# Reversible Template-based Shake & Bake Generation

**Michel Carl, Paul Schmidt and Jörg Schütz**
Institut für Angewandte Informationsforschung
66111 Saarbrücken
Germany
{carl,paul,joerg}@iai.uni-sb.de

## Abstract

Corpus-based MT systems that analyse and generalise texts beyond the surface forms of words require generation tools to re-generate the various internal representations into valid target language (TL) sentences. While the generation of word-forms from lemmas is probably the last step in every text generation process at its very bottom end, token-generation cannot be accomplished without structural and morpho-syntactic knowledge of the sentence to be generated. As in many other MT models, this knowledge is composed of a target language model and a bag of information transferred from the source language.

In this paper we establish an abstracted, linguistically informed, target language model. We use a tagger, a lemmatiser and a parser to infer a template grammar from the TL corpus. Given a linguistically informed TL model, the aim is to see what need be provided from the transfer module for generation.

During computation of the template grammar, we simultaneously build up for each TL sentence the content of the bag such that the sentence can be deterministically reproduced. In this way we control the completeness of the approach and will have an idea of what pieces of information we need to code in the TL bag.

## 1 Introduction

METIS-II[1] investigates 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 translation strategy parallels with shake & bake (S&B)(Whitelock, 1991; Whitelock, 1992). In S&B the bilingual knowledge is exhausted by the equivalence of basic expressions and TL generation as parsing is under direct control of the TL grammar.

Shake & bake generation starts from a bag of TL items. The order of the items in the bag is irrelevant.

Generation freely combines the items to produce all sentences that are compatible with the constraints in the bag and in the TL grammar. While the content of the bag is obtained from the analysis of the source language (SL) and a dictionary lookup, the main challenge in S&B is the generation of TL sentences from a bag of TL items.

In this paper we investigate a corpus-based approach to S&B generation. In contrast to S&B, where the free combination of items in the bag is restricted by constraints of a hand-made TL grammar (Brew, 1992), we automatically induce a TL grammar from a corpus of TL sentences. The TL grammar serves as a model to select and serialise items in the bag according to the TL syntax.

A similar strategy is also proposed by (Cao and Li, 2000) who translate base noun phrases using a dictionary and the web. As in (Habash and Dorr, 2002) we view machine translation as a 'generation heavy' process. We assume a large number of resources in the TL, first of all a huge corpus of TL sentences, so as to shift most of the processing from SL analysis to the TL generation.

Current language modeling in corpus-based machine translation relies on n-grams (Stolcke, 2002; Goodman, 2002; Badia, 2005). Probabilities of overlapping word n-grams are an excellent means to generate and weight coherent sequences of words. However, long distance dependencies cannot easily be handled with n-gram models. In addition, n-grams are usually obtained from inflected words. In METIS, however, we assume lemmatised words in the TL bags. There is thus a gap between lemmatised forms in the input bag and n-gram models based on full word forms.

We present a corpus-derived language model that overcomes these shortcomings. A corpus of English sentences is tagged, lemmatised and parsed. The parsed structures are converted into a normalised context-free grammar and stored in a database. Due to the shape of the representations we call the resulting database a template grammar. The template grammar is the basis of our language model that contains the basic information required to generate

---

English sentences.

Sentence templates have been studied and used for some time. Templates consist of sequences of constant and variable elements which emerge with the identification of similarities and differences with forms in memory. (Cicekli and Guvenir, 2003) give a formalisation of this process while (Malavazos and Piperidis, 2000) establishes a link between templates and analogical modelling.

Recently (Cicekli, 2005; Carl, 2003) extend translation templates with type constraints. (Gough and Way, 2004) produces a set of marker templates by replacing the marker word by its relevant tag. Similarly, we generalise templates from monolingual sentences by replacing constituents by their relevant tag.

To produce a sentence (or a text) from a template grammar, we need additional information from the TL bag. The items and constraints in the bag select and activate a subset of rules in the grammar which then produces a TL sentence. By mapping the bag on the template grammar, word order is determined and features for morpho-syntactic generation are fixed. Thus, the content of the bag should interact with the template grammar such that information is complete to resolve all major morphological and syntactic ambiguities for generation. In the same time the model should be flexible enough to produce all desirable sentences in the TL.

Obviously, the content of the bags depend on the information available in the template grammar and vice versa. In many MT systems, TL generation is seen a consequence of SL analysis. Thus, almost all (symbolic) approaches to MT start from SL sentences and design TL generation according to the information available after transfer. However, statistical (IBM) approaches have shown that a reversed method is not only possible but also leads to a reasonable decomposition of the translation task: To find the most likely translation SL $\rightarrow$ TL, Bayes' theorem allows to train probabilities TL $\rightarrow$ SL and an (independent) target language model. Thus parameters are trained in the inverted order of the intended translation direction while for translation the reversed model is used.

In this paper we follow this intuition for the generation of a language model from a TL corpus. Simultaneously to the language model, we generate for each sentence a bag of items and constraints that complements the language model such that the original sentences can be reproduced. That is, for each step in the construction of an abstracted TL model, we compute and assemble the bits of information that enable the reproduction of the original sentence. In this way we obtain a template grammar and a set of bags for the English sentences. The bags contain lemmas, structural and morpho-syntactic information such that the original text can be reproduced.

Only if we know how a bag looks like in order to generate a particular sentence with a given (template) grammar, we can try to obtain similar bags as a result of transfer and through a bilingual dictionary from a SL sentence. Further research will show whether and to what extent this is an appropriate basis for S&B translation.

We incrementally build a target language model on four levels:

- First we have trained the TnT tagger(Brants, 2000) with the BNC data to obtain tagged sentences.

- Section 2 describes a reversible lemmatisation/token generation tool that takes as its input the tagged text[2].

- Section 3 describes a number of experiments to generate word forms from lemmas with partial information.

- Section 4 describes reversible parsing and morphological processing

It turns out that constraints are essentially determined by the way we implement parsing and morphological processing. To make the process reversible, the bags need to be extended with additional structural and morpho-syntactic information, while near perfect token generation can be obtained even with restricted information.

## 2 Reversible Lemmatisation

This section describes a reversible lemmatiser/token-generator for English. The lemmatiser produces a normalized form for word-tokens in the following sense:

1. convert the lemma into lower-case alphabetical characters

2. apply rules or a token-lemma dictionary to generate the lemma

Lemmatisation rules are used to strip off or modify regular inflection suffixes from the tokens. A lemmatisation lexicon is used for the irregular cases.

The lemmatiser reads a CLAWS5-tagged[3] file, generates a lemma together with two additional features indicating the orthographic properties (O) and the inflection rule (IR) that applies to the word. The token-generator reads a lemma together with a CLAWS5-tag (henceforth CTAG) and the O and the IR feature. Token generation is to a 100% reversible, that is: a token set {token,CTAG} is equivalent to a lemma set {lemma,CTAG,O,IR} and both sets can

---

[2]The lemmatiser can be obtained from the authors
[3]`http://www.comp.lancs.ac.uk/computing/research/ucrel/claws/`

be transformed without loss of information into each other.

In section 2.1 we give a small introduction to the CLAWS5 tag set. The material is essentially copied from their web-site at `http://www.comp.lancs.ac.uk/computing/research/ucrel/claws/`.

Orthographic normalisation is described in section 2.2. The lemmatiser makes use of a lemmatisation lexicon and lemmatisation rules as described in section 2.3 and 2.4. Section 2.5 explains token-generation under the assumption that all required information is available.

| | |
|---|---|
| NN0 | Common noun, neutral for number (e.g. aircraft, data, committee) |
| NN1 | Singular common noun (e.g. pencil, goose, time, revelation) |
| NN2 | Plural common noun (e.g. pencils, geese, times, revelations) |
| NP0 | Proper noun (e.g. London, Michael, Mars, IBM) |
| VVB | The finite base form of lexical verbs (e.g. forget, send, live, return) [Including the imperative and present subjunctive] |
| VVD | The past tense form of lexical verbs (e.g. forgot, sent, lived, returned) |
| VVG | The -ing form of lexical verbs (e.g. forgetting, sending, living, returning) |
| VVI | The infinitive form of lexical verbs (e.g. forget, send, live, return) |
| VVN | The past participle form of lexical verbs (e.g. forgotten, sent, lived, returned) |
| VVZ | The -s form of lexical verbs (e.g. forgets, sends, lives, returns) |

Table 1: Subset of the CLAWS5 tag set

## 2.1 The CLAWS Tag set

The POS tagging software for English text, CLAWS (the Constituent Likelihood Automatic Word-tagging System), has been continuously developed since the early 1980s (see `http://www.comp.lancs.ac.uk/computing/research/ucrel/claws/`).

Accuracy CLAWS has consistently achieved 96-97% accuracy (the precise degree of accuracy varying according to the type of text). Judged in terms of major categories, the system has an error-rate of only 1.5%, with c.3.3% ambiguities unresolved, within the BNC. The amount of error in the tagging of the corpus varies greatly from one tag to another. The most error prone-tag, by a large margin, is VVB, with more than 17 per cent error, while many of the tags are associated with no errors at all, and well over half the tags have less than a 1 per cent error.

The CLAWS5 tagset for the BNC has just over 60 tags. This tagset was kept small because it was designed for handling much larger quantities of data than were dealt with up to that point. For instance there are four different tags for nouns and six for verbs as shown in table 1.

In addition, there are 30 "Ambiguity Tags". These are applied wherever the probabilities assigned by the CLAWS automatic tagger to its first and second choice tags were considered too low for reliable disambiguation. So, for example, the ambiguity tag AJ0-AV0 indicates that the choice between adjective (AJ0) and adverb (AV0) is left open, although the tagger has a preference for an adjective reading. The mirror tag, AV0-AJ0, again shows adjective-adverb ambiguity, but this time the more likely reading is the adverb.

The term 'multiwords' denotes multiple-word combinations which function as one wordclass - for example, a complex preposition, an adverbial, or a foreign expression naturalised into English as a compound noun.

| | | |
|---|---|---|
| AV0 | of course | (adverb) |
| PRP | according to | (preposition) |
| NN1 | persona non grata | |
| | ('naturalised' compound noun) | |

## 2.2 Orthographic Normalisation

The lemmatiser converts characters into lower case. The O feature keeps track of the orthographic properties of the original word. The O feature has the following values:

**n** the token consists only of digits [0-9]

**s** the token does not contain alphabetical characters

**l** the token consists only of lower-case alphabetical characters, and may contain digits and the special characters -\'

**c** the token consists only of upper-case alphabetical characters, and may contain digits and the special characters -\'

**f** the first character is upper-case and all others are lower-case, digits or the special characters -\'

**m** for all other tokens.

The lemma is identical to the word form for the cases **n**, **s**, and **m**. That is, no explicit lemma conversion takes place if the token is not a proper word. For **l**, **c** and **f**, the lemma is converted into lower-case characters and inflection is checked[4]

---

[4]We allow digits to occur in proper words because some special symbols (e.g. blanks) in (compound) words can be escaped with a backslash (\) followed by their ASCII code.

Lemmatisation Rules

| {CTAG, | IR} | token-suffix | | ⇒ | lemma-suffix |
|--------|-----|--------------|---|---|--------------|
| VVG | 1 | `ffing` | | ⇒ | `ff` |
| VVG | 2 | `^(.{1,3}ll)ing` | | ⇒ | `\$1` |
| VVG | 3 | `ssing` | | ⇒ | `ss` |
| VVG | 4 | `([bcdfghjklmnpqrstvwxz])\1ing` | | ⇒ | `\$1` |

Token-generation Rules

| {CTAG, | IR} | lemma-suffix | ⇒ | token-suffix |
|--------|-----|--------------|---|--------------|
| VVG | 1 | `ff` | ⇒ | `ffing` |
| VVG | 2 | `(.{1,3}ll)` | ⇒ | `$1ing` |
| VVG | 3 | `ss` | ⇒ | `ssing` |
| VVG | 4 | `([bcdfghjklmnpqrstvwxz])` | ⇒ | `$1$1ing` |

Table 3: First four rules of the VVG paradigm: rules are reversed for token-generation

| CTAG | IR | token | lemma |
|------|-----|---------|----------|
| NN2 | L8 | analyses | analysis |
| VVN | L28 | gone | go |
| VVD | L29 | went | go |
| VVZ | L6 | goes | go |
| PNQ | whom | whom | who |
| PNQ | whose | whose | who |
| AJC | 1 | better | good |
| AJS | 17 | best | good |

Table 2: Excerpt from the Lemmatisation Lexicon

## 2.3   The Lemmatisation Lexicon

The lemmatisation lexicon encodes a word token together with a CTAG, its lemma and an IR as shown in table 2. Each {token,CTAG} combination is associated with one {lemma,IR} combination. To ensure reversibility of lemmatisation, the IR must be chosen such that each {lemma,CTAG,IR} is unique. In this way every {token,CTAG} combination is equivalent to exactly one {lemma,CTAG,IR}.

Lexical lemmatisation looks up a {token,CTAG} in the dictionary and retrieves a {lemma,CTAG,IR}.

The IR can encode morpho-syntactic and even semantic information in a systematic way such that it can be used during processing in subsequent processes. For instance, a finer grained distinction can be modeled between "us" and "we" or "whom" and "whose" while both forms can be reduced to the same lemma. Otherwise the IR can also consist of any distinguishing string or number as shown in the case of "best" and "better" in table 2. Lemmatisation should be lexicalised only one of the following conditions apply:

1. the word belongs to a closed class

2. the word is an inflectional exception or irregular form such as "better" and "best"

3. further morphological information is required that can be coded in the IR

## 2.4   Lemmatisation Rules

Lemmatisation rules map a word on its lemma by modifying the suffix of the word. This is particularly important for regular inflection of open class words.

A CTAG represents an inflection paradigm that is covered by a number of lemmatisation rules. For each {token,CTAG} — if it is not lexicalised — the lemmatiser applies a number of lemmatisation rules in a predefined order. If a lemmatisation rule matches the token, it is modified and thereby transformed into a lemma. The lemmatiser returns the lemma together with a CTAG, the O feature and the IR.

For instance, the "VVG" paradigm is associated with a list of 28 lemmatisation rules. The first 4 lemmatisation rules are shown in table 3.

The body of the rules are regular expressions that are mapped on the word tokens. A matching token suffix is substituted be a lemma suffix. Parts of in the token-suffix can be enclosed in brackets, as in rule 2. The variable `$1` in the lemma-suffix will be instantiated with the bracketed sequence of the token-suffix such that sequences are copied from the token-suffix to the lemma-suffix.

## 2.5   Token Generation

The lemmatisation process is reversed for token generation. Reversing the lemmatisation lexicon (see section 2.3) becomes a tokenisation lexicon: For every lemma set {lemma,CTAG,IR} that is found in the lexicon, the associated {tag,CTAG} is returned. Reversing the lemmatisation rules (see section 2.4) becomes token-generation rules: The token-generator looks up the lemmatisation rule indicated by IR in the CTAG paradigm and applies the retrieved lemmatisation rule in the reversed order as shown in table 3

As outlined above, token generation is to a 100% reversible if the lemma set is complete. That is: a token set {token,CTAG} is equivalent to a lemma set

| token | CTAG | ⇔ | lemma | CTAG_O_IR |
|-------|------|---|-------|-----------|
| sniffing | VVG | ⇔ | sniff | VVG_l_1 |
| dialling | VVG | ⇔ | dial | VVG_l_2 |
| DRESSING | VVG | ⇔ | dress | VVG_c_3 |
| Setting | VVG | ⇔ | set | VVG_f_4 |

Table 4: Input and Output of Lemmatisation and Token-generation

{lemma,CTAG,O,IR} and both sets can be transformed without loss of information into each other. In the remainder of this paper we abstract from orthographic properties (upper/lower case characters) of the word forms as coded in the O feature. That is, we restrict the lemma set to {lemma,CTAG,IR} and consider it equivalent to a token set.

## 3 Generating Incomplete Lemma Sets

However, we cannot always assume to have all the bits of information even in a reduced lemma set available. Assume, for instance, a verb has to be regenerated in present tense, or a singular noun should be transformed into a plural noun to adjust a stored sentence fragment to a new context. In these cases we still know the lemma of the word and the CTAG. It is unclear, however, what inflection rule should apply to generate the correct word-form.

In this section we report on some experiments to "guess" an appropriate IR for an incomplete lemma set {token,CTAG}. We show that lemmas can be re-converted into word tokens with a very high degree of accuracy even if only partial information is available. We investigate several methods to infer an appropriate inflection rule for generation from corpora and achieve accuracy of more than 99.5%.

Depending on what information is available we distinguish three cases:

1. if the full lemma set is available proceed as described in section 2.5.

2. else if IR is missing, look up the lemmatised BNC whether it contains a form {lemma,CTAG$_{new}$} and retrieve the associated IR. This approach is described in section 3.1

3. else if the BNC does not contain a suitable lemmatised form, "guess" an IR by comparing suffixes of the lemmas. This is described in section 3.2

### 3.1 Re-generating known Wordforms

In this first model we retrieve an IR of an incomplete lemma set {lemma,CTAG$_{new}$} from the lemmatised BNC. The word-form is re-generated that corresponds to the most frequent IR associated to a {lemma,CTAG} in the BNC. We call this model the

| token | lemma | CTAG | IR | freq | generated |
|-------|-------|------|----|----|-----------|
| burned | burn | VVD | 29 | 542 | burned |
| burnt | burn | VVD | L29 | 150 | burned |
| focussed | focus | VVD | L29 | 34 | focused |
| focused | focus | VVD | L29a | 411 | focused |
| brothers | brother | NN2 | 10 | 3511 | brothers |
| brethren | brother | NN2 | L8 | 157 | brothers |
| aquariums | aquarium | NN2 | 10 | 48 | aquaria |
| aquaria | aquarium | NN2 | L8 | 82 | aquaria |
| cookin' | cook | VVG | 29 | 303 | cooking |
| cooking | cook | VVG | 28 | 1043 | cooking |
| coming | come | VVG | 27 | 17726 | coming |
| comeing | come | VVG | 28 | 2 | coming |
| comin' | com | VVG | 29 | 89 | coming |
| comming | com | VVG | 4 | 5 | coming |

Table 5: Regenerating word tokens in the Frequency model

F model since the sought IR is available in the BNC it has access to their frequency distribution.

As plotted in table 8, from a set of 244,500 different words, this produces 0.3648% 'noise'. That is, 892 re-generated words differ from their original form.

In some cases a given {lemma,CTAG} combination occurs with several IR in the BNC. Some examples are given in table 5. For many of these cases several writing variants are possible as e.g. British vs. American writing. When regenerating the word the more frequent variant is chosen. This caused 'noise' in the case of burnt ⇒ burned and focussed ⇒ focused where both variants are correctly reduced to the same {lemma,CTAG} but the more frequently occurring variant is re-generated. Note that the original variant could have been re-generated with the appropriate IR. Thus, {focus,VVD,L29} would generate "focussed" and {burn,VVD,L29} would generate "burnt".

In some cases an erroneous regular form is detected by an inflection rule but the irregular, correct form is re-generated (e.g. aquariums ⇒ aquaria, comeing ⇒ coming). Note also here that the incorrect forms would be re-generated with the appropriate IR.

Most of the 'noise' is, however, due to speech subscription which is part of the BNC (e.g. cookin', comin'). These spoken forms are regenerated in their correct written form (cooking, coming) as shown in table 5. It is 324 -in' forms out of the 892 noisy regenerated words that are reproduced as -ing which accounts for more than 1/3 of the 'noise'.

### 3.2 Guessing a new IR

In case a {lemma,CTAG$_{new}$} does not occur in the BNC and in the tokenisation lexicon, we have to find some other means to infer an appropriate IR.

As a first method we have applied the token generation rules in their pre-defined order. When a lemma suffix matches a generation rule, a word token would be produced. When no token-generation

| token | lemma | CTAG | IR | re-generated |
|---|---|---|---|---|
| surfing | surf | VVG | 4 | surffing |
| boiling | boil | VVG | 4 | boilling |
| aborting | abort | VVG | 4 | abortting |

Table 6: Erroneous token generated in the base-line model

| lemma-suffix | CTAG | IR | rel.freq |
|---|---|---|---|
| Suffix Model $S_1$ | | | |
| t | VVG | 28 | 0.6092 |
| t | VVG | 4 | 0.3022 |
| t | VVG | 18 | 0.0800 |
| t | VVG | L28 | 0.0072 |
| t | VVG | 29 | 0.0013 |
| | | | |
| Suffix Model $S_2$ | | | |
| rt | VVG | 28 | 0.9989 |
| rt | VVG | 29 | 0.0011 |
| | | | |
| Suffix Model $S_3$ | | | |
| ort | VVG | 28 | 0.9998 |
| ort | VVG | 29 | 0.0001 |
| | | | |
| Suffix Model $S_4$ | | | |
| bort | VVG | 28 | 1 |

Table 7: Lemma suffixes from the BNC with CTAG, IR and relative frequencies of IR

| model | # noise | % noise |
|---|---|---|
| F | 892 | 0.365% |
| base | 17357 | 7.099% |
| $S_1$ | 5220 | 2.135% |
| $S_2$ | 3095 | 1.266% |
| $S_3$ | 1798 | 0.735% |
| $S_4$ | 1756 | 0.718% |
| $S_{dyn}$ | 1023 | 0.418% |

Table 8: Comparing noise of different IR estimation models from a set of 244,500 different words.

rule matches, the token is assumed to be identical to the lemma. The method can be seen as a base-line since it just inverses the lemmatisation process.

This method performs quite poorly producing 7.099% noise from the 244,500 word tokens (see table 8). That is, 17,357 words were re-produced differently from how they appear in the original list.

Most error prone were (endings of) plural noun and some verb forms. In the VVG paradigm, for instance, the first matching rule was in many instances IR 4. This rule transforms a double consonant into a single consonant for token $\Rightarrow$ lemma transformation. However, for lemma $\Rightarrow$ token transformation this produces many erroneous tokens as shown in table 6.

In another approach, we have indexed the suffixes of the lemmas from the BNC together with their CTAG and IR. The idea was to match the suffix of the lemmas to be re-generated together with its $CTAG_{new}$ on the indexed lemma suffixes and retrieve the associated IR.

Thus, to retrieve an IR for the incomplete lemma set {abort,VVG} we would look into a list suffixes as in table 7. By checking the last character "t" we have a choice of 5 rules with their relative distribution in the BNC. Similar to the base-line model, we apply the rules in the order of their relative frequencies and generate the token with the first applying rule. Thus, IR 28 is the most frequent inflection rule for the VVG paradigm that occurs with lemmas ending on "t". The inferred set {abort,VVG,28} generates also the correct form "aborting".

This model $S_1$ can be seen as an extention of the base-line model. It reorders the inflection rules according to frequencies in the BNC. As shown in table 8, the suffix model $S_1$ reduces noise to 2.135%.

In further experiments we have extended the length of the suffixes to 2, 3 and up to 5, where each model $S_i$ includes the suffixes of the models $S_{i-1}$. The IR of the lemma to be generated would be chosen from the longest possible suffix. As can be seen in table 7, longer suffixes tend to be associated with fewer IR and show a stronger discrimination between different choices. With a suffix length of 4, inflection rule 28 can be deterministically applied for "abort".

A further enhancement of the method consists in keeping suffixes dynamically up to the length where only one inflection rule applies. There is, for instance, no point in storing {abort,VVG,29} in the suffix lexicon when {bort,VVG,29} is already unambiguous. This not only reduces the number of stored suffixes to slightly more than 20,000 compared to more than 30,000 for the $S_3$ model, but also increases accuracy considerably. Table 8 shows that the model $S_{dyn}$ is only marginally worse than the frequency model F. With 0.05% more noise we can assume to re-generate word tokens from incomplete lemma sets with reasonable precision.

This also means that word tokens can be represented as lemma sets {lemma,CTAG} little loss of information. Lemma sets are the basic entities from which the original word tokens can be re-generated with high accuracy.

## 4 Reversible Parsing/Morphological Generation

This section describes the morpho-syntactic level of the language model. It builds up on the lemma sets and formalises morpho-syntactic properties of the BNC in a reversible manner.

First we parse the lemmatised English sentences.

We use a flat parser that is implemented in KURD (Carl, 2005). 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.

The parser generates 'internal' nodes that express relations between terminal lemma sets similar to a constituent tree. It uses a unique set of parsing tags (PTAGs) that characterise the properties of the subsumed nodes.

From the parses we extract two distinct sets: a normalised context free grammar and a set of "constraints". The set of "constraints" and the grammar complement each other such that the original lemmatised English sentences can be reproduced.

For parsing we consider the lemma sets {lemma,CTAG} the leaves of a phrase-structure tree. For internal nodes, the parser uses a distinct set of 'internal' tags and features.

### 4.1 Parsing

Partial parsing yields a bracketed structure, as shown in the table 9. The proper noun "john", the noun "apple" and the noun phrase "an apple" are bracketed.

{john,NP0}, {eat,VVZ}, {a,AT0}, {apple,NN1}

```
        <s,pres>
           |
  <np,3sg,subj>    {eat,VVZ}    <np,3sg>
       |                            |
  {john,NP0}                    {a,AT0}      <n,3sg>
                                                 |
                                            {apple,NN1}
```
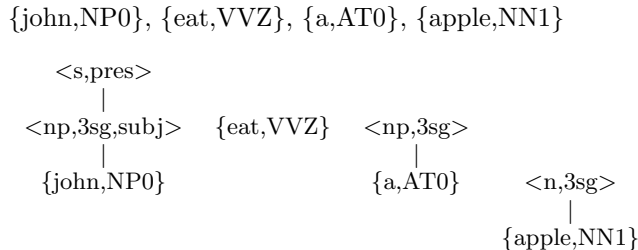
Table 9: Lemma sets and parsed sentence "John eats an apple"

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

In addition we percolate agreement and other information into the internal nodes. Currently we use three features <fcase>, <agr> and <tns>. The <fcase> feature can take the values subject, objective or genitive. The <agr> feature can take (among others) the value 3sg, and the and <tns> feature has the values pres and past.

### 4.2 Reversible Morphological Generation

Lemmatisation abstracts away from number in nouns and number, person and tense in verbs. That is, the PTAG features <agr> and <tns> exhaustively describe the inflectional properties of the subsumed terminal lemma sets. The structure of the parse is designed such that all relevant inflection information for every lemma set is assembled and mirrored in the immediate dominating internal node. Thus, a singular noun, coded as NN1 in the lemma set, is represented as 3sg in the dominating node, a VVZ verb is coded as 3sg.

To transform a singular noun into plural we need to replace NN1 with NN2; to transform a past tense verb into present we transform VVD into VVB or VVZ for 3rd person singular.

In this way, by knowing the <agr> and <tns> values of the internal nodes we can re-produce the original terminal CTAGs with 100% accuracy. Knowing the lemma and the CTAG for each lemma set guarantee reversible deterministic generation of the word token as shown in section 3.

CTAG information in the parse tree that is independent from the PTAGs remains untouched and serves as a default for the token-generation.

We have verified reversibility of the parsed structure on a set of 1.000.000 sentences take from the BNC. In future we also intend to tackle closed class words such as articles, pronouns and prepositions in the same way.

### 4.3 Grammar Inference

We extract a CFG grammar from the parse in the following way. On the one hand, we extract rules from the bracketed structures by transforming the tag into the left-hand side (LHS) of the rules and the content into the right-hand side RHS. Thus the tag <noun> appears on the LHS in rule (4) while the content of the bracketed expression {apple,NN1} occurs in the RHS. On the other hand, templates are generated by replacing the bracketed constituents with their tags. A template consists of terminal symbols and nonterminal symbols. Template (1) consists of one leaf {eat,VVZ} and two non-terminals <np,3sg,subj> and <np,3sg>.

| LHS | | RHS |
|-----|---|-----|
| <snt> | → | <np,subj> , {eat,VVZ} , <np> |
| <np> | → | {a,AT0} , <np> |
| <np> | → | {john,NP0} |
| <n> | → | {apple,NN1} |

Table 10: A sentence template grammar extracted from the parse

The grammar extracted from the parse in table 9 consists of 4 context-free rules, where <snt> is the top-level symbol and <np>, <n> are non-terminals.

Note that at least one terminal symbol must occur in the RHS of the rules.

To reduce the number of different rules in the grammar and to make them consistent amongst each other, some CTAGS are normalised. Thus, all plural nouns are converted into singular (NN2 ⇒ NN1) and all finite verbs VVD and VVB are converted into VVZ (3rd person singular). Internal features in non-terminal nodes are set to 3sg and pres. In the current example, all features correspond already to the default setting.

### 4.4  Extraction of Constraints

In addition to the template grammar, a set of constraints is extracted from the parse. The constraints contain features and structural information of the internal nodes of the parse. Feature information includes the tags <fcase>, <agr> and <tns> as outlined in section 4.1. Examples of the extracted constraints are given in table 11.

The structural information is represented by the numbers of the words that are matched in the structure. Each structural constraint consists of the word numbers matched on the top-level template followed by the sets of word numbers matched in the daughter nodes.

For instance the <snt> node has three daughters from which the first and the third nodes are non-terminals. The terminal {eat,VVZ} is the second word in the sentence. The subtrees of the first and the third daughter nodes are instantiated by word 1 and the set of words 3 and 4 respectively. This information is represented as "2_1_3|4". That is, the first set of number(s) (i.e. 2) represents the words matched by the top-level template, while the words matched in the successive daughter nodes are separated by an underscore "_". This information is extracted for every internal node in the parse. Thus, the <np> node covering "an apple" is linked to the partial tree 3_4, where the 4th wort in the sentence (apple) is a subi-structure of the 3rd word "the".

| wnr | PTAG | agr | fcase | tns |
|---|---|---|---|---|
| 2_1_3\|4 | snt | 3sg | — | pres |
| 1 | np | 3sg | subj | — |
| 3_4 | np | 3sg | — | — |
| 4 | n | 3sg | — | — |

Table 11: Constraints extracted from the parse

### 4.5  Reversible Syntactic Generation

In this section we show how the original parse tree (as e.g. in table 9) can be reproduced from the bag of TL lemmas (e.g. as in table 12), a sentence grammar (as in table 10) and a set of constraints (as in table 11).

Syntactic generation starts from a bag of TL lemmas. Each lemma in the bag is associated with a unique index as shown in table 12. The bag is mapped on the sentence grammar and the word indexes are copied into the matching nodes (see table 13). Thereafter the rules are stitched together to form a parse tree taking into account structural constrains.

| wnr | lemma |
|---|---|
| 1 | john |
| 2 | eat |
| 3 | a |
| 4 | apple |

Table 12: Bag of TL lemmas

Since in the reversible setting, we have for each sentence grammar a consistent set of structural constraints, an optimal combination of the grammar rules can be found from any starting point. That is, irrespectively with which grammar rule we start, the constraints will always lead to the initial best parse tree.

For instance, there is only one structural constraint (i.e. 3_4) that applies to rule #3 in table 13. This constraint requires word number 3 (i.e. "a") to be linked to the template where a subtree is linked to word number 4. The constraint thus favors the partial structure (a (apple)) which combines rules #3 and #4. This partial tree can be stored and inserted in the second slot of rule 2 as required be the constraint "2_1_3|4". Once an optimal parse tree is generated, the internal nodes are instantiated with PTAG features. The results is then input to morphological generation as described in section 4.

| # | LHS | | RHS |
|---|---|---|---|
| 1 | <np> | → | 1: {john,NP0} |
| 2 | <snt> | → | <np> , 2: {eat,VVZ} , <np> . |
| 3 | <np> | → | 3: {a,AT0} , <np> |
| 4 | <n> | → | 4: {apple,NN1} |

Table 13: Instantiated generation grammar

## 5  Conclusion and Outlook

In this paper we have presented a method to decompose sentences into three disjoint sets: a bag of lemmas, a set of structural and morpho-syntactic constraints and a template grammar. Several steps of analysis are involved in the construction of these sets: tagging, lemmatisation, and parsing. We have shown that the decomposition is reversible to a very high degree, i.e. the original sentence can be deterministically re-composed from these sets with only the token-generation remains with 0.05% noise below an optimal result.

While the template grammar serves as an abstracted language model, the bag of items and con-

straints select their preferred combination combinations and fix morphological properties of the senence to be generated.

In the future we want to extend the approach in several ways. We need to investigate in how far and with what precision a particular sentence grammar can be retrieved from a large set of templates and grammar rules. This investigation will follow the approach of a previous study in (Carl et al., 2005). Sophisticated weighing and selecting strategies are required. For instance, in many cases more than 200,000 rules are extracted from a templates grammar with 1.8 million entries. The extracted rules share one or more tokens with the lemmas in the bag. Since exhaustive combination of all rules is infeasible, the matched rules have to be weighted and graded using different knowledge resources.

Once we know what item and constraints are required in the TL bag to extract a particular sentence grammar, we will try to generate new sentences that are not in the original TL corpus.

Within the METIS-II consortium we plan to run an experiment where TL bags obtained from a bilingual dictionary are to be generated in the TL. These bags can be expected to contain ambiguities and noise and constraints be partially inconsistent with the retrieved sentence grammar.

These experiments will shed more light on the usefulness of the approach proposed in this paper and will show whether further constraints and mechanisms that certainly will turn out to become necessary can be implemented consistently in the proposed framework.

## References

Toni Badia. 2005. An n-gram approach to exploiting monolingual corpus for MT. In *Second EBMT Workshop*.

Thorsten Brants. 2000. A Statistical Part-of-Speech Tagger? In *Proceedings of TUANLP*, pages 224–231.

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

Yunbo Cao and Hang Li. 2000. Base Noun Phrase Translation: Using Web Data and the EM Algorithm. In *COLING*.

Michael Carl, Ecaterina Rascu, and Paul Schmidt. 2005. Using template grammars for shake & bake paraphrasing. In *EAMT*.

Michael Carl. 2003. Inducing Translation Grammars from Bracket Alignments. In *Recent Advances in Example-Based Machine Translation*.

Michael Carl. 2005. *KURD*. IAI, electronic working paper 38. forthcoming.

Ilyas Cicekli and H. A Guvenir. 2003. Learning Translation Templates from Bilingual Translation Examples. In *Recent Advances in Example-Based Machine Translation*.

Ilyas Cicekli. 2005. Learning Translation Templates with Type Constraints. In *Workshop on EBMT*.

Joshua Goodman. 2002. The State of the Art in Language Modeling. In *Tutorial Presented at AMTA*.

Nano Gough and Andy Way. 2004. Example-Based Controlled Translation. In *EAMT*.

Nizar Habash and Bonnie Dorr. 2002. Handling Translation Divergences: Combining Statistical and Symbolic Techniques in Generation-Heavy Machine Translation. In *AMTA*.

Christos Malavazos and Stelios Piperidis. 2000. Application of Analogical Modelling to Example Based Machine Translation. In *COLING*.

A. Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *ICSLP*.

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

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