Tree-based Target Language Modeling

Vincent Vandeghinste
Centre for Computational Linguistics - KULeuven
Leuven, Belgium
vincent@ccl.kuleuven.be

Abstract

In this paper we describe an approach to target language modeling which is based on a large treebank. We assume a bag of bags as input for the target language generation component, leaving it up to this component to decide upon word and phrase order. An experiment with Dutch as target language shows that this approach to candidate translation reranking outperforms standard n-gram modeling, when measuring output quality with BLEU, NIST, and TER metrics.

1 Acknowledgements

2 Introduction

In this paper we describe an approach to target language modeling using large treebanks. This introduction starts with a description of the MT system for which this target language modeling component is intented and continues with a short description of related research.

In section 3 we describe the details of the target language modeling component and in section 4 we describe an evaluation experiment for this component. Section 5 draws conclusions and sketches future work.

2.1 System description

We are developing a data-driven hybrid approach towards machine translation, reusing as much as possible already existing tools and resources to set up an MT architecture much like a classic rule-based transfer system. Instead of manually designing the rules, we intend to derive them from large parallel and monolingual (uncorrected) treebanks.

The system requires a source language parser and a parallel treebank, aligned from the sentence level up to the word level (Och and Ney, 2003), including sub-sentential alignment (Tiedemann, 2003; Tinsley et al., 2007, Macken and Daelemans, 2008). To get a parallel treebank we parse both the source and target language components of parallel corpora à la Europarl (Koehn, 2005). Each tree pair, sub-tree pair or word pair presents an example translation pair, and becomes a dictionary entry. This way we are removing the conceptual distinction between a dictionary and a parallel corpus, like Vandeghinste (2007).

In a similar fashion, but making abstraction of the concrete words, we derive a set of transfer rules from the available alignments. A translation model is built by counting the frequencies of occurrence of all these alignments.

The source language sentence is syntactically parsed, and the parse tree (and its sub-trees) is matched with the source language side parse trees of the dictionary/parallel treebank. The retrieved target fragments are then restructured according to the information in the transfer rules resulting in a target language bag of bags, which is structured like a parse tree, but without implying any surface order in the daughters of each node. When larger units are retrieved from the dictionary, their surface order is preserved, implying that some nodes in the bag of bags are not bags but trees, with ordered daughters.

It is up to the target language generation component to determine the lexical selection (which translation alternatives are preferred) and optimal surface ordering using the target language treebank. It is this component which we describe and

evaluate in the rest of this paper.

When the system has generated a translation, it is up to the human post-editor to accept the translation or to correct it. For this purpose a web-based post-editing interface is being designed, which allows adding, deleting, substituting, and moving words and phrases. The post-editor can choose amongst several translation alternatives for the sentence, or for certain parts of the sentence. When a sentence is accepted the post-editing information is fed back into the system's databases, updating the weights of both the translation model and the target language generation model.

2.2 Related Research

The hybrid MT system described in the previous section is similar to the *Data-Oriented Translation* (DOT) approach, which was first proposed by Poutsma (1998) and further researched by Hearne (2005). DOT uses Data-Oriented Parse Trees (Bod, 1992), whereas we use either rule-based parsers based on a set of linguistic rules and a stochastic disambiguation component or we use stochastic parsers trained on a manually parsed or corrected treebank. The DOT approach only uses small corpora and a limited domain, whereas we intend to use large corpora and a general domain (news).

The target language generation approach is somewhat similar to the *feature templates* used by the translation candidate reranking component of Velldal (2007), although there are some important differences: Velldal's feature templates can have a higher depth, whereas the patterns we extract can be seen as context-free rewrite rules, only capturing information about a mother and its immediate daughters. This can be attributed to the fact that the LOGON system (Lønning et al., 2004) for which Velldal built the component is a limited domain MT system (Tourist information) whereas we intend to build a large domain system (News), so we are using much larger corpora. Storing information at a similar level as Velldal is not feasible with such large treebanks.

Furthermore, our system borrows ideas for combining target language fragments from the METIS-II system (Carl et al., 2008; Vandeghinste, 2008).

Our system is being implemented from Dutch to English and French, and vice versa. In the rest of this paper, we assume Dutch as the target language.

3 The Target Language Generation Component

This section describes the approach we use for target language modeling. In section 3.1 we describe the input this component expects, section 3.2 describes the training procedure and the preprocessing steps applied on the training data, and section 3.3 describes how the target language generation component actually works.

The target language generation component is based on a large target language treebank. The input is assumed to be a source language independent bag of bags, as all elements in this bag are coming from the target language side of the dictionary, and the structure of the bag of bags is mapped onto the target language structure through the dictionary and the transfer rules.

3.1 Bag of Bags as input

We define a bag of bags as a *set of sets*, or in our case, as a parse tree representing the target language sentence, in which for each node,[1] the surface order of the daughters of that bag is undetermined, representing all permutations of the list of daughters. It is up to the target language generation component to resolve these bags and come up with the best solution.

In figure 1 you find an example of a bag of bags in xml-format representing the Dutch sentence "*Zie ook het kaartje hieronder.*" [Eng: Also see the map below.]. A regular parse tree for this sentence is presented in figure 2. Figure 1 represents besides this sentence numerous ($2! \times 4! \times 2! = 96$) other surface strings, each a permutation of the words in the sentence.

Note that in figure 1 we left out some features in the `<bag>` tags of the bag of bags for clarity and presentational purposes. The bag of bags is exactly the same as the xml output of the syntactic parse for the same sentence generated by the Alpino parser (van Noord, 2006), apart from the fact that the `<node>` tags in the parse tree have been replaced by `<bag>` tags in the bag of bags, indicating that these bags still need to be resolved, and from the fact that it does not contain position information.

The Alpino parser is the parser we use for Dutch syntactic analysis. It is a parser which is based on head-driven phrase structure grammar (Pollard

---

[1] Some of the sub-trees are coming straight from the dictionary, so they are not sub-bags and do not need to be resolved.

Figure 1: An example bag of bags

```
<bag cat="top" rel="top">
 <bag cat="sv1" rel="--">
  <bag frame="verb(hebben,sg1,
            transitive_ndev_ndev)"
     pos="verb" rel="hd" word="Zie"/>
  <bag frame="sentence_adverb" pos="adv"
     rel="mod" word="ook"/>
  <bag cat="np" rel="obj1">
   <bag frame="determiner(het,nwh,nmod,
             pro,nparg,wkpro)"
      pos="det" rel="det" word="het"/>
   <bag frame="noun(het,count,sg)"
      pos="noun" rel="hd"
      word="kaartje"/>
  </bag>
  <bag frame="er_adverb(onder)" pos="pp"
     rel="mod" word="hieronder"/>
 </bag>
 <bag frame="punct(punt)" pos="punct"
     rel="--" word="."/>
</bag>
```

Figure 2: Parse tree for the example sentence (without frames)
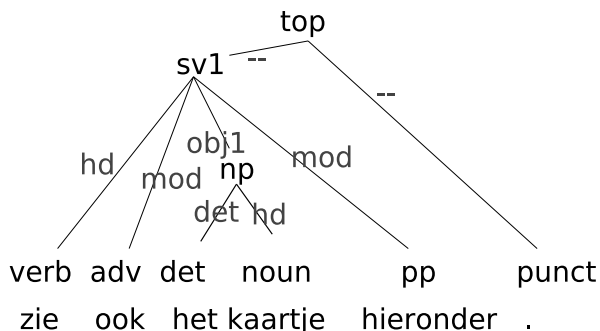
and Sag, 1994) giving both phrase structure and dependency information.

Resolving the bag of bags in a bottom-up fashion, we first resolve the noun phrase (NP) "*het kaartje*" [Eng: the map]. There are two possible permutations for this NP, and we want to find the most probable. How this is done is explained in section 3.3.

When the NP is resolved, we need to resolve the *sv1*, which stands for *a sentence with the verb in first position*. The sv1 has four daughters, so this amounts to 24 (4!) different possible surface orders.[2] One of these daughters has two possible outcomes, so this already totals 48 translation alternatives under investigation.

This procedure is applied on all non-terminal bags.

3.2 Training the target language generation component

In order to resolve the bags, we train the target language generation component on a large treebank. For Dutch, this treebank was automatically annotated by the Alpino parser (van Noord, 2006), and is available online at http://www.let.rug.nl/∼vannoord/trees/.

It consists, amongst others, of the following corpora: the Spoken Dutch corpus (CGN) (Oostdijk et al., 2002), the Lassy corpus (van Noord et al., 2006), the Dutch part of Europarl (Koehn, 2005), and the Dutch wikipedia.

The total corpus used in the experiments in section 4 consists of 290,658,861 words in 18,048,702 sentences, averaging 16.10 words per sentence.

From each of these sentences, we collect the *rewrite rules* at different levels of abstraction. For instance, for the example sentence "*Zie ook het kaartje hieronder*", we would collect the information in table 1.[3]

Note that we abbreviated some of the frames to fit in the table and that we use "|" as a field separator between the different kinds of information represented in our rewrite rules. Consecutive elements on the right-hand side of the rules are written with a space inbetween or on a new line. For instance, the sv1 rule has four right-hand side symbols on every abstraction level.

We distinguish several different levels of abstraction, going from very abstract (Level 1: Relations) to very concrete (Level 7: Head + Frame/Cat + Relations).

1. Relations (Rel): Containing the dependency relations and the function information.

2. Part-of-speech/Category (Pos/Cat): containing the parts-of-speech of terminal nodes and the category for non-terminal nodes.

3. Pos/Cat + Rel: containing the combinations of parts-of-speech/category information and dependency information.

4. Frame/Cat: Containing frame information for terminal nodes and the category information for non-terminals. Frames are generated by the Alpino parser, and are a very fine-grained part-of-speech tag.

---

[2]Because we treat all categories the same, we do not make use of the fact that for an *sv1* we know that the verb should be, by definition, in first position.

[3]This sentence has a parse tree exactly like the example bag of bags, apart from replacing the <bag> tags with <node> tags.

Table 1: Extracting information from a sentence at different abstraction levels

Level 1: **Relations**
| ```
top :    -- --
sv1 :   hd mod obj1 mod
np :    det hd
``` |

Level 2: **Pos/Cat**
| ```
top :    sv1 punct
sv1 :   verb adv np pp
np :    det noun
``` |

Level 3: **Pos/Category + Relations**
| ```
top :    sv1|-- punct|--
sv1:   verb|hd adv|mod np|obj1 pp|mod
np :    det|det noun|hd
``` |

Level 4: **Frame/Category**
| ```
top :    sv1 punct(punt)
sv1 :   verb(hebben,sg1,transitive...)
         sentence_adverb
         np
         pp
np :    determiner(het,nwh,nmod,pro...)
         noun(het,count,sg)
``` |

Level 5: **Frame/Category + Relations**
| ```
top :    sv1|-- punct(punt)|
sv1 :   verb(hebben,sg1,transitive...)|hd
         sentence_adverb|mod
         np|obj1
         pp|mod
np :    determiner(het,nwh,nmod,pro...)|det
         noun(het,count,sg)|hd
``` |

Level 6: **Head + Pos/Cat + Relations**
| ```
top :    sv1|--|Zie punct|--|.
sv1 :   verb|hd|Zie
         adv|mod|ook
         np|obj1|kaartje
         pp|mod|hieronder
np :    det|det|het noun|hd|kaartje
``` |

Level 7: **Head + Frame/Cat + Relations**
| ```
top :    sv1|--|Zie punct(punt)|--|.
sv1 :   verb(hebben,sg1,...)|hd|Zie
         sentence_adverb|mod|ook
         np|obj1|kaartje
         pp|mod|hieronder
np :    determiner(het,nwh...)|det|het
         noun(het,count,sg)|hd|kaartje
``` |

Table 2: Number of different labels and bags

Abstraction Level	Labels	Bags
1 Rel	32	50,233
2 Pos/Cat	48	568,299
3 Pos + Rel	510	1,584,535
4 Frame	36,729	9,764,647
5 Frame + Rel	50,130	10,251,079
6 Head + Pos + Rel	22,924,782	60,753,604
7 Head + Frame + Rel	26,400,004	61,283,814

5. Frame/Cat + Rel: containing the combinations of frame/category and relation information.

6. Head + Pos/Cat + Rel: containing the combination of the head word of a node with the parts-of-speech /category and relation.

7. Head + Frame/Cat + Rel: containing the combination of the head word of a node with the frame and relation.

In table 2 we present some information about our database for the total corpus size of 18 million sentences. The second column (Labels) indicates the number of different labels (types) for that abstraction level. The third column (Bags) shows the number of different bags at that level. If the corpus contains two or more permutations of the same bag, then these are counted as one bag.

All this data is collected over the whole treebank, and put in a database, precalculating which patterns are permutations of each other, and adding the frequency of occurrence for each of these permutations.

We have one database table per category per abstraction level, and we have 25 categories for Dutch, resulting in 175 tables. Each of these tables contains one row per bag and one column per subcorpus. For each bag and each corpus, we store the surface order of the bag elements and their frequency, allowing multiple surface orders and frequencies per database cell.

The use of separate columns for sub-corpora allows us to easily activate and deactive certain parts of the total corpus. It is a design choice that facilitates adapting the MT system to specific domains by activating the appropriate columns.

3.3 Matching the Bag of bags with the training data

We want to resolve the noun phrase-bag "*het kaartje*", knowing that there are two possible permutations.

We start of on the most concrete level, looking for the occurrence in the training data of either

```
np :  det(...)|det|het
      noun(...)|hd|kaartje
```
or
```
np :  noun(...)|hd|kaartje
      det(...)|det|het
```

If one or both of these occur in the training data, then we use their relative frequencies as weights for the solution. When neither of them occurs in the training data, we go to a more abstract level, hoping to find information regarding the relative higher occurrence of one permutation over the other, cascading over the different abstraction levels, until the bag is resolved. In the rare case that none of the abstraction levels can resolve the bag, all permutations get the same weight.

We use a set of cut-off parameters to limit the number of alternative analyses under consideration to a manageable number. Currently, we keep only track of the 10 best scoring alternatives. When no information or equal frequencies are found, and the bag would generate more than 30 permutations, we cut off at 30. This is especially required in the experimental conditions where the corpus size is still low (cf section 4). We stop processing an alternative solution if its weight is 10 times lower than the weight of the current best solution, and for each node, we allow a maximum of 100 combinations of the solutions of the daughters. As the system is currently fast enough, we have not yet investigated different values for these cut-off parameters, but it is clear that cutting off sooner would lead to faster processing but lower accuracy. Most of these cut-off parameters come in action only at low corpus sizes and/or in experimental conditions with only high abstraction levels.

4 Experiment

In this section we describe an experiment in which we evaluate the target language generation component of our MT system in isolation, excluding factors that might contribute to the translation quality in good or bad sense that are not part of the target language model.

Section 4.1 describes the methodology that is used for the experiment, and section 4.2 describes the evaluation results.

4.1 Methodology

In a way, we are translating from Dutch to Dutch, only evaluating the ordering mechanism used in the target language generation component.

We tested the quality of the output of the target language generation component by comparing it to the input sentence from which the bag of bags originates, which serves as a reference translation when evaluating with BLEU (Papineni et al., 2002), NIST (Doddington, 2002), and TER (Snover et al., 2006).

Additionally we also measured the number of exact matches: those cases in which the output sentence is identical to the input sentence.

We have constructed a test set of 575 real-life sentences from a real translation context that were parsed with Alpino and converted into bags.

We have several test conditions in two dimensions:

1. *Corpus size*: expressed in number of sentences. The treebank consists of several sub-corpora, and we tested the system while gradually adding these sub-corpora. The size of these sub-corpora serves as data points on the X-axis in figures 3, 4, 5, 6, and 7.

2. *Abstraction level*: we have described the seven abstraction levels for Dutch in section 3.2. We tested the system with only the data for the most abstract level available, gradually adding less abstract levels. These are the data series 1 to 7 in the legend.

As a baseline, we also calculated the quality of a *trigram language models*. We used the SRILM toolkit (Stolcke, 2002) to train a backoff trigram model. Additional baseline testing with a fourgram model with Chen and Goodman's (1998) modified Kneser-Ney discounting did not yield better results. As it is not feasible to generate all permutations and then calculate their likelihood, we implemented a branch and bound approach. For each sub-bag, all permutations were generated and these were ordered according to their likelihood, keeping only the 10 best for each sub-bag. When any of these permutations contained more than n words, a sliding window of size n was used to estimate their likelihood. This procedure was recursively applied until the whole bag is resolved.

Figure 3: Effect of corpus size and abstraction level on BLEU score

Figure 5: Effect of corpus size and abstraction level on TER score

Figure 4: Effect of corpus size and abstraction level on NIST score

Figure 6: Effect of corpus size and abstraction level on percentage of Exact Matches

For exact match we calculated a baseline by counting the total number of possible permutations and the probability of ranodmly picking an exact match.

4.2 Results

When looking at figure 3 it is clear that the addition of the least abstract levels yields the best results, although there is not much difference between levels 6 and 7. At the largest corpus size, level 6 even outperforms level 7. This can be explained by the fact that there is only a relatively small difference in granularity between levels 6 and 7, which is clear when looking at table 2. There is a reduction of numbers of bags of less than 1%, so the

abstraction is very limited. In future versions of the system, we might omit level 7 as it does not add any accuracy.

It is also clear that for all corpus sizes, abstraction levels 4, 5, 6, and 7 outperform the baseline.

The results are consistent for the NIST scores shown in figure 4.

When looking at the TER scores in figure 5, the same observations are still true. Note that TER expresses an error rate, so lower scores are better.

A somewhat unexpected result is the fact that level 1 consistently outperforms levels 2 and 3. We assume some kind of artefact and will investigate this further.

The percentage of exact matches, as presented

in figure 6 confirms the results from the other metrics. Note that the probability of randomly picking one of the possible permutations of the input bag of bag as its solution would result in an exact match baseline of 0.0000911%, so all experimental conditions improve over this baseline.

5 Conclusions and Future Work

We have set up a translation generation component for a parse and corpus-based MT system. This component requires a bag of bags as input, each bag and sub-bag representing all permutations of their respective daughters.

We trained the component on a large target language treebank (with fully automatic parses) so we can look up for each of the bags whether it occurs in the corpus, in what surface order, and with what frequency.

Comparing our system to a standard *n-gram* model we can conclude that our system clearly outperforms this baseline.

Although the results of the experiment suggest that we have reached some kind of ceiling in translation quality, we intend to at least double the size of the target language treebank and test whether we can break through these ceilings.

Figure 7 shows the percentages of new bags to be added to the database for each of the abstraction levels when gradually adding the subcorpora. Adding new corpora seems to add relatively little new information to the most abstract levels, but for the more concrete levels, growth percentages are still more than 50%, meaning that more than 50% of the bags found in the new corpus were unseen in the previous corpora.

We set up this experiment in order to estimate the upper bound of our MT system. Connecting this component to the other components of our MT system will reveal its true quality, but the results up to now are very encouraging.

We will also implement this approach for the other languages in our MT system, but probably with less abstraction levels. For instance, for English we use the Stanford parser (Klein and Manning, 2003), which generates parts-of-speech, dependency relations, categories, and words, but not frames or anything equivalent.

References

Bod, R. (1992). A Computational Model of Language Performance: Data-Oriented Parsing.

Figure 7: Growth percentage for each abstraction level

In. C. Boîtet (ed.), *Proceedings of the fifteenth International Conference on Computational Linguistics (COLING'92)*. International Committee on Computational Linguistics. Nantes, France. pp. 855-859.

Carl, M., Melero, M., Badia, T., Vandeghinste, V., Dirix, P., Schuurman, I., Markantonatou, S., Sofianopoulos, S., Vassiliou, M., and Yannoutsou, O. (2008). METIS-II: Low Resources Machine Translation : Background, Implementation, Results, and Potentials. *Machine Translation* 22(1). pp. 69-99. Springer.

Chen, S.F., and Goodman, J. (1998). An Empirical Study of Smoothing Techniques for Language Modeling. *Technical Report TR-10-98*. Computer Science Group, Harvard U., Cambridge, MA.

Doddington, G. (2002). Automatic Evaluation of Machine Translation Quality using N-gram Co-occurrence Statistics. In *Proceedings of the Second Human Language Technology Conference (HLT)*. Morgan Kaufmann. San Diego, USA. pp. 138-145.

Hearne, M. (2005). *Data-Oriented Models of Parsing and Translation*. PhD thesis. Dublin City University. Ireland.

Klein, D., and Manning, C. (2003). Accurate Unlexicalized Parsing. In *Proceedings of 41st Annual Meeting of the Association of Computational Linguistics (ACL)*. Sapporo, Japan. pp. 423-430.

Koehn, P. (2005). Europarl. A parallel corpus for statistical machine translation. In *Proceedings of MT Summit X*. Phuket, Thailand. pp. 79-97.

Lønning, J.T., Oepen, S., Beermann, D., Hellan, L., Carroll, J., Dyvik, H., Flickinger, D., Johannsen, J.B., Meurer, P., Nordgård, T., Rosén, V., and Velldal, E. (2004). LOGON. A Norwegian MT effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*. Uppsala, Sweden.

Macken, L., and Daelemans, W. (2009). Aligning linguistically motivated phrases. In *Computational Linguistics in the Netherlands 2007: Selected papers from the eighteenth CLIN meeting*. LOT Netherlands Graduate School of Linguistics. Utrecht. pp. 37-52

Och, F., and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29 (1), pp. 19-51.

Oostdijk, N., Goedertier, W., Van Eynde, F., Boves, L., Martens, J.P., Moortgat, M., and Baayen, H. (2002). Experiences form the Spoken Dutch Corpus Project. In *Proceedings of the 3rd International conference on Language Resources and Evaluation (LREC)*. Las Palmas, Spain. pp. 340-347.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). BLEU: a method for automatic evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Philadelphia, USA. pp. 311-318.

Pollard, C., and Sag, I. (1994). *Head-driven Phrase Structure Grammar*. CSLI Stanford. University of Chicago Press. Stanford, USA.

Poutsma, A. (1998). Data-Oriented Translation. Presented at the *Ninth Conference of Computational Linguistics in the Netherlands*. Leuven, Belgium.

Snover, M., Dorr, B., Schwartz, R., Micciula, L, and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation inthe Americas (AMTA)*. Cambridge, USA. pp. 223-231.

Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing*. Denver, Colorado, September 2002.

Tiedemann, J. (2003). *Recycling Translations - Extraction of Lexical Data from Parallel Corpora and their Application in Natural Language Processing*. PhD. Studia Linguistica Upsaliensia 1.

Tinsley, J., Zechev, V., Hearne, M., and Way, A. (2007). Robust Language Pair-Independent Sub-Tree Alignment. *Proceedings of MT Summit XI*. Copenhagen. pp. 467-474.

Vandeghinste, V. (2007). Removing the Distinction Between a Translation Memory, a Bilingual Dictionary and a Parallel Corpus. In *Proceedings of Translating and the Computer*, 29. ASLIB. London, UK.

Vandeghinste, V. (2008). *A Hybrid Modular Machine Translation System*. PhD. Katholieke Universiteit Leuven. LOT Netherlands Graduate School of Linguistics. Utrecht.

van Noord, G., Schuurman, I., and Vandeghinste, V. (2006). Syntactic Annotation of Large Corpora in STEVIN. In *Proceedings of the 5th International conference on Language Resources and Evaluation LREC*. Genova, Italy.

van Noord, G. (2006). At Last Parsing Is Now Operational. In *Proceedings of TALN*, Leuven, Belgium.