilities to develop a data-driven MT system using a huge monolingual target language (TL) corpus and a bilingual dictionary. While the dictionary is used to map SL items onto the TL, the corpus serves as a model to generate the TL sentences. This parallels with shake & bake (S&B) (Whitelock, 1992). In S&B the bilingual knowledge is exhausted by the equivalence of basic expressions and TL generation is under direct control of the TL grammar. This makes large scale structural reorganisation of the TL possible and overcomes the inherent problems in conventional third-generation transfer-based MT. In these latter systems, TL generation is merely a matter of traversing and printing out intermediate representations which are defined by the structure of the SL text (Whitelock, 1991).

The S&B approach also entails that the transfer component and the generation component can be developed and tested independently.

In this paper we focus on the generation component of the system. Coherent (possibly discontinuous) sequences of words are coded in one template. We (re) generate the sentences by using template grammars and paraphrase grammars as well as a weighing procedure to rank the produced paraphrases. We compare the paraphrases that are produced with the different grammars.

In the following section we outline our approach to TL generation. Then, in section 3., we show how a template-based generation grammar is induced from a large English corpus and how a paraphrase grammar is generated by using lexical and grammatical variation patterns. In section 4. we describe the paraphrase generation algorithm and how weights are assigned and trained. In section 5. we give an evaluation of the approach and show that even a small paraphrase grammar outperforms a template grammar that has a 100 times more rules. Last we outline future work on this approach.

[1] http://www.ilsp.gr/metis2/

2. Approach

Shake & bake generation starts from a bag of TL items, where the order of the items in the bag is irrelevant (Whitelock, 1992). Generation freely combines the items to produce all sentences that are compatible with the constraints in the bag and in the TL grammar. According to Brew(Brew, 1992), all items in the bag are to be used in the generated target sentence.

While this paper investigates whether the S&B approach can be combined with template grammars, our approach deviates from standard S&B in that:

1. a paraphrase may contain additional items which are not contained in the bag.
2. a paraphrase may be generated that does not contain all the items in the bag.
3. SL word order is marked and therefore accessible in the bag of items. This information can be considered as a preference mechanism and help score solutions, if there are more than one.
4. Grouping information may be carried over from the SL to express grouping preferences in the TL.
5. Additional constraints may be used to guide the generation process.

This paper would not consider items 3, 4 and 5. See section 6. for a discussion on future investigation.

Since one of the aims of METIS-II is to exploit the knowledge inferable from corpora to a maximum degree, we automatically induce a TL template-grammar from a huge monolingual corpus. The templates contain different degrees of generalisations as outlined in section 3.1.. In this way we induce many more generation rules than could ever be produced by hand. As a second step of generalisation, we generate a paraphrase grammar from the template grammar by enriching the latter with information concerning lexical and grammatical variation as described in section 3.2..

On the basis of the induced template grammar and paraphrase grammar, TL sentences are produced from the initial bag of words and scored according to the length and appropriateness of the rules and templates involved in their generation. We use learning methods to train weights of the rules in order to assign higher scores to better paraphrases.

In contrast to S&B, where the free combination of items in the bag is restricted by constraints of a hand-made TL grammar, in our approach most of the constraints, in particular word-order, are left implicit in the context of the templates.

3. Generation of Shake & Bake Grammar

The METIS-II project uses the British National Corpus (BNC) as a target language corpus. The BNC[2] is a collection of 4,054 annotated spoken and written English texts tagged with the CLAWS tagger[3].

We have extracted from the BNC two sets of sequences[4]:

- set_1 with 1,000 sequences
- set_2 with 100,000 sequences

All sequences contain at least one finite verb and are between 5 and 15 words long. The sets have 8,555 and 863,245 words respectively.

We have partially parsed the sequences and extracted a CFG-like template grammar. The template grammars for set_1 and set_2 are stored in grammars G_1 and G_2 respectively. In addition, we have generated the paraphrase grammars G_{1P} and G_{2P} from G_1 and G_2. In the remainder of this section we outline how we have parsed the sets, extracted the four grammars and generated the paraphrase grammars.

3.1. Inducing a Template Grammar

Partial parsing yields a bracketed structure, as shown in the following example. The pronoun "i", the adverb "never" and the NP "the embarrassment" are bracketed.

$$(\text{i})_{\texttt{pron}}\ \text{'ll}\ (\text{never})_{\texttt{adv}}\ \text{forget}\ (\text{the}\ (\text{embarrassment})_{\texttt{noun}})_{\texttt{np}}$$

We do not allow overlapping and/or ambiguous segmentation but enable recursive bracketing. Thus, a "noun" can be bracketed within a larger "np" which can be part of a "pp" etc. For instance, the bracketed noun "embarrassment" is contained in the larger "np" (the (embarrassment)$_{\texttt{noun}}$)$_{\texttt{np}}$.

```
1  seg   →  <pron> 'll <*adv> forget <np> .
2  np    →  the <noun>
3  pron  →  i
4  noun  →  embarrassment
5  adv   →  never
```

We extract a CFG grammar from these trees in the following way. On the one hand, we extract rules from the bracketed structures by transforming the

[2] http://www.natcorp.ox.ac.uk

[3] http://www.comp.lancs.ac.uk/computing/research/ucrel/claws/

[4] The BNC manual refers to "sequence" rather than sentences: The sequence "is the basic organisational principle for the whole corpus ..." In many cases a sequence corresponds with regular orthographic sentences.

G_1		G_2		G_{1P}		G_{2P}	
1837	seg	158211	seg	2714	seg	243149	seg
1775	noun	34957	noun	1782	noun	34968	noun
508	adj	9640	adj	515	adj	11468	sentence
128	adv	7373	sentence	157	sentence	9651	adj
115	card	3669	card	139	adv	3669	card
99	sentence	1290	adv	116	card	1299	adv
95	np	358	np	98	np	358	np
54	pp	215	pp	55	pp	215	pp
5	clause	26	clause	5	clause	26	clause
4616		215739		5581		304803	

Table 1: Distribution of Rules in Template Grammars G_1 and G_2 and in Paraphrase Grammars G_{1P} and G_{2P}

tag into the left-hand side (LHS) of the rules and the contents into the right-hand side RHS. Thus the tag noun appears on the LHS in rule (4) while the contents of the bracketed expression "embarrassment" occurs in the RHS. On the other hand, templates are generated by replacing the bracketed constituents with their tags. A tag can be marked optional with an asterisk $*$. For instance, the adverb "never" is marked as an optional slot in template (1) carrying the feature <*adv> while the <np> object is a compulsory slot in the same template.

The grammar extracted from the above example consists of 5 context-free rules, where seg is the top-level symbol and np, noun and adv are non-terminals. Note that at least one terminal symbol must occur in the RHS of the rules.

The partial parser is implemented in KURD (Carl and Schmidt-Wigger, 1998). The parser consists of three sets of rules which incrementally produce larger brackets: the LEX set marks only the lexical items: nouns, adjectives, adverbs and numbers. The PHRASE set marks adjective phrases, noun phrases, conjunctions of noun phrases and prepositional phrases. The CLAUSE set marks subordinate clauses and sentences.

A separate template is generated for each set of rules. Thus, sequence (6) can be generalised in three different ways: template (7) is generated with the LEX set by substituting only lexemes. Template (8) is generated with the PHRASE set by substituting phrases whereas template (9) contains a substituted clause. In addition to the generated templates of the sequences, constituents and their generalisations are also extracted.

6 seg → we 'll see if we can get you to rights .
7 seg → <pron> 'll see if <pron> can get <pron> to <noun> .
8 seg → <pron> 'll see if <pron> can get <pron> <pp> .
9 seg → <pron> 'll see <clause> .

Table 1 shows size and contents of the grammars G_1 and G_2 generated from set$_1$ and set$_2$ respectively. The number of rules in G_1 is more than 4 times higher than the number of sequences in set$_1$ while for G_2 on average slightly more than two rules are generated for each sequence. For each sequence in the sets, 1.8 respectively 1.5 sequence templates were generated on average.

3.2. Inducing a Paraphrase Grammar

We enrich the template grammars G_1 and G_2 with synonyms, variants and paraphrases using a methodology described in (Carl et al., 2004). Similar to the approach in (Rinaldi et al., 2003), we operate on two levels: on the phrase level, lexical paraphrases are generated while on the sentence level, paraphrases based on grammatical variation are produced. This paraphrasing process relies on two types of manually collected resources: lexical variation patterns including writing variants and synonyms of words and phrases on the one hand and grammatical variation patterns on the other.

The following examples illustrate the paraphrasing process for sequence (10). The template grammar (as described in section 3.1.) produces template (11). This template is the input to the paraphrasing process.

10 seg → it wo n't dry till next week.
11 seg → <pron> wo n't dry till next <noun>.
12 seg → <pron> will not dry until the following <noun>.
13 seg → <pron> is n't going to dry till next <noun>.
14 seg → <pron> is;am;are not going to dry until the following <noun>.

The paraphrase grammar operates in several steps, adding new templates to the template grammar. In a first step only the lexical variation patterns are used to generate paraphrase templates (12). Then, the grammatical variation patterns are applied to template (11) producing paraphrase template (13) whereas paraphrase template (14) is produced by combining the

lexical and grammatical variation patterns.[5]

The figures for the generated paraphrase grammars are given in table 1. Compared to the template grammars, the paraphrase grammars G_{1P} and G_{2P} contain on average one more rule for each initial sequence. Thus, grammar G_{1P} is more than five times larger than set$_1$ while grammar G_{2P} contains slightly more than three times the number of sequences in set$_2$.

4. Shake & Bake Paraphrasing

In this section we show how we match a bag of words on a template grammar and how paraphrases are generated.

4.1. Paraphrase Generation

We produce a set of paraphrases $p_0 \dots p_{n-1}$ from an English sentence s using the generation grammars G_1 G_2 G_{1P} and G_{2P} as outlined in section 3.. First we transform the English sentence s into a bag of words B enriched with further constraints. Each subset of B is matched on the grammar and the retrieved rules are assembled in a generation grammar G_s for s as outlined in section 4.2..

Generation of paraphrases from G_s works in a top-down manner starting from rules with the top-level tag `seg` recursively filling the variable slots, thereby building a derivation for a particular paraphrase p_i. When filling the slots of templates, the slot's tag must unify (actually be identical) with the LHS of the filling rule. In order to avoid non-terminating recursion, we only allow rules with at least one terminal symbol on the RHS.

We describe how the paraphrases are weighted in section 4.3.. Weights are computed while the paraphrases are generated. In section 4.4. we propose an algorithm to train the weights. We evaluate the approach based on the similarity of paraphrases p_i and the original sentence s in section 5..

4.2. Retrieving Rules from the Template Grammar

To find most suitable rules in the template grammar, two weights are computed $rw(r_j|B)$ and $ew(B|r_j)$. The weight $rw(r_j|B)$ indicates whether the rule r_j contains additional items that are not in B. The weight $ew(B|r_j)$ indicates to which extent the rule r_j covers the items in B

[5]In the present architecture surface forms of words are used rather than lemmas. In future versions we will consider lemmas instead of the surface forms.

Thus, we seek two functions that have the following properties:

$rw(r_j|B) = 1$ if r_j is a subset of B.
$rw(r_j|B) < 1$ if r_j contains items that are not in B.
$ew(B|r_j) = 1$ if B is a subset of r_j.
$ew(B|r_j) < 1$ if B contains items that are not in r_j.

We grade the retrieved rules by the product of the two functions:

$$rew(r_j, B) = rw(r_j|B) * ew(B|r_j)$$

The rules in G_s are selected according to the weight $rew(.)$[6]. The generation grammar G_s contains the first n-best (e.g. n = 20) rules for each subset of B.

Since not every item in B and r_j is equally important, we weigh the items by their inverse logarithmic frequencies. That is, an item that occurs very frequently in the grammar, as e.g. articles, will contribute less to the overall weights than items that occurs only few times. In this way we hope to assign higher weighs to rules that realise to a better extent the contents of the bag.

$$rw(r_j|B) = \sum_{t \in r_j \cap B} 1/[cnt(r_j) * log(f(t))]$$
$$ew(B|r_j) = \sum_{t \in r_j \cap B} 1/[cnt(B) * log(f(t))]$$

where $log(f(t))$ is the logarithm of the frequencies a token t occurs in the grammar and the count cnt is the inverse sum over all token log frequencies in X:

$$cnt(X) = 1/\sum_{t \in X} log(f(t))$$

Thus, the bag B is mapped onto a grammar from which a set of rules is retrieved. The set of rules is considered a generation grammar G_s for s. From this generation grammar, a number of paraphrases $\{p_0 ... p_{n-1}\}$ are produced as outlined above. The next sections describe how paraphrases are weighted.

4.3. Weighing Paraphrases

In order to grade the produced paraphrases, we compute a weight which is composed of the weights of the involved rules and the length of the produced paraphrases. The basic idea is that paraphrases composed of one (or few) piece(s) (i.e. few specific rules) are likely to be better than paraphrases composed of

[6]$rew(.) = 1$ if r_j and B are identical, else $rew(.) < 1$.

	G_1	G_{1P}	G_2	G_{2P}
$md(S,t_0)$	0.195349	0.171970	0.205970	0.185604
$md(S,t_1)$	0.151062	0.132748	0.173205	0.178096
$md(S,t_2)$	0.122849	0.109684	0.159651	0.164168
$md(S,t_3)$	0.103013	0.094421	0.141437	0.156727
$md(S,t_4)$	0.088030	0.083225	0.138416	0.141413
$md(S,t_5)$	0.076157	0.074405	0.113293	0.126913
$md(S,t_6)$	0.066480	0.067166	0.119586	0.119260
i will never love him .	i will never love him .	i will never love him .	i will never love him .	
will , love never .	i never love him .	will i my love gain never .	i will never understand him .	
i and him never .	i will never see him .	him will never love i .	i shall never love him .	
i , him never .	i shall never love him .	i will love him .	i 'll never love him .	
i will tell him .	i and him never .	fi rst i will gain him .	oh i will never call him .	

Table 2: Convergence of the the template and paraphrase grammars on a set of 15 test sequences and some of the best produced paraphrases after 6 iterations

many small pieces (i.e. many rules). Accordingly, weights assigned to paraphrases decrease the more they are built from small pieces. In addition, a paraphrase should be as complete as possible. That is, compared to the bag of items from which it is generated, the paraphrase should contain

We initialise the weight of a rule r_j at time t_0 based on the number of its terminal symbols s_j and its non-terminal symbols n_j:

$$w(r_j, t_0) = s_j/(s_j + n_j)$$

The initial weight $w(r_j, t_0)$ is equal to 1 if rule r_j contains only terminal symbols. The more non-terminal symbols r_j contains, the smaller will be its weight.

Since rules map a certain number of items in B, we contextualise the weight of r_j according to the amount of items it covers in B. The weight $w(r_j, B, t_k)$ of a rule r_j given a bag of items B at time t_k is:

$$w(r_j, B, t_k) = w(r_j, t_k) * s_j/|B|$$

The weight $w(r_j, B, t_0)$ is equal to 1 if r_j and B share the same tokens and if r_j contains only terminal symbols. The weight $w(r_j, B, t_0)$ decreases if r_j contains non-terminal symbols and/or if r_j covers a subset of B.

Finally, we compute the weight $w(p_i, B, t_k)$ of an English paraphrase p_i as the sum of the contextualised weights for the rules r_j which are used to build p_i:

$$w(p_i, B, t_k) = \sum_{r_j \mapsto p_i} w(r_j, B, t_k)$$

which is equivalent to:

$$w(p_i, B, t_k) = 1/|B| * \sum_{r_j \mapsto p_i} w(r_j, t_k) * s_j$$

Weights are computed compositionally and recursively as the derivation tree of a particular paraphrase is built up. Since these two processes are processed in parallel, the weight $w(r_j, B, t_k)$ gives us the possibility to grade and order the next rule to apply in every step of the generation process. The rules and constraints, on the other hand, tell how the pieces should be stitched together. This makes it possible to generate n-best paraphrases or stop below a certain threshold or beyond a fixed number of generated sentences.

4.4. Training Weights

We re-estimate the weights using an "objective score". The objective score evaluates each paraphrase $p_0 \dots p_{n-1}$ according to some external criteria. The goal of the algorithm is to adjust the rule weights $w(r_j, t_k)$ so that the mean error between the objective score and the internal weight $w(p_i, B, t_k)$ is minimised.

We use BLEU as an objective score to compare the produced paraphrases $p_0 \dots p_{n-1}$ with the English source string s. We compute for each paraphrase p_i the delta $D = bleu(p_i, s) - w(p_i, B_s, t_k)$. The weights of the rules r_j used to generate p_i are adjusted at time t_{k+1} according to the following equation:

$$w(r_j, t_{k+1}) = w(r_j, t_k) + d(r_j, t_k)/n_j$$

where $d(r_j, t_{k+1})$ and n_j are computed as in figure 3.

for all (sequences $s_p \in$ test set)
  generate a bag of items B_p from s_p
  extract generation grammar G_p from G
  produce paraphrases $P_p : \{p_0 ... p_{n-1}\}$ from G_p
  for all (paraphrases $p_i \in P_p$)
    $D_p = bleu(p_i, s_p) - w(p_i, B_p, t_k)$
    for all (rules r_j used to generate p_i)
      $d(r_j, t_{k+1}) += D_p$
      n_j++;
    end
  end
end

Table 3: Re-estimating rule weights

5. Evaluation

We have trained the algorithm on a test set (S) of 15 sequences containing no out of vocabulary words. The sequences in S were between 5 and 12 words long. For each sequence approximately 100 paraphrases were generated. Each paraphrase was assigned an internal score $w(p_i, B_p, t_k)$ as discussed in section 4.3. The BLEU score was computed for each paraphrase p_i measuring the 'distance' to the original sentence s_p. The numerical difference between the internal score and the BLEU score was fed back into the rules of the grammars as discussed in section 4.4..

Table 2 shows the mean arithmetic distance (md) between the internal and the external weight for the test set at the successive iteration steps t_0 to t_6. The mean arithmetic distance was computed as follows:

$$md(S, t_k) = \sqrt{1/|S| * \sum_{s_p \in S} (bleu(p_i, s_p) - w(p_i, B_p, t_k))^2}$$

As shown in table 2, the training algorithm minimises the mean arithmetic distance (md) in successive iteration steps. However, larger grammars converge more slowly. We expect this to be due to the small size of the test set.

It is also interesting to note that the paraphrase grammars (G_{1P} and G_{2P}) start with a better lower error than the template grammars. An example of the produced paraphrases is shown in the lower part of the table. This small test seems to indicate that paraphrase grammars are better suited for S&B paraphrasing than the template grammars.

6. Future Work

In the future we intend to extend the approach in several directions.

The reference material as well as the items in the bags are to be lemmatized. In order for the lemmatizer to be useful for generation, it has to come along with a reversible token-generator. That is, not only it must be possible to re-generate the original word-forms but the token-generator also has to produce related forms. For instance, a noun might occur in its singular in a template as it was extracted from the BNC. When re-using the template in a different context it might be necessary to re-generate the word's plural form.

In conjunction with a reversible lemmatizer/token-generator, a morphological generator becomes indispensable. While the token-generator produces a particular word-form given a lemma and additional features, the morphological generator figures out which forms are to be produced. Since the lemmatizer abstracts from number, person and tense, the morphological generator must, in principle, make sure that each lemma is associated with the correct information concerning number, person and tense for the token-generator to produce the correct forms. Thus, the morphological generator is responsible for correct agreement, realisation of (personal) pronouns. We are currently investigating the possibilities of these tools and elaborate a list of requirements to be met by the system.

Eventually we want to generate the bag of words not from a TL sentence, but obtain the items from a source language. This requires a bilingual dictionary and an appropriate lexical transfer module. Consistent integration of the mathematics for retrieval of rules (see in section 4.2.), weighing (section 4.3.) and training (4.4.) would have to extend to lexical weights obtained from dictionary lookup. Training of weights will have to feed-back into the dictionary resources.

As mentioned in section 2., additional constraints shall be used to guide the generation process as described in this paper. These constraints may stem from different sources and will have an impact on the rules retrieved from the grammar (cfg. section 4.2.) and how they are stitched together.

These constraints and preferences include:

- collocation preferences of certain word combinations
- chunking or parsing information carried over from the SL
- requirements wrt. type of text and domain
- user preferences wrt. terminology, paraphrasing, style etc.

Our generation architecture can be viewed as a sort of "machine translation without a source text" (Somers et al., 1990). With Somers we assume that some kind of "conventional compositional translation" is required but also additional guiding information (which they call dialogue context DC in their paper) is necessary. While flexibility and recall of our system is achieved by the compositionality of the template grammar, Somers' vision is that "the equivalence relation of two expressions is not guaranteed by the expressions themselves but by the DCs which are given rather independently of the information content ..."

Against this background we are investigating possibilities to express different types of constraints in a homogeneous format. It should be possible to easily adopted constraints to different needs and they should well integrate into the architecture of the system.

7. Conclusion

The paper presents a first step towards a corpus-based MT system using a huge monolingual target language corpus and a hand-crafted dictionary. The paper presents a generation component from a bag of TL words.

We discuss two approaches to paraphrase generation that are simultaneously used in one system: (a) a 'rule-based' paraphrasing system which extends a template grammar with lexical and grammatical variation patterns and (b) a 'data-driven' shake & bake generation module that generates a number of paraphrases from a bag of words using the original template grammar and the paraphrase-extended template grammar. We show that a small extended template grammar produces better results than a large grammar that is not extended with paraphrases.

The use of two different techniques, i.e. rule-based and data-driven paraphrasing, to achieve one goal is motivated by the nature of the approach adopted for TL generation within the METIS-II MT system. On the one hand the input from which we want to generate the paraphrases is a bag of items and not a syntactically correct text. Therefore special constraints are needed in the TL grammar in order to produce an adequate output. On the other hand, we seek to exploit the knowledge contained in a large text corpus to a maximum extent. Hence, the rule-based paraphrasing method was used as an instrument to extend and refine the resources that are induced from the corpus. We show that adding further rule-induced knowledge to a text corpus increases the performance of paraphrase generation.

8. References

Barzilay, R., K. McKeown, and M. Elhadad, 1999. Information Fusion in the Context of Multi-Document Summarization. In *Proceedings of the ACL*.

Brew, Chris, 1992. Letting the Cat out of the Bag: Generation for Shake & Bake MT. In *Proceedings of COLING92*.

Carl, Michael, Ecaterina Rascu, Johann Haller, and Philippe Langlais, 2004. Abducing Term Variant Translations in Aligned Texts. *Terminology*, 10(1):103–133.

Carl, Michael and Antje Schmidt-Wigger, 1998. Shallow Postmorphological Processing with KURD. In *Proceedings of NeMLaP3/CoNLL98*. Sydney.

Dologlou, Y., S. Markantonatou, G. Tambouratzis, O. Yannoutsou, A. Fourla, and N. Ioannou, 2003. Using Monolingual Corpora for Statistical Machine Translation: The METIS System. In *Proceedings of the EAMT-CLAW 03: Controlled Language Translation*.

Dras, M., 1999. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. Ph.D. thesis, Macquarie University.

Iordanskaja, L., R. Kittredge, and A Polguere, 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers.

Jacquemin, C., J. Klavans, and E. Tzoukerman, 1997. Expansion of Multi-Word Terms for Indexing and Retrieval Using Morphology and Syntax. In *Proceedings of the ACL*.

Mitamura, T. and E. Nyberg, 2001. Automatic Rewriting for Controlled Language Translation. In *Proceedings of the NLPRS 2001 Workshop on Automatic Paraphrasing: Theories and Applications*.

Rinaldi, F., J. Dowdall, D. Molla, K. Kaljurand, and M. Hess, 2003. Exploiting Paraphrases in a Question Answering System. In *Proceedings of the ACL*.

Robin, J., 1994. *Revision-based Generation of Natural Language Summaries Providing Historical Background*. Ph.D. thesis, New York University.

Somers, Harold, Jun ichi Tsujii, and Danny Jones, 1990. Machine translation without a source text. In *Proceedings of the COLING90*.

Whitelock, P., 1991. *Shake-and-Bake Translation*. Unpublished Draft.

Whitelock, P., 1992. Shake-and-Bake Translation. In *Proceedings of the COLING92*.