Productive Generation of Compound Words in Statistical Machine Translation

Sara Stymne
Linköping University
Linköping, Sweden
sara.stymne@liu.se

Nicola Cancedda
Xerox Research Centre Europe
Meylan, France
nicola.cancedda@xrce.xerox.com

Abstract

In many languages the use of compound words is very productive. A common practice to reduce sparsity consists in splitting compounds in the training data. When this is done, the system incurs the risk of translating components in non-consecutive positions, or in the wrong order. Furthermore, a post-processing step of *compound merging* is required to reconstruct compound words in the output. We present a method for increasing the chances that components that should be merged are translated into contiguous positions and in the right order. We also propose new heuristic methods for merging components that outperform all known methods, and a learning-based method that has similar accuracy as the heuristic method, is better at producing novel compounds, and can operate with no background linguistic resources.

1 Introduction

In many languages including most of the Germanic (German, Swedish etc.) and Uralic (Finnish, Hungarian etc.) language families so-called closed compounds are used productively. Closed compounds are written as single words without spaces or other word boundaries, as the Swedish:

gatstenshuggare	*gata + sten + huggare*
paving stone cutter	*street stone cutter*

To cope with the productivity of the phenomenon, any effective strategy should be able to correctly process compounds that have never been seen in the training data as such, although possibly their components have, either in isolation or within a different compound.

The extended use of compounds make them problematic for machine translation. For translation into a compounding language, often fewer compounds than in normal texts are produced. This can be due to the fact that the desired compounds are missing in the training data, or that they have not been aligned correctly. When a compound is the idiomatic word choice in the translation, a MT system can often produce separate words, genitive or other alternative constructions, or translate only one part of the compound.

Most research on compound translation in combination with SMT has been focused on translation *from* a compounding language, into a non-compounding one, typically into English. A common strategy then consists in splitting compounds into their components prior to training and translation.

Only few have investigated translation into a compounding language. For translation into a compounding language, the process becomes:

- Splitting compounds on the target (compounding language) side of the training corpus;
- Learn a translation model from this split training corpus from source (e.g. English) into decomposed-target (e.g. decomposed-German)
- At translation time, translate using the learned model from source into decomposed-target.
- Apply a post-processing "merge" step to reconstruct compounds.

The merging step must solve two problems: identify which words should be merged into compounds, and choose the correct form of the compound parts.

The former problem can become hopelessly difficult if the translation did not put components nicely side by side and in the correct order. Preliminary to merging, then, the problem of promoting translations where compound elements are correctly positioned needs to be addressed. We call this promoting compound *coalescence*.

2 Related work

The first suggestion of a compound merging method for MT that we are aware of was described by Popović et al. (2006). Each word in the translation output is looked up in a list of compound parts, and merged with the next word if it results in a known compound. This method led to improved overall translation results from English to German. Stymne (2008) suggested a merging method based on part-of-speech matching, in a factored translation system, where compound parts had a special part-of-speech tag, and compound parts are only merged with the next word if the part-of-speech tags match. This resulted in improved translation quality from English to German, and from English to Swedish (Stymne and Holmqvist, 2008). Another method, based on several decoding runs, was investigated by Fraser (2009).

Stymne (2009a) investigated and compared merging methods inspired by Popović et al. (2006), Stymne (2008) and a method inspired by morphology merging (El-Kahlout and Oflazer, 2006; Virpioja et al., 2007), where compound parts were annotated with symbols, and parts with symbols in the translation output were merged with the next word.

3 Promoting coalescence of compounds

If compounds are split in the training set, then there is no guarantee that translations of components will end up in contiguous positions and in the correct order. This is primarily a language model problem, and we will model it as such by applying POS language models on specially designed part-of-speech sets, and by applying language model inspired count features.

The approach proposed in Stymne (2008) consists in running a POS tagger on the target side of the corpus, decompose only tokens with some predefined POS (e.g. Nouns), and then marking with special POS-tags whether an element is a head or a modifier. As an example, the German compound "Fremdsprachenkenntnisse", originally tagged as N(oun), would be decomposed and re-tagged before training as:

fremd	sprachen	kenntnisse
N-Modif	N-Modif	N

A POS n-gram language model using these extended tagset, then, naturally steers the decoder towards translations with good relative placement of these components

We modify this approach by blurring distinctions among POS not relevant to the formation of compounds, thus further reducing the tagset to only three tags:

- N-p – all parts of a split compound except the last
- N – the last part of the compound (its head) and all other nouns
- X – all other tokens

The above scheme assumes that only noun compounds are treated but it could easily be extended to other types of compounds. Alternatively, splitting can be attempted irrespective of POS on all tokens longer than a fixed threshold, removing the need of a POS tagger.

3.1 Sequence models as count features

We expect a POS-based n-gram language model on our reduced tagset to learn to discourage sequences unseen in the training data, such as the sequence of compound parts not followed by a suitable head. Such a generative LM, however, might also have a tendency to bias lexical selection towards translations with fewer compounds, since the corresponding tag sequences might be more common in text. To compensate for this bias, we experiment with injecting a little dose of a-priori knowledge, and add a count feature, which explicitly counts the number of occurrences of POS-sequences which we deem good and bad in the translation output.

Table 1 gives an overview of the possible bigram combinations, using the three symbol tagset, plus sentence beginning and end markers, and their judgment as good, bad or neutral.

Combination	Judgment
N-p N-p	Good
N-p N	Good
N-p $< \backslash s >$	Bad
N-p X	Bad
all other combinations	Neutral

Table 1: Tag combinations in the translation output

We define two new feature functions: one counting the number of occurrences of Good sequences (the *Boost model*) and the other counting the occurrences of Bad sequences (the *Punish model*). The two models can be used either in isolation or combined, with or without a further POS n-gram language model.

4 Merging compounds

Once a translation is generated using a system trained on split compounds, a post-processing step is required to merge components back into compounds. For all pairs of consecutive tokens we have to decide whether to combine them or not. Depending on the language and on preprocessing choices, we might also have to decide whether to apply any boundary transformation like e.g. inserting an 's' between components.

The method proposed in Popović et al. (2006) maintains a list of known compounds and compound modifiers. For any pair of consecutive tokens, if the first is in the list of known modifiers and the combination of the two is in the list of compounds, than the two tokens are merged.

A somewhat orthogonal approach is the one proposed in Stymne (2008): tokens are labeled with POS-tags; compound modifiers are marked with special POS-tags based on the POS of the head. If a word with a modifier POS-tag is followed by either another modifier POS-tag of the same type, or the corresponding head POS-tag, then the two tokens are merged.

In the following sections we describe how we modify and combine these two heuristics, and how we alternatively formulate the problem as a sequence labelling problem suitable for a machine learning approach.

4.1 Improving and combining heuristics

We empirically verified that the simple heuristics in Popović et al. (2006) tends to misfire quite often, leading to too many compounds. We modify it by adding an additional check: tokens are merged if they appear combined in the list of compounds, but only if their observed frequency as a compound is larger than their frequency as a bigram. This blocks the merging of many consecutive words, which just happen to form a, often unrelated, compound when merged, such as *för små* (*too small*) into *försmå* (*spurn*) in Swedish. Compound and bigram frequencies can be computed on any available monolingual corpus in the domain of interest.

We furthermore observed that the (improved) list-based heuristic and the method based on POS patterns lead to complementary sets of false negatives. We thus propose to combine the two heuristics in this way: we merge two consecutive tokens if they would be combined by either the list-based heuristic or the POS-based heuristic. We empirically verified improved performance when combining heuristics in this way (Section 5.2).

4.2 Compound merging as sequence labelling

Besides extending and combining existing heuristics, we propose a novel formulation of compound merging as a sequence labelling problem. The opposite problem, compound splitting, has successfully been cast as a sequence labelling problem before (Dyer, 2010), but here we apply this formulation in the opposite direction.

Depending on choices made at compound splitting time, this task can be either a binary or multiclass classification task. If compound parts were kept as-is, the merging task is a simple concatenation of two words, and each separation point must receive a binary label encoding whether the two tokens should be merged. An option at splitting time is to normalize compound parts, which often have a morphological form specific to compounds, to a canonical form (Stymne, 2009b). In this case the compound form has to be restored before concatenating the parts. This can be modeled as a multiclass classifier that have the possible boundary transformations as its classes.

Consider for instance translating into German the

English sentence:

> Europe should promote the knowledge of foreign languages

Assuming that the training corpus did not contain occurrences of the pair (“knowledge of foreign languages”,”fremdsprachenkenntnisse”) but contained occurrences of (“knowledge”,”kenntnisse”), (“foreign”,”fremd”) and (“languages”,”sprachen”), then the translation model from English into decomposed-German could be able to produce:

> Europa sollte fremd sprachen kenntnisse fördern

We cast the problem of merging compounds as one of making a series of correlated binary decisions, one for each pair of consecutive words, each deciding whether the whitespace between the two words should be suppressed (label “1”) or not (label “0”). In the case above, the correct labelling for the sentence would be $\{0,0,1,1,0\}$, reconstructing the correct German:

> Europa sollte fremdsprachenkenntnisse fördern[1]

If conversely, components are normalized upon splitting, then labels are no longer binary, but come from a set describing all local orthographic transformations possible for the language under consideration. In this work we limited our attention to the case when compounds are not normalized upon splitting, and labels are hence binary.

While in principle one could address each atomic merging decision independently, it seems intuitive that a decision taken at one point should influence merging decisions in neighboring separation points. For this reason, instead of a simple (binary or n-ary) classification problem, we prefer a sequence labelling formulation.

The array of sequence labelling algorithms potentially suitable to our problem is fairly broad, including Hidden Markov Models (HMMs) (Rabiner, 1989), Conditional Random Fields (CRFs) (Lafferty et al., 2001), structured perceptrons (Collins, 2002), and more. Since the focus of this work is on the application rather than on a comparison among alternative structured learning approaches, we limited ourselves to a single implementation. Considering its good scaling capabilities, appropriateness in presence of strongly redundant and overlapping features, and widespread recognition in the NLP community, we chose to use Conditional Random Fields.

4.2.1 Features

Each sequence item (i.e. each separation point between words) is represented by means of a sparse vector of features. We used:

- Surface words: word-1, word+1
- Part-of-speech: POS-1, POS+1
- Character n-grams around the merge point
  - 3 character suffix of word-1
  - 3 character prefix of word+1
  - Combinations crossing the merge points: 1+3, 3+1, 3+3 characters
- Normalized character n-grams around the merge point, where characters are replaced by phonetic approximations, and grouped according to phonetic distribution, see Figure 1 (only for Swedish)
- Frequencies from the training corpus, binned by the following method:

  $$\bar{f} = \begin{cases} 10\lfloor \log_{10}(f) \rfloor & \text{if } f > 1 \\ f & \text{otherwise} \end{cases}$$

  for the following items:
  - bigram, word-1,word+1
  - Compound resulting from merging word-1,word+1
  - Word-1 as a true prefix of words in the corpus
  - Word+1 as a true suffix of words in the corpus
- Frequency comparisons of two different frequencies in the training corpus, classified into four categories: freq1 $=$ freq2 $= 0$, freq1 $<$ freq2, freq1 $=$ freq2, freq1 $>$ freq2

[1]Nouns in German are capitalized. This is normally dealt as a further “truecasing” postprocessing, and is an orthogonal problem from the one we deal with here.

```
# vowels (soft versus hard)
$word =  s/[aouå]/a/g;
$word =  s/[eiyäöé]/e/g;

# consonant combinations and
# spelling alternations
$word =  s/ng/N/g;
$word =  s/gn/G/g;
$word =  s/ck/K/g;
$word =  s/[lhgd]j/J/g;
$word =  s/^ge/Je/g;
$word =  s/^ske/Se/g;
$word =  s/^s[kt]?j/S/g;
$word =  s/^s?ch/S/g;
$word =  s/^tj/T/g;
$word =  s/^ke/Te/g;

#consonants grouping
$word =  s/[ptk]/p/g;
$word =  s/[bdg]/b/g;
$word =  s/[lvw]/l/g;
$word =  s/[cqxz]/q/g;
```

Figure 1: Transformations performed for normalizing Swedish consonants (Perl notation).

- word-1,word+1 as bigram vs compound
- word-1 as true prefix vs single word
- word+1 as true suffix vs single word

where -1 refers to the word before the merge point, and +1 to the word after.

We aimed to include features representing the knowledge available to the list and POS heuristics, by including part-of-speech tags and frequencies for compounds and bigrams, as well as a comparison between them. Features were also inspired by previous work on compound splitting, based on the intuition that features that are useful for splitting compounds, could also be useful for merging. Character n-grams has successfully been used for splitting Swedish compounds, as the only knowledge source by Brodda (1979), and as one of several knowledge sources by Sjöbergh and Kann (2004). Friberg (2007) tried to normalize letters, beside using the original letters. While she was not successful, we still believe in the potential of this feature. Larson et al. (2000), used frequencies of prefixes and suffixes from a corpus, as a basis of their method for splitting German compounds.

4.2.2 Training data for the sequence labeler

Since features are strongly lexicalized, a suitably large training dataset is required to prevent overfitting, ruling out the possibility of manual labelling.

We created our training data automatically, using the two heuristics described earlier, plus a third one enabled by the availability, when estimating parameters for the CRF, of a reference translation: merge if two tokens are observed combined in the reference translation (possibly as a sub-sequence of a longer word). We compared multiple alternative combinations of heuristics on a validation dataset. The validation and test data were created by applying all heuristics, and then manually check all positive annotations.

A first possibility to automatically generate a training dataset consists in applying the compound splitting preprocessing of choice to the target side of the parallel training corpus for the SMT system: separation points where merges should occur are thus trivially identified. In practice, however, merging decisions will need be taken on the noisy output of the SMT system, and not on the clean training data. To acquire training data that is similar to the test data, we could have held out from SMT training a large fraction of the training data, used the trained SMT to translate the source side of it, and then label decision points according to the heuristics. This would, however, imply making a large fraction of the data unavailable to training of the SMT. We thus settled for a compromise: we trained the SMT system on the whole training data, translated the whole source, then labeled decision points according to the heuristics. The translations we obtain are thus biased, of higher quality than those we should expect to obtain on unseen data. Nevertheless they are substantially more similar to what will be observed in operations than the reference translations.

5 Experiments

We performed experiments on translation from English into Swedish and Danish on two different corpora, an automotive corpus collected from a proprietary translation memory, and on Europarl (Koehn, 2005) for the merging experiments. We used factored translation (Koehn and Hoang, 2007), with both surface words and part-of-speech tags on the

	EU-Sv	Auto-Sv	Auto-Da
Corpus	Europarl	Automotive	Automotive
Languages	English→Swedish	English→Swedish	English→Danish
Compounds split	N, V, Adj	N, V, Adj	N
POS tag-sets	POS	POS,RPOS	RPOS
Decoder	Moses	in-house	in-house
Training sentences SMT	1,520,549	329,090	168,047
Training words SMT (target)	34,282,247	3,061,282	1,553,382
Training sentences CRF	248,808	317,398	164,702
Extra training sentences CRF	3,000	3,000	163,201

Table 2: Overview of the experimental settings

target side, with a sequence model on part-of-speech. We used two decoders, Matrax (Simard et al., 2005) and Moses (Koehn et al., 2007), both standard statistical phrase based decoders. For parameter optimization we used minimum error rate training (Och, 2003) with Moses and gradient ascent on smoothed NIST for the in-house decoder. In the merging experiments we used the CRF++ toolkit.[2]

Compounds were split before training using a corpus-based method (Koehn and Knight, 2003; Stymne, 2008). For each word we explored all possible segmentations into parts that had at least 3 characters, and choose the segmentation which had the highest arithmetic mean of frequencies for each part in the training corpus. We constrained the splitting based on part-of-speech by only allowing splitting options where the compound head had the same tag as the full word. The split compound parts kept their form, which can be special to compounds, and no symbols or other markup were added.

The experiment setup is summarized in Table 2. The extra training sentences for CRF are sentences that were not also used to train the SMT system. For tuning, test and validation data we used 1,000 sentence sets, except for Swedish auto, where we used 2,000 sentences for tuning. In the Swedish experiments we split nouns, adjectives and verbs, and used the full POS-set, except in the coalescence experiments where we compared the full and restricted POS-sets. For Danish we only split nouns, and used the restricted POS-set. For frequency calculations of compounds and compound parts that were needed for compound splitting and some of the compound merging strategies, we used the respective training data in all cases. Significance testing was performed using approximate randomization (Riezler and Maxwell, 2005), with 10,000 iterations, and $\alpha < 0.05$.

5.1 Experiments: Promoting compound coalescence

We performed experiments with factored translation models with the restricted part-of-speech set on the Danish and Swedish automotive corpus. In these experiments we compared the restricted part-of-speech set we suggest in this work to several baseline systems without any compound processing and with factored models using the extended part-of-speech set suggested by Stymne (2008). Compound parts were merged using the POS-based heuristic. Results are reported on two standard metrics, NIST (Doddington, 2002) and Bleu (Papineni et al., 2002), on lower-cased data. For all sequence models we use 3-grams.

Results on the two Automotive corpora are summarized in Table 3. The scores are very high, which is due to the fact that it is an easy domain with many repetitive sentence types. On the Danish dataset, we observe significant improvements in BLEU and NIST over the baseline for all methods where compounds were split before translation and merged afterwards. Some of the gain is already obtained using a language model on the extended part-of-speech set. Additional gains can however be obtained using instead a language model on a reduced set of POS-tags (RPOS), and with a count feature explicitly boosting desirable RPOS sequences. The count feature on undesirable sequences did not bring any

[2]Available at `http://crfpp.sourceforge.net/`

improvements over any of the systems with compound splitting.

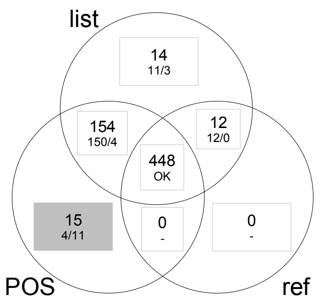
Results on the Swedish automotive corpus are less clear-cut than for Danish, with mostly insignificant differences between systems. The system with decomposition and a restricted part-of-speech model is significantly better on Bleu than all other systems, except the system with decomposition and a standard part-of-speech model. Not splitting actually gives the highest NIST score, even though the difference to the other systems is not significant, except for the system with a combination of a trained RPOS model and a boost model, which also has significantly lower Bleu score than the other systems with compound splitting.

5.2 Experiments: Compound merging

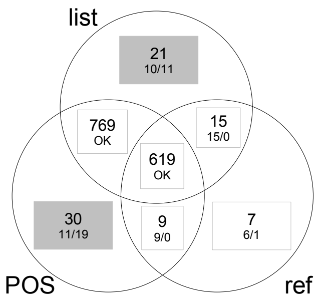
We compared alternative combinations of heuristics on our three validation datasets, see Figure 2. In order to estimate the amount of false negatives for all three heuristics, we inspected the first 100 sentences of each validation set, looking for words that should be merged, but were not marked by any of the heuristics. In no case we could find any such words, so we thus assume that between them, the heuristics can find the overwhelming majority of all compounds to be merged.

We conducted a round of preliminary experiments to identify the best combination of the heuristics available at training time (modified list-based, POS-based, and reference-based) to use to create automatically the training data for the CRF. Best results on the validation data are obtained by different combination of heuristics for the three datasets, as could be expected by the different distribution of errors in Figure 2. In the experiments below we trained the CRF using for each dataset the combination of heuristics corresponding to leaving out the grey portions of the Venn diagrams. This sort of preliminary optimization requires hand-labelling a certain amount of data. Based on our experiments, skipping this optimization and just using ref∨(list∧POS) (the optimal configuration for the Swedish-English Europarl corpus) seems to be a reasonable alternative.

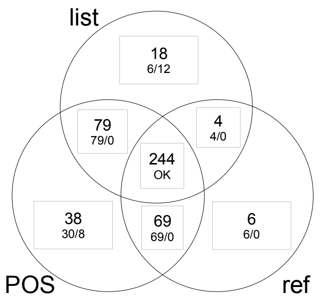
The validation data was also used to set a frequency cut-off for feature occurrences (set at 3 in the following experiments) and to tune the regularization parameter in the CRF objective function.



Automotive, Swedish



Europarl, Swedish



Automotive, Danish

Figure 2: Evaluation of the different heuristics on validation files from the three corpora. The number in each region of the Venn diagrams indicates the number of times a certain combination of heuristics fired (i.e. the number of positives for that combination). The two smaller numbers below indicate the number of true and false positive, respectively. Venn diagram regions corresponding to unreliable combinations of heuristics have corresponding figures on a grey background. OK means that a large fraction of the Venn cell was inspected, and no error was found.

		Danish auto		Swedish auto	
		BLEU	NIST	BLEU	NIST
No compound	Base	70.91	8.8816		
splitting	Base+POSLM	72.08	8.9338	56.79	**9.2674**
	POSLM	74.11*	9.2341*	57.28	9.1717
With	RPOSLM	74.26*	9.2767*	**58.12***	9.1694
compound	punish model	73.34*	9.1543*		
splitting	boost model	**74.96****	9.3028**	57.31	9.1736
	RPOSLM + boost	74.76**	**9.3368****	55.82	9.1088

Table 3: Results of experiments with methods for promoting coalescence. Compounds are merged based on the POS heuristic. Scores that are significantly better than Base+POSLM, are marked '*', and scores that are also better than POSLM with '**'.

Results are largely insensitive to variations in these hyper-parameters, especially to the CRF regularization parameter.

For the Danish auto corpus we had access to training data that were not also used to train the SMT system, that we used to compare the performance with that on the possibly biased training data that was also used to train the SMT system. There were no significant differences between the two types of training data on validation data, which confirmed that reusing the SMT training data for CRF training was a reasonable strategy.

The overall merging results of the heuristics, the best sequence labeler, and the sequence labeler without POS are shown in Table 4. Notice how the (modified) list and POS heuristics have complementary sets of false negatives: when merging on the OR of the two heuristics, the number of false negatives decreases drastically, in general compensating for the inevitable increase in false positives.

Among the heuristics, the combination of the improved list heuristic and the POS-based heuristic has a significantly higher recall and F-score than the POS-based heuristic alone in all cases except on the validation data for Swedish Auto, and than the list-based strategy in several cases. The list heuristic alone performs reasonably well on the two Swedish data sets, but has a very low recall on the Danish dataset. In all three cases the SMT training data has been used for the list used by the heuristic, so this is unexpected, especially considering the fact that the Danish dataset is in the same domain as one of the Swedish datasets. The Danish training data is smaller than the Swedish data though, which might be an influencing factor. It is possible that this heuristic could perform better also for Danish given more data for frequency calculations.

The sequence labeler is competitive with the heuristics; on F-score it is only significantly worse than any of the heuristics once, for Danish auto test data, and in several cases it has a significantly higher F-score than some of the heuristics. The sequence labeler has a higher precision, significantly so in three cases, than the best heuristic, the combination heuristic, which is positive, since erroneously merged compounds are usually more disturbing for a reader or post-editor than non-merged compounds.

The sequence-labelling approach can be used also in the absence of a POS tagger, which can be important if no such tool of suitable quality is available for the target language and the domain of interest. We thus also trained a CRF-based compound merger without using POS features, and without using the POS-based heuristic when constructing the training data. Compared to the CRF with access to POS-tags, on validation data F-score is significantly worse on the Europarl Swedish condition and the Automotive Danish condition, and are unchanged on Automotive Swedish. On test data there are no significant differences of the two sequence labelers on the two Automotive corpora. On Swedish Europarl, the CRF without POS has a higher recall at the cost of a lower precision. Compared to the list heuristic, which is the only other alternative strategy that works in the absence of a POS tagger, the CRF without POS performs significantly better on recall and F-score for Danish automotive, and mostly comparative on the two Swedish corpora.

	Validation data			Test data		
	Precision	Recall	F-score	Precision	Recall	F-score
	Swedish auto					
list	$.9889^{p,lp}$	$.9936^{p}$	$.9912^{p}$	.9900	.9770	.9835
POS	.9757	.9632	.9694	$.9916^{lp}$	.9737	.9826
list∨POS	.9720	1^{p}	$.9858^{p}$	.9822	$.9984^{l,p,c,cp}$	$.9902^{l,p,cp}$
CRF (ref∨list)	$.9873^{p,lp}$	$.9984^{p}$	$.9928^{p,lp}$	.9869	.9869	.9869
CRF without POS	$.9873^{p,lp}$	$.9968^{p}$	$.9920^{p,lp}$	.9836	.9852	.9844
	Swedish Europarl					
list	$.9923^{lp,c,cp}$	.9819	.9871	$.9882^{lp,cp}$	.9849	.9865
POS	$.9867^{lp}$	.9785	.9825	$.9893^{lp}$	.9751	.9822
list∨POS	.9795	$.9958^{l,p,c,cp}$	$.9876^{p,cp}$	.9782	$.9993^{l,p,c,cp}$	$.9886^{p,cp}$
CRF (ref∨(list∧POS))	$.9841^{cp}$	$.9916^{l,p}$	$.9879^{p,cp}$	$.9953^{l,p,lp,cp}$	.9790	$.9871^{p}$
CRF without POS	.9780	$.9882^{p}$	.9831	.9805	$.9882^{p,c}$	.9843
	Danish auto					
list	.9250	.7603	.8346	$.9905^{lp}$	.7640	.8626
POS	$.9814^{l,lp}$	$.9635^{l,cp}$	$.9724^{l,lp,cp}$	.9779	$.9294^{l}$	$.9538^{l}$
list∨POS	.9251	$.9863^{l,p,cp}$	$.9547^{l}$	.9760	$.9878^{l,p,c}$	$.9819^{l,p,c}$
CRF (ref∨list∨POS)	$.9775^{l,lp}$	$.9932^{l,p,cp}$	$.9853^{l,p,lp,cp}$	.9778	$.9659^{l,p}$	$.9718^{l,p}$
CRF without POS	$.9924^{l,lp,c}$	$.8973^{l}$	$.9424^{l}$	.9826	$.9635^{l,p}$	$.9729^{l,p}$

Table 4: Precision, Recall, and F-score for compound merging methods based on heuristics or sequence labelling on validation data and on held-out test data. The superscripts marks the systems that are significantly worse than the system in question (l-list, p-POS, lp-list∨POS, c-best CRF configuration, cp-CRF without POS).

The sequence labeler has the advantage over the heuristics that it is able to merge completely novel compounds, whereas the list strategy can only merge compounds that it has seen, and the POS-based strategy can create novel compounds, but only with known modifiers. An inspection of the test data showed that there were a few novel compounds merged by the sequence labeler that were not identified with either of the heuristics. In the test data we found *knap+start* (*button start*) and *vand+nedsænkning* (*water submersion*) in Danish Auto, and *kvarts sekel* (*quarter century*) and *bostad(s)+ersättning* (*housing grant*) in Swedish Europarl. This confirms that the sequence labeler, from automatically labeled data based on heuristics, can learn to merge new compounds that the heuristics themselves cannot find.

6 Discussion and conclusions

In this article, we described several methods for promoting coalescence and deciding if and how to merge word compounds that are either competitive with, or superior to, any currently known method.

For promoting compound coalescence we experimented with introducing additional LMs based on a restricted set of POS-tags, and with dedicated SMT model features counting the number of sequences known a priori to be desirable and undesirable. Experiments showed that this method can lead to large improvements over systems using no compound processing, and over previously known compound processing methods.

For merging, we improved an existing list-based heuristic, consisting in checking whether the first of two consecutive words has been observed in a corpus as a compound modifier and their combination has been observed as a compound, introducing the additional constraint that words are merged only if their corpus frequency as a compound is larger than their frequency as a bigram.

We observed that the false negatives of this improved list-based heuristic and of another, known, heuristic based on part-of-speech tags were complementary, and proposed a logical OR of them that generally improves over both.

We furthermore cast the compound merging problem as a sequence labelling problem, opening it to solutions based on a broad array of models and algorithms. We experimented with one model, Conditional Random Fields, designed a set of easily computed features reaching beyond the information accessed by the heuristics, and showed that it gives very competitive results.

Depending on the choice of the features, the sequence labelling approach has the potential to be truly productive, i.e. to form new compounds in an unrestricted way. This is for instance the case with the feature set we experimented with. The list-based heuristic is not productive: it can only form a compound if this was already observed as such. The POS-based heuristic presents some limited productivity. Since it uses special POS-tags for compound modifiers, it can form a compound provided its head has been seen alone or as a head, and its modifier(s) have been seen elsewhere, possibly separately, as modifier(s) of compounds. The sequence labelling approach can decide to merge two consecutive words even if neither was ever seen before in a compound.

In this paper we presented results on Swedish and Danish. We believe that the methods would work well also for other compounding languages such as German and Finnish. If the linguistic resources required to extract some of the features, e.g. a POS tagger, are unavailable (or are available only at training time but not in operations) for some language, the sequence-labelling method can still be applied. It is competitive or better than the list heuristic, which is the only heuristic available in that scenario.

Experiments on three datasets show that the improved and combined heuristics perform generally better than any already known method, and that, besides being fully productive, the sequence-labelling version is highly competitive, tends to generate fewer false positives than the combination heuristic, and can be used flexibly with limited or no linguistic resources.

References

Benny Brodda. 1979. Något om de svenska ordens fonotax och morfotax: Iakttagelse med utgångspunkt från experiment med automatisk morfologisk analys. In *PILUS nr 38*. Inst. för lingvistik, Stockholms universitet, Sweden.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurence statistics. In *Proceedings of the Second International Conference on Human Language Technology*, pages 228–231, San Diego, California, USA.

Chris Dyer. 2010. *A Formal Model of Ambiguity and its Applications in Machine Translation*. Ph.D. thesis, University of Maryland, USA.

İlknur Durgar El-Kahlout and Kemal Oflazer. 2006. Initial explorations in English to Turkish statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 7–14, New York City, New York, USA.

Alexander Fraser. 2009. Experiments in morphosyntactic processing for translating to and from German. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 115–119, Athens, Greece.

Karin Friberg. 2007. Decomposing Swedish compounds using memory-based learning. In *Proceedings of the 16th Nordic Conference on Computational Linguistics (Nodalida'07)*, pages 224–230, Tartu, Estonia.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 868–876, Prague, Czech Republic.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th Conference of the EACL*, pages 187–193, Budapest, Hungary.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL, demonstration session*, pages 177–180, Prague, Czech Republic.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit X*, pages 79–86, Phuket, Thailand.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, Williamstown, MA.

Martha Larson, Daniel Willett, Joachim Köhler, and Gerhard Rigoll. 2000. Compound splitting and lexical unit recombination for improved performance of a speech recognition system for German parliamentary speeches. In *Proceedings of the Sixth International Conference on Spoken Language Processing*, volume 3, pages 945–948, Beijing, China, October.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Maja Popović, Daniel Stein, and Hermann Ney. 2006. Statistical machine translation of German compound words. In *Proceedings of FinTAL – 5th International Conference on Natural Language Processing*, pages 616–624, Turku, Finland. Springer Verlag, LNCS.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at ACL'05*, pages 57–64, Ann Arbor, Michigan, USA.

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. 2005. Translating with non-contiguous phrases. In *Proceedings of the Human Language Technology Conference and the conference on Empirical Methods in Natural Language Processing*, pages 755–762, Vancouver, British Columbia, Canada.

Jonas Sjöbergh and Viggo Kann. 2004. Finding the correct interpretation of Swedish compounds, a statistical approach. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

Sara Stymne and Maria Holmqvist. 2008. Processing of Swedish compounds for phrase-based statistical machine translation. In *Proceedings of the 12th Annual Conference of the European Association for Machine Translation*, pages 180–189, Hamburg, Germany.

Sara Stymne. 2008. German compounds in factored statistical machine translation. In *Proceedings of GoTAL – 6th International Conference on Natural Language Processing*, pages 464–475, Gothenburg, Sweden. Springer Verlag, LNCS/LNAI.

Sara Stymne. 2009a. A comparison of merging strategies for translation of German compounds. In *Proceedings of the EACL 2009 Student Research Workshop*, pages 61–69, Athens, Greece.

Sara Stymne. 2009b. *Compound processing for phrase-based statistical machine translation*. Licentiate thesis, Linköping University, Sweden.

Sami Virpioja, Jaako J. Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In *Proceedings of MT Summit XI*, pages 491–498, Copenhagen, Denmark.