Generation of Compound Words in Statistical Machine Translation into Compounding Languages

Sara Stymne
Uppsala University*

Nicola Cancedda
Xerox Research Centre Europe**

Lars Ahrenberg
Linköping University†

In this article we investigate statistical machine translation (SMT) into Germanic languages, with a focus on compound processing. Our main goal is to enable the generation of novel compounds that have not been seen in the training data. We adopt a split-merge strategy, where compounds are split before training the SMT system, and merged after the translation step. This approach reduces sparsity in the training data, but runs the risk of placing translations of compound parts in non-consecutive positions. It also requires a postprocessing step of compound merging, where compounds are reconstructed in the translation 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 and show that it can lead to improvements both by direct inspection and in terms of standard translation evaluation metrics. We also propose several new methods for compound merging, based on heuristics and machine learning, which outperform previously suggested algorithms. These methods can produce novel compounds and a translation with at least the same overall quality as the baseline. For all subtasks we show that it is useful to include part-of-speech based information in the translation process, in order to handle compounds.

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

* Department of Linguistics and Philology, Uppsala University, Box 635, 751 26 Uppsala, Sweden. E-mail: sara.stymne@lingfil.uu.se.
** Xerox Research Centre Europe, 6 chemin de Maupertuis, 38240 Meylan, France. E-mail: nicola.cancedda@xrce.xerox.com.
† Department of Computer and Information Science, Linköping University, 58183 Linköping, Sweden. E-mail: lars.ahrenberg@liu.se.

Submission received: 25 April 2012; revised submission received: 30 November 2012; accepted for publication: 8 January 2013.

doi:10.1162/COLI_a_00162

productively. Closed compounds are written as single words without spaces or other word boundaries, as in German *Goldring*. We will refer to these languages as **compounding languages**. In English, on the other hand, compounds are generally open, that is, written as two words as in *gold ring*.

This difference in compound orthography leads to problems for statistical machine translation (SMT). 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, systems often produce separate words, genitive or other alternative constructions, or translate only one part of the compound. For an SMT system 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.

Previous work (e.g., Koehn and Knight 2003) has shown that compound splitting improves translation from compounding languages into English. In this article we explore several aspects of the less-researched area of compound treatment for translation into such languages, using three Germanic languages (German, Swedish, and Danish) as examples.[1] The assumption is that splitting compounds will also improve translation for this translation direction and lead to more natural translations. The strategy we adopt is to split compounds in the training data, and to merge them in the translation output. Our overall goal is to improve translation quality by productively generating compounds in SMT systems.

The main contributions of the article are as follows:

- Demonstrating improved **coalescence** (adjacency and order) of compound parts in translation through the use of sequence models based on customized part-of-speech sets and count features
- Designing and evaluating several heuristic methods for compound merging that outperforms previous heuristic merging methods
- Designing and evaluating a novel method for compound merging based on sequence labeling
- Demonstrating the ability of these merging methods to generate novel unseen compounds

In addition, we report effects on translation performance from a number of variations in the methods for compound splitting and merging.

The rest of the article is structured as follows. Section 2 gives an overview of compound formation in the three target languages used in this work. Section 3 reports related work on compound processing for machine translation. Section 4 describes the compound processing strategy we use and Section 5 describes compound splitting. Section 6 addresses compound coalescence, followed by compound merging in Section 7. In Section 8 we present experimental results and in Section 9 we state our conclusions.

1 This work is a synthesis and extension of Stymne (2008); Stymne and Holmqvist (2008); Stymne, Holmqvist, and Ahrenberg (2008); Stymne (2009); and Stymne and Cancedda (2011).

2. Closed Compounds in German, Swedish, and Danish

Compounds in German, Swedish, and Danish are generally closed, written without word boundaries, as exemplified for German in Example (1). Compounds can be made up of two (1a) or more (1b) parts where parts may also be coordinated (1c). In a few cases compounds are written with a hyphen (1d), often when one of the parts is a proper name or an abbreviation. Most compounds are nouns (1a–1e), but they can also be adjectives (1f), verbs (1g), and adverbs (1h). English translations of compounds can be written as open compounds with separate words (1a) or with hyphens (1f), as other constructions, possibly with inserted function words (1g) and reordering (1b), or as single words (1e). Generally the last part of the compound is the **compound head**, that is, it conveys the main meaning of the compound, and determines its part of speech. The other parts, **compound modifiers**, modify the meaning of the compound head in some way and need not have the same part of speech as the full compound.

(1) a. Regierungskonferenz *intergovernmental conference*
Regierung+Konferenz *government conference*

b. Friedensnobelpreisträger *Nobel Peace Prize laureate*
Frieden+Nobel+Preis+Träger *peace Nobel prize bearer*

c. See- und Binnenhäfen *sea and inland ports*
See- und Binnen+Häfen *sea and interior ports*

d. EU-Mitgliedstaaten *EU member states*
EU-Mitglied+Staaten *EU member states*

e. Jahrtausend *millennium*
Jahr+tausend *year thousand*

f. dunkelblau *dark-blue*
dunkel+blau *dark blue*

g. kennenlernen *get to know*
kennen+lernen *know learn*

h. grösstenteils *in most instances*
grössten+teils *largest partly*

Compound modifiers often have a special form, such as the addition of an "s" to the base form of *Regierung* in Example (1a). We will refer to these form variants as **compounding forms**. Table 1 exemplifies the type of operations used in Swedish, German, and Danish to form compounding forms. There are many more alternative compounding forms in Swedish and German than in Danish. For an overview of the possible compounding forms in German see Langer (1998) or Kürschner (2003), in Danish see Kürschner (2003), and in Swedish see Thorell (1981) or Stymne and Holmqvist (2008). Some compounding forms coincide with paradigmatic forms, such as German *Jahres* that can also be genitive, and *Stadien* that can also be plural, from Table 1. There are different views on whether to treat these forms as paradigmatic forms or as compounding forms. We follow Langer (1998) in viewing them as compounding forms, because they often do not correspond to plural or possessive semantics. Many individual compound modifiers have more than one possible compounding form. In Example (2) we provide examples of several possible forms of the modifier *Kind* (*child*) in German compounds.

(2) *0* Kind+phase (*child-caring period*)
+s Kinds+lage (*fetal position*)
+es Kindes+unterhalt (*child support*)
+er Kinder+film (*children's film*)
+- Ein-Kind-Politik (*one-child policy*)

In some cases concatenating two words would lead to three identical consecutive consonants. In the Scandinavian languages, there is a spelling rule that does not allow this, and three identical consonants are reduced to two, as in Example (3^{SV}). This spelling rule was also used for some German compounds before 1996, when it was changed by a spelling reform, so that nowadays three identical consecutive consonants are never reduced to two at compound boundaries in German (Institut für Deutsche Sprache 1998), as shown in Example (3^{DE}).

Table 1
Operations used for forming compounding forms with examples.

Type	Examples	
Null operation	DE *0*	umweltfreundlich (*environmentally-friendly*) Umwelt+freundlich (*environment friendly*)
	SV *0*	naturkatastrof (*natural disaster*) natur+katastrof (*nature disaster*)
	DA *0*	håndbagage (*hand luggage*) hånd+bagage (*hand luggage*)
Addition	DE *+es*	Jahreswechsel (*turn of the year*) Jahr+Wechsel (*year change*)
	SV *+s*	kvalitetstecken (*quality mark*) kvalitet+tecken(*quality sign*)
	DA *+e*	spillekonsol (*game console*) spill+konsol(*game console*)
Deletion	DE *-e*	Lymphreaktion (*lymphatic response*) Lymphe+Reaktion (*lymph response*)
	SV *-a*	flickskola (*girls' school*) flicka+skola (*girl school*)
Combination	DE *-on/+en*	Stadienexperte(*stadium expert*) Stadion+Experte (*stadium expert*)
	SV *-e/+s*	arbetsolycka (*industrial accident*) arbete+olycka (*work accident*)
	DA *-e/+s*	embedsmand (*civil servant*) embede+mand(*job man*)
Umlaut	DE *″+er*	Völkerrecht(*international law*) Volk+Recht(*people right*)
	SV *″-er/+ra*	brödrakärlek (*brotherly love*) broder+kärlek (*brother love*)
	DA *″+e*	børnesko (*children's shoe*) barn+sko (*child shoe*)

(3) SV tullagstiftning *customs legislation*
tull+lagstiftning *custom legislation*

DE Zelllinie *cell line*
Zell+Linie *cell line*

Compounding is common and productive; new compounds can be readily formed and understood. This is confirmed in a number of corpus studies. In German, compounds have been shown to make up 5–7% of tokens and 43–47% of types in news text (Baroni, Matiasek, and Trost 2002; Schiller 2005). If function words are removed, an even higher number of the tokens are compounds; in both Swedish and German 10% of the content words in a news text have been found to be compounds (Hedlund 2002). That compounding is productive means that it is likely that a high number of compounds have a very low frequency in texts. Baroni, Matiasek, and Trost (2002) found that 83% of the compounds in a large German news corpus occur less than five times. In Swedish, compounds are the most common type of hapax words, that is, words that occur only once in a text (Carlberger et al. 2005). The most common type of compound is the noun+noun compound, which makes up 62% of the compounds in the German news corpus of Baroni, Matiasek, and Trost.

3. Related Work

The problems arising from differences in compounding strategies in translation from German into English have been addressed by several authors. The most common architecture for translation from German is to split compounds in a preprocessing step prior to training and translation using some automatic method, which has been suggested both for SMT (Nießen and Ney 2000; Koehn and Knight 2003; Popović, Stein, and Ney 2006; Holmqvist, Stymne, and Ahrenberg 2007) and example-based MT (Brown 2002). German compounds are split into their component parts in a preprocessing step and the translation model is then trained between modified German and English. At translation time, the German source text is also run through a compound splitter. In the studies cited here, only one splitting option is given as input to the decoder, which can be problematic in case the splitting is wrong, or if any of the parts are unknown. In Dyer (2009) several splitting options were given to the decoder in the form of a lattice. It is, however, not straightforward to use lattices during training, and in order to solve this, the training corpus was doubled, one part being without splits and the other part having the best splitting option for each word.

For translation into German, Popović, Stein, and Ney (2006) investigated three different strategies for compound processing. The first was to split compounds during training and after translation merge compound parts back into full compounds, the second merged English compounds prior to training instead of splitting German compounds, and the third used compound splitting only to improve word alignment. The split–merge strategy gave the best results, but using splitting only for alignment gave similar results. The merging of English compounds led to an improvement over a baseline without compound processing, but was not as good as the other two strategies. Popović, Stein, and Ney also presented one of few previous suggestions for compound merging. Each word in the translation output was looked up in a list of compound parts, and merged with the next word if it resulted in a known compound. This method led to improved overall translation results from English to German. The drawback of this method is that novel compounds cannot be merged. It might also merge words that should not be merged, but that happen to coincide with known compounds.

Fraser (2009) merged split German compounds after translation from English, by applying a second phrase-based SMT (PBSMT) system trained on German with split compounds and normal German. No separate results were presented for this extension alone, but in combination with other morphological processing the strategy led to worse results than the baseline. This method also suffers from the same drawbacks as that of Popović, Stein, and Ney (2006), that novel compounds cannot be merged and words that should not be merged can still be.

Koehn, Arun, and Hoang (2008) discussed the treatment of hyphenated compounds for translation into German. They used a separate mark-up token for hyphened compounds, where the hyphen was split into a separate token, and marked by a symbol. The impact on the translation result was small.

Botha, Dyer, and Blunsom (2012) discussed the approach of using customized language models to target German compounds. They presented hierarchical Pitman-Yor language models, where the compound head is conditioned on the words preceding the full compound, and the compound modifiers are modeled by a reverse compound language model. They used this model as a replacement of a standard language model for SMT and found that although the perplexity of the language model was reduced, there were only minor improvements on Bleu for the SMT task. In this case the SMT pipeline was left unchanged, meaning that novel compounds were not considered.

Compound merging has also been performed for speech recognition. An example of this is Berton, Fetter, and Regel-Brietzmann (1996), who extended the word graphs output by a German speech recognizer with possible compounds by combining edges of words during a lexical search. The final hypotheses were then identified from the graph using dynamic programming techniques. Compound merging for speech recognition is a somewhat different problem than for machine translation, however, because coalescence is not an issue, as compared with SMT, where there is no guarantee that the order of the parts in the translation output is correct.

Another somewhat related problem to compound merging is that of detection of erroneously split compounds in human text, which is faced by grammar checkers. Writing compounds as separate words, with spaces between parts, is a common writing error in compounding languages. Carlberger et al. (2005) described a system for Swedish that used handwritten rules to identify, among other errors, erroneously split compounds. The rules used parts of speech and morphological features. On a classified gold standard of writing errors they had a recall of 46% and a precision of 39% for identifying split compounds, indicating that it is a difficult problem to find split compounds in free, unmarked text.

There is also work on morphological merging, which is needed when the target words have been split into morphs in the training corpus. Virpioja et al. (2007) marked morphs with a symbol and merged all marked words with the next word for translation between Finnish, Swedish, and Danish, without showing any improvements over an unmarked baseline. This strategy does have the advantage of being able to merge novel word forms, but has a drawback in that it can merge parts into non-words if the parts are misplaced in the translation output.

El-Kahlout and Oflazer (2006) used a similar symbol-based merging strategy for translation from English into Turkish, but with the addition of morphographemic rules. They had positive results when performing limited splitting and grouping the split morphs, but not when splitting all morphs. They reported that there were problems with the order of morphs in the output. Badr, Zbib, and Glass (2008) reported results for translation from English to Arabic, where they used a combination merging method

where forms were picked from the corpus for known combinations of morphs and words, and generated based on handwritten recombination rules otherwise, which led to improvements over the baseline. They also used the morphs+POS as factors in a factored translation model (Koehn and Hoang 2007) where surface forms were generated from this information; this gave a small improvement on a large corpus, but at the cost of high runtime. El Kholy and Habash (2010) extended the merging scheme of Badr, Zbib, and Glass (2008) by using the conditional probability and a language model score to pick the best known merging option. They showed a small effect of this on an MT task, even though this strategy was the best option in an intrinsic evaluation based on human reference translations.

Compound splitting has been addressed in many articles, as a separate task (Schiller 2005) or targeted for applications such as information retrieval (Holz and Biemann 2008), speech recognition (Larson et al. 2000), grammar checking (Sjöbergh and Kann 2004), lexicon acquisition (Kokkinakis 2001), word prediction (Baroni, Matiasek, and Trost 2002), and machine translation (Koehn and Knight 2003; Dyer 2009; Fritzinger and Fraser 2010; Macherey et al. 2011).

The most successful strategies that address compound processing for MT apply compound splitting as a preprocessing step before training the translation models. Koehn and Knight (2003) presented an empirical splitting algorithm targeted at SMT from German to English. They split words in all possible places, and considered a splitting option valid if all its parts had been seen as words in a monolingual corpus. They allowed the addition of *-s* or *-es* at all splitting points. If there were several valid splitting options they chose one based on the number of splits, the geometric mean of part frequencies, or based on alignment data. They evaluated the splitting algorithms intrinsically on a gold standard of manually split noun phrases and on machine translation of noun phrases. The best results for PBSMT were achieved by using either the geometric mean, or the highest number of splits. There were no correlations between translation results and the intrinsic evaluation.

Several other researchers have also explored compound splitting for translation from German to English. Nießen and Ney (2000) used a morpho-syntactic analyzer for splitting German compounds prior to translation. Popović, Stein, and Ney (2006) used the geometric mean version from Koehn and Knight (2003) as well as the morphosyntactic algorithm from Nießen and Ney (2000) for splitting, with similar positive results for both options. Fritzinger and Fraser (2010) combined linguistic analysis with corpus-driven scoring and showed an improvement compared to using only a corpus-driven approach. Macherey et al. (2011) described a corpus-driven method that learns compounding form transformations of a language in addition to just compound splitting, and showed an improvement for translation from several languages into English compared to a baseline without compound treatment. Dyer (2010) suggested a compound splitting method based on sequence labeling, which gave good results for lattice-based translation from German.

4. Compound Translation

For translation into a compounding language, we adopt the compound processing strategy suggested by Popović, Stein, and Ney (2006). The process is:

1. Split compounds on the target (compounding language) side of the training corpus.

2. Learn a translation model from source (e.g., English) into decomposed-target (e.g., decomposed-German).
3. At translation time, translate using the learned model from source into decomposed-target.
4. Apply a postprocessing **merge** step to reconstruct compounds.

The merging step must solve two problems: Identify which words should be merged into compounds, and choose the correct compounding form for the compound modifiers. The first 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 coalescence needs to be addressed, that is, translations where compound elements are correctly positioned should be promoted.

Figure 1 gives an overview of the translation process with compound processing. Compounds are split before training the translation system, and merged after translation. We use factored decoding (Koehn and Hoang 2007), where features other than just surface words can be used by the system. In our case we tag the data with parts of speech and use a factored model with POS-tags on the target side, which allows us to have a POS-sequence model, beside the standard language model. This configuration has a very small overhead compared with decoding without factors. As we show, using part-of-speech tags on the target side helps to improve the coalescence of compound parts, and can be used to guide the merging process.

For the tuning step there are two options: either we can split the development set and tune with a translation with split compounds compared with a reference with split

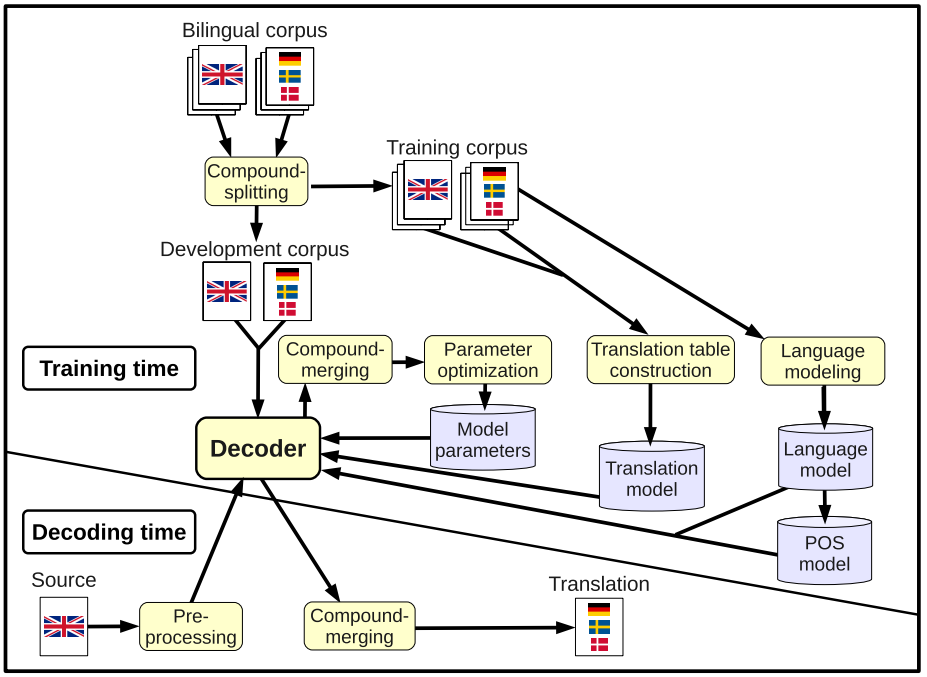


Figure 1
The SMT system architecture.

compounds, or we can merge compounds in the translation output, before performing the optimization. We found empirically that we got the best results using the second approach, of performing compound merging during the tuning process. When we tuned on split texts, the results were more unstable, and especially the number of words in the translation output varied substantially. We thus use merging also during the tuning process in all experiments, as shown in Figure 1.

5. Compound Splitting

Our method for compound splitting is based on Koehn and Knight (2003) and Stymne (2008). For each word all possible segmentations are explored, with the restrictions that all parts must have at least three characters, and the last part, the compound head, must have the same part-of-speech tag as the word itself. Hyphens are treated as additions to compound modifiers, just as *+s* or *+e*. Segmentations are scored with the arithmetic mean of frequencies for each part in the training corpus and the segmentation with the highest score is chosen.

We have investigated several variants of the basic method and their effects on compound translation. The variants investigated are:

- Using geometric or arithmetic mean for choosing the highest frequency candidate split
- Restricting the length of each split part, either to three or four characters
- Restricting the highest number of parts per compound to two, or allowing any number of compound parts
- Allowing all known compounding forms (c.f. Table 1), or restricting them to the most common ones
- Restricting the last part of the compound to be known from the corpus with the same part-of-speech tag as the full compound, or allowing all possible parts-of-speech tags

6. Promoting Coalescence of Compounds

In this section we describe our approach to improve the coalescence of compounds, which is based on POS-sequence models. We first present the representation schemes we use for compounds, which form the basis of the sequence model approach, and then describe the sequence modeling approach in more detail. Finally, we discuss how POS-tags can be used for count features.

6.1 Representation of Compound Parts

As a result of compound splitting the segmentation of words is changed, and we know from Table 1 that compounding forms often do not coincide with any forms that can be used as standalone words. This raises several design decisions:

1. Should alternative forms of the same compound modifier be normalized to a canonical form?

2. Should compound modifiers be marked with a special symbol? Or should the separation between compound parts be marked?
3. How should compound parts be tagged in a factored system?

Although apparently innocuous, these decisions do have some influence on the whole process. In this work we have used three combinations of marking and normalization, and three different tagsets.

In Example (4a), called the **unmarked** scheme, compound modifiers are normalized (*kamps->kamp*) and the words carry no special marking. In Example (4b), called the **marked** scheme, compound modifiers are not normalized, but they are marked with the symbol "#". For these two marking schemes, an extended tagset, **EPOS**, is used. It contains tags for compound modifiers that also indicate the part of speech of the head word, such as "N-modif" if the head is a noun (N) or "ADJ-modif" if the head is an adjective (ADJ).

In Example (4c), called the **sepmarked** scheme, the split itself is marked, by using the special token "@#@" to mark split points, and compound modifiers are normalized. In this case we use a standard POS-tagset with the addition of a COMP-tag for the inserted split-token, and use the POS-tags that were found for the compound modifiers when they were looked up in the monolingual corpus during splitting. We call this tagset the **SPOS**-tagset.

(4) a. fem+kamps+seger:N (*pentathlon (five battle) victory*)
fem:N-modif kamp:N-modif seger:N

b. fem+kamps+seger:N (*pentathlon (five battle) victory*)
fem#:N-modif kamps#:N-modif seger:N

c. fem+kamps+seger:N (*pentathlon (five battle) victory*)
fem:NUM @#@:COMP kamp:N @#@:COMP seger:N

As a third alternative tagset we use a modified variant of the EPOS-tagset, where distinctions among parts of speech that are not relevant to the formation of compounds are blurred. This reduces the tagset to only a few tags, as in the case where only nouns are split:

- N-modif – 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

We call this tagset the reduced POS-tagset (**RPOS**). The RPOS-tagset could easily be extended to other types of compounds—for example, by extending it to five tags by also including ADJ and ADJ-modif if we want to split adjectives as well. Using the RPOS-tagset, the POS-based sequence model will only be useful for controlling the order and form of compound parts. With the EPOS-tagset it also aids in controlling the order of other words. All POS-based sequence models are trained on POS-tagged data, where the tags have been modified after applying a compound splitting algorithm. When referring to either of the three tagsets EPOS, RPOS, or SPOS, we will use the designation ***POS**.

6.2 Part-of-Speech–Based Sequence Models

If compounds are split in the training data, 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 sequence models on the customized part-of-speech sets. These sequence models can be either standard language models trained on texts where split compound modifiers are marked with symbols, or they can be POS-sequence models trained on the specially designed tagsets, such as EPOS. Both these types of models can encourage compound parts to occur in the correct order; the former, however, is a more lightweight approach, because it relies on surface words.

A language model solution to compound coalescence could be viewed as a soft constraint in the decoder, which encourages good sequences of compound parts over bad sequences. An alternative to this would have been a hard constraint in the decoder that prohibits compound parts to be placed in an incorrect order. We opted for a soft constraint approach because we found that it gave sufficiently good results, and because previous work on soft versus hard constraints for other areas has shown that soft constraints give more stable and generally better results, for instance, for phrase cohesion (Cherry 2008).

As described in the previous section, we can add special POS tags and symbols to identify compound modifiers. Table 2 shows examples of the different representation schemes. We can then train a *POS *n*-gram sequence model using any of the *POS-tagsets, which naturally steers the decoder towards translations with good relative placement of these components.

A lightweight model that gives some additional information of the order of compound parts compared to a language model on unmarked data is to train language models on texts where compounds are represented either using the marked or sepmarked schemes but without parts-of-speech tags. These models are, however, not as strong as the *POS-based models, since they cannot generalize from surface words, and mainly can aid in keeping known compounds together.

6.3 Sequence Models as Count Features

We expect a *POS-based *n*-gram sequence model 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

Table 2
Examples of representation schemes for the German phrase *die Fremdsprachenkenntnisse* [*the knowledge of foreign languages* / foreign language knowledge], originally tagged as *DET N(oun)*.

Original	die	Fremdsprachenkenntnisse				
Unmarked	die	fremd		sprache		kenntnisse
Marked	die	fremd#		sprachen#		kenntnisse
Sepmarked	die	fremd	@#@	sprache	@#@	kenntnisse
EPOS	DET	N-Modif		N-Modif		N
RPOS	X	N-Modif		N-Modif		N
SPOS	DET	ADJ	COMP	N	COMP	N

Table 3
Tag combinations in the translation output.

Combination	Judgment	Boost	Punish
N-Modif N	Good	1	0
N-Modif N-Modif	Good	1	0
N-Modif </s>	Bad	0	1
N-Modif X	Bad	0	1
all other combinations	Neutral	0	0

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 RPOS-sequences that we deem good and bad in the translation output. Table 3 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.

We define two new feature functions: the **boost model** counting the number of occurrences of Good sequences, and the **punish model**, counting the occurrences of Bad sequences, as indicated in Table 3. In the punish model we want to punish the two bad combinations, where compound modifiers are placed in isolation without a suitable head. In the boost model, we want to support the formation of compounds by rewarding the two good combinations. We definitely want to boost the combination of a compound part with its head. In addition, we can boost the formation of long compounds by boosting the combination of two compound modifiers as well. The boost and punish models can be used either in isolation or combined, with or without a further *POS *n*-gram sequence model.

To exemplify the two models, consider the translation hypothesis given in Example (5). The word pairs *skogs bruks* and *bruks plan*, marked in bold in the example, constitute good sequences of a modifier followed by another modifier or a head, and would give the count 2 with a boost model. The word sequence in italics, *skogs av*, constitutes a bad sequence, since the preposition *av* (*of*) cannot be a compound head, and would give the count 1 to a punish model. The other word sequences do not influence either of these models because they do not involve any compound modifiers.

(5) En|X **skogs**|**N-Modif bruks**|**N-Modif plan**|**N** ger|X en|X översikt|N *skogs*|*N-Modif av*|*X*
A forest cultivation plan gives an overview forest of

7. Compound Merging

Once a translation is generated using a system trained on split compounds, a post-processing step is required to merge components back into compounds. All methods we are aware of only consider consecutive tokens for merging: We stick to this assumption, having delegated to the methods described earlier the task to promote a good relative positioning of component translations. 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 such as, for example, inserting *-s* between components.

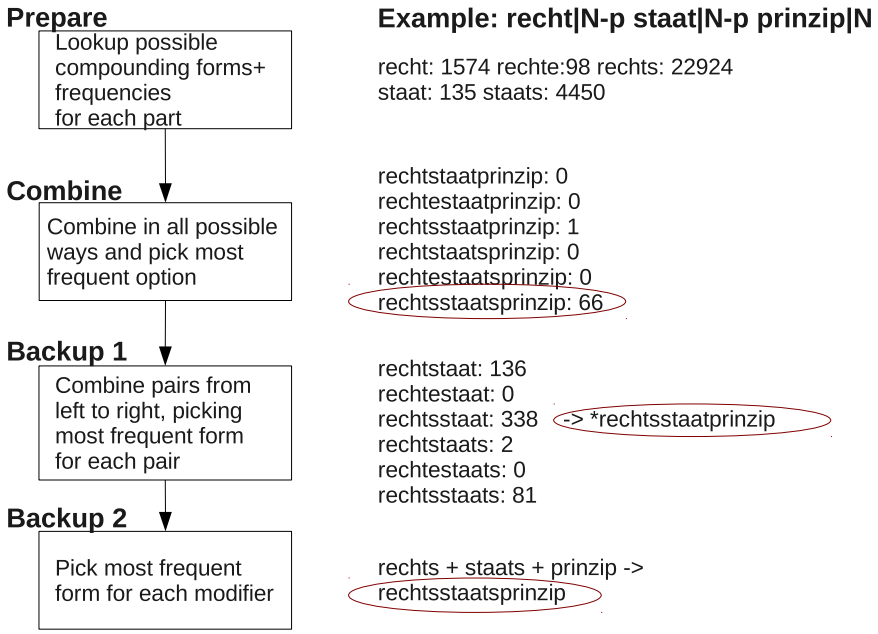


Figure 2
Overview of the reverse normalization algorithm, exemplified using the compound *Rechtsstaatsprinzip* [*rule-of-law principle* / right-state principle]. The chosen option for each strategy is circled.

In this section we first describe reverse normalization, then we describe our new heuristic merging methods and modifications to existing heuristics. Finally we describe a novel sequence-labeling formulation for compound merging.

7.1 Reverse Normalization

For compound modifiers that were normalized in the training data the reverse process (reverse normalization) is needed at merging time to recreate the correct form for the specific compound. We designed a method for reverse normalization that is based on corpus frequencies of compound modifiers and compounds that can be collected during compound splitting.

The strategy is illustrated and exemplified in Figure 2.[2] First we look up all known compounding forms, and their frequencies for all the compound modifiers. In Figure 2, for instance, we found three options for the word *recht* [*right*], with *rechts* being the most common option. Next, we try all combinations of forms to form a full compound, and if any matches are found we pick the most frequent match. If this fails we have two back-up strategies. First, we try to find known compounding forms for pairs of parts, starting from left to right, looking up frequencies of the resulting forms from adding two parts. If that fails we use our second back-up strategy, which is to use the most common form of each modifier, and concatenate those. For binary compounds, the second strategy is

2 Nouns in German are capitalized. This is normally dealt with as further recasing postprocessing, and is an orthogonal problem from the one we deal with here.

superfluous, because there is only one pair of compound parts, and the second strategy is always used.

7.2 Heuristic Approaches to Compound Merging

We have investigated three major approaches to heuristic compound merging. Our major contribution is a novel merging method based on part-of-speech matching. We contrast this method with adaptations of previous merging suggestions based on symbols and word lists. We also suggest improvements to these methods, and combination methods that combine the strengths of word lists and parts of speech.

7.2.1 POS-Based Merging. The POS-match algorithm uses the fact that it is possible to have several output factors besides surface form in a factored translation system. It merges words that are marked as compound modifiers in either the EPOS or RPOS tagsets if the next POS-tag matches. As described in Section 6.1, the part of speech of a compound modifier is based on the part of speech of its head word, so a word is considered matching if the next word is a compound modifier of the same type, or a head with a matching part of speech. In addition, if the next word does not match, the modifier could be part of a coordinated compound, which is checked by seeing if the next word is a conjunction, in which case a hyphen is added to the modifier. We investigated versions of the algorithm both with and without treatment of coordinated compounds.

If a compound modifier is followed by anything other than a matching part of speech or a conjunction it has most likely been misplaced in the translation process. These items are left as they are in the translation output, which is often fine, because only compound parts that occur as separate words in a corpus are split. When two matching compound modifiers are merged, the process is iterated to see if the next word is a matching compound modifier, head, or conjunction. This allows compounds with an arbitrary number of parts to be merged.

In summary, the POS-based merging algorithm has the following steps:

- Step through each word+POS pair from left to right[3]
- If a compound-POS, X-MODIF, is found:
  - Remove mark-up of the part if present
  - Store the compound part
  - When the next POS is a matching part, X-MODIF:
    Remove mark-up of the part if present
    Store the compound part
  - If the next POS is a matching head, X:
    Store the compound head
  - If at least two parts have been found (either several modifiers or a head):
    Perform reverse normalization on the stored parts if parts are normalized
    Merge all parts
    For Swedish and Danish: remove a consonant if any of the merges resulted in three identical consecutive consonants

3 The words that are processed in the inner if-clause are skipped in the outer loop.

– If the next POS is a conjunction and no head was found:
  Add a hyphen at the end of the compound part

This heuristic has the advantage over previous merging algorithms that it can form novel compounds while reducing the risk of erroneous merging through the matching constraint. It is also the only merging method we are aware of that addresses coordinated compounds. However, it requires a factored decoder that can carry part-of-speech tags through the translation process. It also requires tagsets where compound modifiers are marked based on the compound head, such as EPOS or RPOS. In the current form the POS-match strategy cannot be used for the SPOS-tagset that does not use head-based tags for compound modifiers, because we cannot enforce the matching constraint in this way. It would be possible to design a POS-match strategy based on standard tags, which for instance allowed merging of noun+noun, but not of noun+preposition. Such a strategy requires linguistic knowledge and customization for each language, however.

7.2.2 Other Merging Heuristics. The symbol-based method is inspired by work on morphology merging (El-Kahlout and Oflazer 2006; Virpioja et al. 2007). It merges words that are marked with a symbol with the next word in the marked scheme. In the sepmarked scheme, when a standard symbol is found, the words on both sides of it are merged. These algorithms have the disadvantage, compared with the POS-match algorithm, that it is more likely that words are merged into non-compounds, since no matching check is carried out.

We have also investigated methods based on word lists, proposed by Popović, Stein, and Ney (2006). These methods use frequency word lists compiled at split time. Three types of lists were used: lists of compound modifiers, of compounds, and of words. If a compound modifier is encountered, it is checked whether merging it with the next word results in either another compound modifier, or a compound or word. Unlike Popović, Stein, and Ney (2006), we perform this process recursively, to allow compounds with several parts. Again, reverse normalization is performed when needed. Still, no novel compounds can be formed, and coordinated compounds are not handled. The method does not merge words into non-words, but there is another risk, that of merging words that should be separate in a specific context, but that happen to form a valid compound or other word when combined, such as the examples in Example (6). It has the advantage over the other proposed methods that it can be used on output from any MT decoder, without the use of customized POS-tags or symbols in the output. We investigate the usage of two types of frequency lists for this strategy, either a list of compounds, which can be created as a byproduct of the splitting algorithm, like Popović, Stein, and Ney (2006), or by using a list of all words in a corpus.

(6) DE bei der (*at the*)
beider (*both*)

SV för små (*too small*)
försmå (*spurn*)

We empirically verified that the list-based heuristics tend to misfire quite often, leading to too many compounds, such as merging the words in Example (6). We thus modified them in two ways: (1) by additionally requiring the head word to be a content word and (2) by requiring the generated compound to be more frequent in a corpus than the corresponding bigram of isolated words. The head word restriction can block some erroneous merges such as those in Example (6^{DE}), whereas the frequency restriction can

potentially block both examples in Example (6). Compound and bigram frequencies can be computed on any available monolingual corpus in the domain of interest. We also investigated combinations of the heuristics by using either the union or intersection of merges from two different strategies.

7.3 Compound Merging as Sequence Labeling

Besides extending and combining existing heuristics, we propose a novel formulation of compound merging as a sequence labeling problem. The opposite problem, compound splitting, has successfully been cast as a sequence labeling 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 multi-class 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. If compounds were normalized at splitting time, the compound form has to be restored before concatenating the parts. This can be modeled either as a multi-class classifier that has the possible boundary transformations as its classes or in a two-step process where the form of words are modeled in a separate step after the binary merging decision. In the latter case it is possible to use the reverse normalization process described in Section 7.1. In this work we limited our attention to binary classification.

Consider for instance translating into German the English in Example (7).

(7) 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 Example (8).

(8) 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 example case, the correct labeling for the sentence would be {0,0,1,1,0}, reconstructing the correct German as shown in Example (9).

(9) Europa sollte fremdsprachenkenntnisse fördern

Although 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, especially because compounds can be made up of more than two parts. For this reason, instead of a simple (binary or n-ary) classification problem, we prefer a sequence labeling formulation.

Depending on the choice of the features, this approach has the potential to be truly productive, that is, to form new compounds in an unrestricted way. As discussed, in fact, the list-based heuristics can only form compounds that were observed in the training data or in some suitable monolingual corpus, and are thus not productive. The POS-match heuristic is more flexible, but is still limited in that it can only form

a compound if a modifying element (non-head) has been observed and tagged as such in the training data.

The array of sequence labeling algorithms potentially suitable to our problem is fairly broad, including hidden Markov models (Rabiner 1989), conditional random fields (CRFs) (Lafferty, McCallum, and Pereira 2001), Semi-CRFs (Sarawagi and Cohen 2004), structured perceptrons (Collins 2002), structured support vector machines (Tsochantaridis et al. 2005), Max-Margin Markov networks (Taskar, Guestrin, and Koller 2003), and more. Because 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, capability to handle strongly redundant and overlapping features, and widespread recognition in the NLP community, we chose to use CRFs.

7.3.1 Features. Each sequence item (i.e., each separation point between words) is represented by means of a vector of features. Our aim was to include features representing the knowledge available to the heuristics, such as part-of-speech tags, frequencies for compounds and bigrams, as well as comparisons between them. Features were also inspired by previous work on compound splitting, with the intuition that features that are useful for splitting compounds could also be useful for merging. Character *n*-grams have 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, besides using the original letters. Although 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. We used the following features where -N refers to the nth position before the merge point, and +N to the nth position after the merge point:

- Previous tag
- Surface words: word-2, word-1, word+1, bigram word-1–word+1
- Parts of speech: POS-2, POS-1, POS+1, bigram POS-1–POS+1
- Character *n*-grams around the merge point
  - three-character suffix of word-1
  - three-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 3 (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

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

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

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

Figure 3
Transformations performed for normalizing Swedish characters (Perl notation).

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

7.3.2 Training Data for the Sequence Labeler. Because features are strongly lexicalized, a suitably large training data set is required to prevent overfitting, ruling out the possibility of manual labeling.

We created our training data automatically, using a subset of the compound merging heuristics described in Section 7.2.2, plus an additional heuristic 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 subsequence of a longer word). We compared multiple alternative combinations of heuristics on a validation data set. The validation and test data were created by applying all heuristics, and then manually having the positive instances checked.

A first possibility to automatically generate a training data set 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 to 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 the training of the SMT system.[4] We thus settled for a compromise: We trained the SMT system on the whole training data, translated the whole source side, and 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 real SMT output than the reference translations with automatic splits.

8. Experiments

In this section we report experimental results for the strategies proposed in this article. We first describe, in Section 8.1, the overall experimental set-up. We then report on the experiments, as follows:

- In Section 8.2 we report on general effects of compound processing.
- Section 8.3 reports effects on coalescence of different representation schemes. Here we do not vary merging or splitting.
- Section 8.4 reports on compound splitting and the way different parameter settings affect translation quality. Here we do not vary representation scheme or merging.
- Section 8.5 reports results from varying the merging process. In particular we compare the novel sequence labeling method with the heuristic methods. Here we do not vary representation scheme or splitting.
- In Section 8.6 we apply the overall best strategies to corpora of different sizes and to out-of-domain data.

8.1 Experimental Set-up

We performed experiments on translation from English into German, Swedish, and Danish, all of which have closed compounds. We tested all experimental conditions on Europarl (Koehn 2005) for translation from English to German. We also give contrasting results on English–Swedish Europarl and for an automotive corpus for translation from English to Swedish and Danish. The automotive corpus was gathered from translation memory data. The two corpora are quite different. The automotive corpus is from a limited domain, and of a homogeneous nature, whereas Europarl is more diverse, and tends to have a more complex language than the automotive corpus. Table 4 summarizes the sizes for the corpora that we used in Sections 8.2–8.5. The German test set is the *test2007* set from the WMT 2008 workshop.[5] We have chosen to use a smaller part of Europarl, to reduce runtimes and allow more experiments. In Section 8.6 we report results from scaling experiments and on out-of-domain data.

4 This option can also be extended so that the training data is split into chunks where each chunk is translated by a system that is trained on the remaining chunks, a strategy that has been successfully used for parse reranking (Collins and Koo 2005). Because we found that using fresh data did not give any significant improvements on the merging task except with little training data (see Section 8.5.2, Table 22), we choose not to use this approach since it would be very time consuming and thus impractical.

5 `http://www.statmt.org/wmt08`.

Table 4
Overview of the experimental settings for the experiments in Sections 8.2–8.5.

Name	Europarl German	Europarl Swedish	Auto Swedish	Auto Danish
Corpus	Europarl	Europarl	Automotive	Automotive
Languages	En→De	En→Sv	En→Sv	En→Da
Training sentences	701,157	701,157	329,090	168,047
Avg. target sentence length	20.5	19.4	9.3	9.2
Dev sentences	500	500	2,000	1,000
Test sentences	2,000	2,000	1,000	1,000

We used factored translation (Koehn and Hoang 2007) in our experiments, with both surface words and part-of-speech tags on the target side, with a sequence model on parts-of-speech. For part-of-speech tagging we used TreeTagger (Schmid 1994) for German, an in-house hidden Markov model tagger based on Cutting et al. (1992) for Danish and Swedish, and for Swedish also the Granska tagger (Carlberger and Kann 1999). For the majority of experiments we used the Moses decoder (Koehn et al. 2007), which is a standard phrase-based statistical decoder, which allows factored decoding. For word alignment, Giza++ (Och and Ney 2003) was used and for language modeling we used SRILM (Stolcke 2002). For parameter optimization we used minimum error rate training (Och 2003). For each experiment we ran minimum error rate training three times in order to reduce the effect of optimizer instability, and report the average result and standard deviation. In the merging experiments based on sequence labeling we used the CRF++ toolkit.[6] For the merging experiment with sequence labeling, we used the Matrax decoder (Simard et al. 2005) on the automotive corpus. Matrax is a phrase-based decoder that allows discontiguous phrases, and parameter optimization based on gradient descent for smoothed NIST. We extended the original Matrax decoder with factored decoding on the target side.

Compounds were split before training using the corpus-based method described in Section 5. Except for the experiments comparing different compound merging methods, we used the POS-match merging algorithm developed by us.

We report results on three metrics: Bleu (Papineni et al. 2002), NIST (Doddington 2002), and Meteor. For Meteor we use the version tuned on adequacy and fluency (Lavie and Agarwal 2007) for German, and the original version with default weights (Banerjee and Lavie 2005) for Swedish and Danish, since there is no tuned version for those languages. For Bleu and Meteor we use the %-notation. Significance testing was performed using approximate randomization (Riezler and Maxwell 2005), with 10,000 iterations, on output based on three optimizer runs, as recommended by Clark et al. (2011).

8.2 General Effects of the Compound Processing Strategy

One effect of compound splitting is that the number of word types is reduced. Tables 5 and 6 show the number of types (number of unique words) and tokens (total number of words) and the type/token ratio for Europarl in the different representation schemes. The type count and the type/token ratio in the baseline system are much higher in

6 http://crfpp.sourceforge.net/.

Table 5
Type and token counts and ratio for the German Europarl corpus.

System		Tokens	Types	Ratio
German	baseline	14,356,051	184,215	1.28%
	marked	15,674,728	93,746	0.60%
	unmarked	15,674,728	81,806	0.52%
	sepmarked	17,007,929	81,808	0.48%
English		15,158,429	63,692	0.42%

Table 6
Type and token counts and ratio for the Swedish Europarl corpus.

System		Tokens	Types	Ratio
Swedish	baseline	13,603,062	182,000	1.34%
	marked	14,401,784	107,047	0.74%
	unmarked	14,401,784	100,492	0.70%
English		15,043,321	67,044	0.45%

German and Swedish than in English, and is drastically reduced after compound splitting in all markup schemes, even though it is still higher than in English. One reason for the type count still being higher in German and Swedish than in English is morphological complexity, which is low in English and higher in German and Swedish. The type count is higher in the marked representation scheme than in the other schemes due to the fact that compound modifiers are marked, and do not coincide with other words as they do in the other schemes.

We believe that the fact that compound splitting leads to both type and token counts that are more similar to English is a positive thing, because it likely contributes to making the two languages structurally more similar, which makes the SMT process easier. This is supported by Birch, Osborne, and Koehn (2008), who showed that morphological complexity of a target language correlates negatively with Bleu scores. A lower type/token ratio translates into fewer out of vocabulary (OOV) words, and can give more accurate probability estimates for the SMT translation and language models.

These changes in the number of types and tokens could potentially influence the translation process. In Tables 7 and 8 we show the effect of splitting on average phrase length and average phrase length ratio in the phrase table and during translation in the baseline and in the unmarked system. For the unmarked system we also compensated for the split compounds by counting phrase length as if compounds were merged using the POS-match heuristic. For both languages the lengths and ratios are similar between the baseline and the unmarked system when we compensate for split compounds. This indicates that the type of phrases used are not much affected by the higher number of tokens in the split system. We can also see that the phrases used for translations are on average longer on the source side and shorter on the target side, with a very different ratio than the phrases in the phrase table.

Table 7
Average phrase lengths and English/German length ratios in the phrase table and during translation for German Europarl. German-m is German with merged compounds, where we calculated the phrase length as if compounds were merged using the POS-match method.

		Baseline		Unmarked		
		English	German	English	German	German-m
Phrase table	Phrase length	1.95	2.76	1.91	2.88	2.68
	Phrase ratio		0.86		0.81	0.86
Translation	Phrase length	2.46	2.29	2.48	2.51	2.33
	Phrase ratio		1.19		1.10	1.17

Table 8
Average phrase lengths and English/Swedish length ratios in the phrase table and during translation for Swedish Europarl. Swedish-m is Swedish with merged compounds, where we calculated the phrase length as if compounds were merged using the POS-match method.

		Baseline		Unmarked		
		English	Swedish	English	Swedish	Swedish-m
Phrase table	Phrase length	1.95	2.76	1.96	2.81	2.70
	Phrase ratio		0.86		0.85	0.88
Translation	Phrase length	2.51	2.22	2.61	2.38	2.29
	Phrase ratio		1.21		1.75	1.22

8.3 Promoting Compound Coalescence

In this section we describe three sets of experiments that investigate the use of the *POS-tagsets in sequence models and as count features. In the first experiment we explore the effect of different representation schemes for translation into German, and in the second experiment we do the same, on a smaller scale, for Swedish. In the third experiment we investigate the effect of the RPOS-tagset and the boost and punish models. In all experiments merging was performed using the POS-match heuristic (see Section 7.2.1), except for the sepmarked scheme that used the SPOS-tagset that does not allow this type of POS-match, and for which the symbol merging algorithm (see Section 7.2.2) was used.

8.3.1 Experiment 1. In the first experiment we investigated different compound representation schemes for German Europarl. Table 9 shows the results using different representation schemes, comparing them with two baselines, with and without a POS-sequence model. There is generally a small significant improvement when a POS-based sequence model is used compared with the same model without a POS-based sequence model. Especially for the systems with compound splitting, it was clearly worse not to use *POS-models, and all such systems perform significantly worse than both baselines on most metrics. The representation scheme with sepmarked words and SPOS-tags did not perform well, which can both be due to less power of the mark-up system, and to the fact that it cannot use the POS-match merging heuristic. The two systems with EPOS-models do perform well though, and are on par with the factored baseline, and mostly better than the unfactored baseline. The marked and unmarked systems with EPOS perform similarly, with no significant differences between them. It is thus hard to

Table 9
Translation results for different representation schemes on German Europarl. Significant differences (positive or negative) from the baseline are marked *(5% level) and **(1% level), and differences from baseline+POS are marked similarly with #.

	Bleu	NIST	Meteor
Baseline	20.0 (0.3)	5.94 (0.04)	27.5 (0.2)
Baseline+POS	20.2 (0.1)**	5.93 (0.02)	27.6 (0.2)
Unmarked	19.7 (0.2)**##	5.90 (0.04)**#	27.4 (0.2)##
Marked	19.7 (0.2)**##	5.91 (0.02)**	27.5 (0.1)
Sepmarked	19.0 (0.1)**##	5.81 (0.02)**##	27.0 (0.0)**##
EPOS-unmarked	20.1 (0.0)**	5.97 (0.03)*	27.8 (0.1)**
EPOS-marked	19.9 (0.1)	5.96 (0.02)*	27.7 (0.1)*
SPOS-sepmarked	19.4 (0.3)**##	5.83 (0.01)**##	27.3 (0.2)*##

say which of these classification systems are preferable, but at least it is clear that the use of EPOS-tags are useful.

A more detailed analysis was performed of the compound parts in the output. The outcomes of the merging process were classified into four groups: known compounds; novel compounds; parts of coordinated compounds; and unmerged, single parts. They were further classified into good or bad outcomes. Compounds were judged as bad if they formed non-words or had the wrong form, and compound parts were judged as bad if they should have been merged with the next word, or did not work as a standalone word.

Table 10 shows the results of this analysis. The majority of the merged compounds are known from the training corpus for all representation schemes. There is, again, a marked difference between the systems that use POS-match merging and the sepmarked systems that do not have that information. The sepmarked system found the highest number of novel compounds, but also had the highest error rate, which shows that it is useful to match POS-tags. The EPOS systems have a higher number of novel compounds than the marked and unmarked options without an EPOS model. All these systems had a low error rate of the novel compounds, however. Very few errors were due to reverse normalization; In the EPOS-unmarked system there were only three such errors.

Table 10
Analysis of merged compounds from different representation schemes. The sepmarked systems do not do any matching, and can thus not leave any single parts.

		EPOS-unmarked	EPOS-marked	SPOS-sepmarked	unmarked	marked	sepmarked
Known		3,339	3,375	3,594	3,747	3,587	3,762
Novel	Good	168	105	176	104	93	245
	Bad	20	8	97	10	7	64
Coordinated	Good	43	42	43	42	44	37
	Bad	9	3	9	22	5	7
Single part	Good	6	5	–	136	33	–
	Bad	11	16	–	52	46	–
Total		3,596	3,554	3,919	4,113	3,815	4,115

Generally, the percentage of bad parts or compounds is lower for the systems with a *POS-sequence model, which shows that the sequence model is useful for the ordering of compound parts. The number of single compound parts is also much higher for the systems without a POS sequence model.

The number of compounds found by the splitting algorithm in the German reference text was 4,472. All systems produce fewer compounds than this number. The numbers in Table 10 cannot be directly compared to the baseline system, however, because we do not know which words in its output are compounds.

An indication of how many compounds there are in a text is the number of long words. In the reference text there are 231 word types with at least 20 characters and 1,178 word types with at least 15 characters, which we used as the limits for long words. Other length limits give similar results. Table 11 gives data on absolute numbers of these long word types and recall, precision, and F-score compared with long words found in the reference for the different systems. The distribution of long words in the systems with compound processing generally follows the distribution of compounds from Table 10, which indicates that this is a good indicator of the number of compounds. Both baseline systems have fewer long words than all compound processing systems, indicating that the split-merge strategy does indeed help in increasing the number of compounds in the translation output. The SPOS systems had the highest number of long words; for up to 15 characters there are even more long words than in the reference. The EPOS systems have the same number for up to 15 characters, but a bit lower for up to 20 characters. The EPOS systems also have the highest recall of all systems for both character lengths. The F-scores are similar for the baseline systems and the EPOS systems, with the baseline higher on precision, and the EPOS systems higher on recall. Although the baseline has a higher precision, the absolute number of overlapping words is still higher in many of the systems with compound processing—for instance, for words of up to 20 characters there are 431 in EPOS-unmarked compared with 388 in baseline+POS. It is important to notice, however, that the reference is not a trustworthy gold standard, because there are many possible alternative good translations, and the real quality of long words are likely underestimated. Thus, as can be seen in Table 10, the clear majority of produced compounds in the EPOS systems are judged acceptable, even though the precision of overlap for long words with the reference is at most 47% for the EPOS models.

Table 11
Number of long word types in the translations, and R(ecall), P(recision), and F(-score) compared with long word types in the reference.

	≥20 chars				≥15 chars			
	Number	R	P	F	Number	R	P	F
Reference	231	–	–	–	1,178	–	–	–
Baseline	157	.27	.49	.35	743	.33	.53	.41
Baseline+POS	151	.26	.49	.34	733	.33	.53	.41
Unmarked	192	.27	.40	.32	826	.34	.49	.40
Marked	217	.29	.38	.33	862	.35	.48	.40
Sepmarked	279	.28	.28	.28	1,031	.33	.38	.35
EPOS-unmarked	231	.30	.37	.33	936	.36	.46	.40
EPOS-marked	233	.30	.37	.34	895	.36	.47	.41
SPOS-sepmarked	290	.30	.30	.30	1,043	.34	.39	.36

Table 12
Results for Swedish Europarl. Significant improvements over baseline+POS are marked **(1% level).

System	Bleu	NIST	Meteor
Baseline+POS	21.6 (0.0)	6.11 (0.00)	57.8 (0.1)
EPOS-unmarked	22.0 (0.1)**	6.14 (0.01)	58.3 (0.1)**
EPOS-marked	21.9 (0.0)	6.21 (0.00)**	58.3 (0.0)**

8.3.2 Experiment 2. To further test whether a compound processing strategy using a customized tagset is useful, we performed experiments on an additional language pair, English–Swedish. In this case we always used the successful EPOS-tagset, in combination with either the marked or unmarked representation of compounds.

Table 12 shows the results for translation into Swedish. Both systems with compound processing have higher scores on all metrics. The difference is significant on Meteor, and on either Bleu or NIST for the two systems with compound processing. The only significant differences between using marked and unmarked representations is that the marked system is better on NIST.

To investigate compound translation specifically, we manually classified the system translations corresponding to the first 100 compounds in the reference text that had a clear counterpart in the English source text. As good translations we considered identical translations to the reference, and alternative translations with the same meaning, which we also distinguished between other compounds, single words, or other constructions. We also had a category for word groups that were translated as separate words, but should have been compounded, split compounds.

The result of this evaluation can be seen in Table 13. There are more translations that are identical to the reference in the two systems with splitting, but the total number of identical and alternative translations is approximately the same in the three systems. The number of split compounds is higher in the baseline system. The unmarked system produces more split compounds and partial translations than the marked system. This can be seen as an indication of marking having an effect, which, however, is not clear in the automatic evaluation.

We also investigated the quality of the merged compounds, especially with regard to the reverse normalization that is needed in the unmarked systems, where compounding forms were normalized. There were no merging errors in the marked system. In the

Table 13
Analysis of the translations of 100 source items yielding compounds in a Swedish reference text.

	Baseline+POS	EPOS-Unmarked	EPOS-Marked
Identical comp.	48	53	57
Alt. compound	14	9	10
Alt. word	16	16	12
Alt. other	9	8	9
Split compound	7	5	3
Partly transl.	4	7	4
No equivalent	0	0	2
OOV	2	2	3

Table 14
Results of coalescence experiment on the automotive corpus. Scores that are significantly different from the baseline are marked *(5% level), and differences from Baseline+POS are marked with #.

	Auto Danish			Auto Swedish		
	Bleu	NIST	Meteor	Bleu	NIST	Meteor
Baseline	81.3 (0.0)	9.74 (0.01)	89.5 (0.1)	67.7 (0.3)	9.96 (0.02)	85.4 (0.1)
Baseline+POS	81.4 (0.1)	9.73 (0.02)	89.5 (0.1)	67.8 (0.1)	9.94 (0.03)	85.1 (0.1)
EPOS	80.6 (0.2)*	9.67 (0.02)	89.1 (0.2)	68.5 (0.2)	10.06 (0.01)*	85.5 (0.2)
RPOS	80.9 (0.2)	9.69 (0.03)	89.4 (0.2)	68.5 (0.2)*	10.07 (0.02)*#	85.8 (0.1)#
boost	81.0 (0.3)	9.72 (0.03)	89.7 (0.2)	68.3 (0.1)	10.05 (0.01)*	85.7 (0.0)#
punish	80.7 (0.1)	9.70 (0.01)	89.4 (0.1)	68.3 (0.2)	10.04 (0.01)*	85.7 (0.1)#
RPOS+boost	81.0 (0.1)	9.73 (0.01)	89.7 (0.2)	68.2 (0.2)	10.04 (0.02)	85.7 (0.2)
RPOS+punish	81.0 (0.1)	9.71 (0.03)	89.6 (0.2)	68.2 (0.2)	10.04 (0.03)*	85.6 (0.2)#

unmarked system there were only two minor errors due to failed reverse normalization. The first error is a missing insertion of an *+s* for *medlem/+s/länder* (*member countries*) and in the second case, **samhäll/-e+s/politiska* (*socio-political*), a combination change *-e/+s* is wrong.

8.3.3 Experiment 3. In the third set of experiments with factored translation models we investigated the RPOS tagsets and the use of count features for the Danish and Swedish automotive corpus and German Europarl. Compound parts were merged using the POS-match heuristic.

Results on the two automotive corpora are shown in Table 14. The scores are very high, which is due to the fact that it is an easy domain with many repetitive sentence types. In this case there are no significant differences between the two baselines when using a POS-model. For Swedish, the results are overall higher for the systems with compound splitting, a difference that is significant for some systems and metrics. Overall, the RPOS system performs best for Swedish, with results significantly better than at least one baseline on all metrics. For Danish, on the other hand, the only significant difference between any baseline and system with splitting is for the EPOS system, which is slightly worse than the baseline with POS. For the other systems there are no significant differences to the baseline.

Table 15 shows results using RPOS for German Europarl. For this corpus neither the RPOS model nor the boost and punish models works well. They all give significantly worse results than the baseline. As we have seen before, however, the EPOS model gives competitive results to the baseline for this corpus.

Table 15
Results of coalescence experiment on German Europarl. Scores that are significantly different from the Baseline+POS are marked with **(1% level).

System	**Bleu**	**NIST**	**Meteor**
Baseline+POS	20.2 (0.1)	5.93 (0.02)	27.6 (0.2)
EPOS	20.1 (0.1)	5.97 (0.03)	27.7 (0.1)
RPOS	16.5 (0.1)**	5.32 (0.03)**	25.0 (0.2)**
boost	16.3 (0.2)**	5.29 (0.03)**	24.8 (0.3)**
punish	16.6 (0.1)**	5.33 (0.03)**	25.1 (0.2)**

Overall, we found that it was successful to use *POS-sequence models in a factored translation model in order to handle compound coalescence. Our best models with compound splitting perform at least on par with the baseline, and sometimes better. We also showed that there are other advantages to using compound processing, especially that the number of long words are similar to the baseline, which is not the case for our baseline systems, which have too few long words. We also showed that *POS-sequence models are essential for the compound processing approach to be successful. On Europarl the EPOS-tagset was the most successful, whereas the RPOS-tagset and count features could help for the automotive domain.

8.4 Influence of Splitting Strategies

In the following experiments we investigated how compound splitting strategies of different quality influenced the suggested compound processing method. In order to do this we performed an intrinsic evaluation of several splitting methods, and compared it to translation results using the same splitting strategies. In this experiment we used German Europarl.

We used a modified version of the splitting strategy of Koehn and Knight (2003) as a basis for the comparison, and applied it to all nouns, adjectives, adverbs, and verbs of minimum length six characters into parts of minimum three characters, by allowing all splits where the parts were found in a corpus, and were tagged as content words. Parts were allowed to be modified by all compound suffixes from Langer (1998). The best splitting option, which can be no split, was chosen based on the arithmetic mean of the corpus frequencies of the parts. We also imposed the restriction on the compound head that its part-of-speech tag needs to be the same as for the full word. We call this system **arith**. In other variants of splitting strategies, one feature at the time was changed, based on the arith system:

geom – using the geometric mean instead of the arithmetic mean
eager – choosing the split with the highest number of parts, instead of using the arithmetic mean
part4 – limiting the length of compound parts to four characters
max2 – limiting the maximum number of parts per compound to two
common – only allowing the common compound suffixes *-s*, *-es*, *-n*, *-nen*
anypos – not using part-of-speech tags

The corpus frequencies were gathered from the target side of the training corpus. We compared the systems with compound splitting with a factored baseline system without any compound processing.

To measure the success of the different compound splitting algorithms we performed an intrinsic evaluation. We used the gold standard from Stymne (2008), created by manually annotating the first 5,000 words of the test text for one-to-one correspondence with the English reference text, similar to Koehn and Knight (2003). A one-to-one correspondence occurs when the words in a German compound are translated as separate words in English. In addition there can be inserted function words. As an example, *Medienfreiheit* is in one-to-one correspondence with *freedom of the media*, since the two German parts *Medien* and *Freiheit* corresponds to two separate words, *media* and *freedom*. The two function words *of, the* are ignored. Out of the 5,000 words 174 were compounds in one-to-one correspondence with English. The result of the

one-to-one evaluation is shown in Table 16. The same metrics and categories as in Koehn and Knight (2003) were used:

correct split: words that were correctly split
correct not: words that should not be split and were not
wrong not: words that should be split but were not
wrong faulty: words that were split but in an incorrect way
wrong split: words that should not be split but were
precision: (correct split) / (correct split + wrong faulty + wrong split)
recall: (correct split) / (correct split + wrong faulty + wrong not)
accuracy: (correct) / (correct + wrong)

The splitting options have their strengths on different metrics, with three different methods having the best results for the three metrics used. Compared to the arith method it can be seen that both imposing length restrictions and using the geometric mean increases the results on all three metrics. Limiting the number of parts gives the highest recall. Using only common compound suffixes gives higher precision, and not using part-of-speech gives lower precision. The baseline system without splitting actually has the highest accuracy. To a large extent this is due to the fact that the test set is taken from running text, with a high number of non-compounds.

Overall, the results for all splitting variations are quite low. However, most of the strategies tend to split too much rather than too little, which is preferable, because the phrase-based translation system has the possibility of recovering from over-splitting by handling compounds that are split into too many parts in consistent phrase pairs. Evaluation towards a one-to-one gold standard is rather harsh because the splitting algorithm has no knowledge of the corresponding English structure. If we instead use a gold standard of linguistically motivated compounds, which are many more (545 for the 5,000 word set), precision is much higher for all systems, with mostly a lower recall. For the arith system, for instance, the precision is more than doubled at .597 with a slightly lower recall of .522.

The results for the translation task from English into German are shown in Table 17. In this experiment we used a different selection of 701K training sentences from Europarl than in Section 8.3, which resulted in overall lower scores both for the baseline and for the systems with compound processing. Overall, there are very small differences between both the baseline system and the systems with different types of splitting, and most of the small differences between the systems are not significant. The

Table 16
One-to-one correspondence of split compounds compared with a manually annotated gold standard for the different splitting methods.

	Correct		Wrong			Metrics		
	split	not	not	faulty	split	Precision	Recall	Accuracy
baseline	0	4,826	174	0	0	–	0	.966
arith	99	4,504	22	52	323	.209	.572	.921
geom	109	4,614	33	31	213	.309	.630	.945
eager	43	4,243	18	112	584	.058	.249	.857
part4	120	4,692	36	17	135	.441	.693	.962
max2	133	4,546	29	11	281	.313	.769	.936
common	99	4,714	58	16	113	.434	.572	.963
anypos	100	4,216	8	65	611	.128	.578	.863

Table 17
Translation results using different splitting methods for German Europarl. Significant changes from the baseline are marked *(5% level) and **(1% level).

	Bleu	NIST	Meteor
baseline+POS	19.1 (0.1)	5.79 (0.02)	26.8 (0.0)
arith	19.0 (0.0)	5.79 (0.01)	26.6 (0.1)
geom	18.9 (0.1)*	5.78 (0.02)	26.5 (0.2)**
eager	18.8 (0.1)**	5.80 (0.01)	26.6 (0.1)
part4	18.8 (0.1)**	5.79 (0.01)	26.5 (0.1)*
max2	18.8 (0.1)	5.80 (0.01)	26.6 (0.1)
common	18.9 (0.1)	5.78 (0.01)	26.6 (0.1)
anypos	18.9 (0.1)	5.75 (0.01)**	26.6 (0.1)

arith, max2, and common systems are on par with the baseline on all metrics, with no significant differences, whereas the other systems are worse than the baseline on at least one metric.

On the intrinsic evaluation there were quite large differences between the systems, which are not found on the translation task. This is similar to previous work by Koehn and Knight (2003) for translation in the other direction, where systems similar to the eager and geom system performed similarly on a translation task, while the eager system was much worse on their intrinsic evaluation. In our evaluation only the eager system is significantly worse than any other system on Bleu, where it is worse than the baseline and the arith system. The arith system overall is competitive with both the baseline and the other split systems, despite the relatively low results on the intrinsic evaluation.

Overall, it seems that the translation task is not very sensitive to the quality of the splitting strategy. As in previous research (Koehn and Knight 2003; Stymne 2008) there are no clear relations between the intrinsic evaluation and the MT evaluation. Both systems with low intrinsic accuracy (such as anypos) and with high accuracy (such as geom) tend to be worse than other systems on at least some MT metrics, even though the differences are small. We thus think that for the task of translation into compounding languages, the MT performance cannot be predicted based on intrinsic evaluations.

In a previous similar study using a smaller corpus (Stymne 2008), we found somewhat bigger differences between the systems, and in that study most of the systems with splitting were better than the baseline. In that study, too, the arith system was among the best performing on the MT task, even though many of the differences to the other systems with splitting were not significant. Because the arith strategy consistently has given good results for the MT task, we chose to use it in our other experiment. There would, however, have been other reasonable choices of splitting, such as using the max2 system.

8.5 Compound Merging

In the following studies we investigated the impact of the compound merging strategy used. We compared previously suggested strategies to the new strategies we have developed. We first present results for heuristic merging strategies, and then go on to compare the best heuristic methods to a sequence labeling merging strategy. For the heuristic merging we used German Europarl, and for the comparison with sequence labeling we used German Europarl and the automotive corpus for Swedish and Danish. In these experiments we used the arith compound splitting method from Section 8.4.

8.5.1 Heuristic Merging. Three main types of merging algorithms were investigated in this study. The first group, inspired by Popović, Stein, and Ney (2006), is based on frequency lists of words or compounds, and of parts, compiled at split-time. The second group uses symbols to guide merging, inspired by work on morphology merging (Virpioja et al. 2007). The third group takes *POS-tags for compounds into account, so that merging takes place if and only if the part-of-speech tags match. In addition, we examined combined merging methods, using either the union or intersection of merges proposed by different methods. We also extended the list- and symbol-based methods by a restriction that the head of the compound should have a compounding part of speech, that is, a noun, adjective, or verb. By using these additions and also combinations of the main algorithms, a total of eleven algorithms were explored, as summarized in Table 18. For all algorithms, compounds can have an arbitrary number of parts.

In these experiments we used the unmarked representation scheme, which means that compound modifiers were normalized, but not marked with any symbol, and we used the EPOS-tagset. To handle normalization, we used the reverse normalization process in combination with all merging strategies. If there were any compound modifiers that could not be combined with the next word, in any of the algorithms, that part was left as a single word. For frequency calculations for the list-based strategies we used the German part of the SMT training corpus.

Table 19 shows the translation results using the different merging algorithms. The merging methods based on lists all give significantly worse results than the baseline on all metrics, except when combined with the symbol method. The compound-list+ method is the best method of the systems that only uses list information; it is significantly better than word-list and compound-list on all metrics, but still significantly worse than the baseline, symbol, and POS-match systems. When using the head-pos restriction, the results get even better, but here POS-tag information is used in addition

Table 18
Merging algorithms.

Name	Description
word-list	Merges each token that has been seen as a compound part with the next part if it results in a known word
word-list + head-pos	As word-list, but only words where the last part is a noun, adjective, or verb are merged
compound-list	As word-list, but for known compounds from split-time, not for all known words
compound-list+	As compound-list, but also includes a check that the frequency of the resulting compound is higher than the frequency of the unmerged bigram
symbol	Merges all tokens that are marked as compound modifiers with the next token
symbol + head-pos	As symbol, but only merges words where the head is a noun, adjective, or verb
symbol ∧ word-list	Merges marked modifier parts if the result is a known word
POS-match	Merges tokens with a compound part-of-speech tag with the next token, if their tags match
POS-match + coord	As POS-match, but also adds a hyphen if a compound modifier is followed by the conjunction *und* [*and*]
POS-match ∧ compound-list+	Merges tokens if they are selected by both of these methods
POS-match ∨ compound-list+	Merges tokens if they are selected by either of these methods

Table 19
Translation results for German Europarl using an EPOS-model and the unmarked representation scheme. Significant differences (positive or negative) from the baseline are marked **(1% level).

	Bleu	NIST	Meteor
baseline+POS	20.2 (0.1)	5.93 (0.02)	27.6 (0.2)
word-list	17.9 (0.0)**	5.70 (0.02)**	25.8 (0.1)**
word-list + head-pos	19.3 (0.0)**	5.83 (0.03)**	27.1 (0.1)**
compound-list	18.1 (0.0)**	5.61 (0.02)**	26.0 (0.1)**
compound-list+	18.7 (0.0)**	5.66 (0.03)**	26.6 (0.1)**
symbol	20.0 (0.0)	5.96 (0.03)	27.7 (0.1)
symbol + head-pos	20.0 (0.0)	5.95 (0.03)	27.8 (0.1)
symbol ∧ word-list	20.0 (0.0)**	5.95 (0.03)	27.8 (0.1)
POS-match	20.1 (0.1)	5.97 (0.03)	27.7 (0.1)
POS-match + coord	20.1 (0.0)	5.97 (0.03)	27.8 (0.1)
POS-match ∧ compound-list+	19.0 (0.1)**	5.70 (0.03)**	26.7 (0.1)**
POS-match ∨ compound-list+	18.8 (0.0)**	5.74 (0.03)**	26.3 (0.1)**

to only frequency list information. There are no significant differences between the baseline and the systems with symbol or POS-match merging, but the trend on most metrics is that the POS-match systems have a slightly higher score than the other methods. However, as shown in Section 8.3 (Table 10), POS-match does block some erroneous merging. Especially with EPOS the order is so good that the matching constraint is rarely needed. Adding treatment of coordinated compounds to the POS-match algorithm changes scores marginally, but again, as shown in Table 10, it does improve merging for coordinated compounds. The combination of POS-match and compound-list+ was not successful, neither as a union nor as an intersection of the strategies. Table 19 only shows results with an EPOS-model. The result pattern for systems without an EPOS-model are similar to using an EPOS-model, but lower.

Table 20 shows the number of merges performed by applying the different algorithms. The word-list–based method produces the highest number of merges, including many cases that should not be merged. The number of merges is greatly reduced by the head-pos restriction, or by combining the word-list method with some other merging

Table 20
Number of merges for the different merging algorithms. The numbers can be compared with the number of compounds found by the splitting algorithm in the reference text, which is 4,472.

	with EPOS	without EPOS
word-list	5,275	5,897
word-list + head-pos	4,161	4,752
compound-list	4,460	5,116
compound-list+	3,843	4,427
symbol	4,431	5,144
symbol + head-pos	4,323	4,832
symbol ∧ word-list	4,178	4,753
POS-match	4,363	4,867
POS-match + coord	4,361	4,865
POS-match ∧ compound-list+	3,502	4,018
POS-match ∨ compound-list+	4,706	5,281

strategy. An investigation of the output of the word-list–based method shows that it often merges common words that incidentally form a new word, such as *bei* [*at*] and *der* [*the*] to *beider* [*both*]. Another type of error is due to errors in the corpus, such as merging *umwelt* [*environment*] and *und* [*and*], which exists as a mistyped word in the corpus, but is not a correct German word. These two error types are often prohibited by the head-pos restrictions and by the symbol and POS-match algorithms. The extensions of the simple word-list method avoid these types of errors, resulting in a lower number of merges. The compound-list+ method has the overall lowest number of splits, but it still is the best of the methods based only on frequency lists, without the use of POS or symbols. With the EPOS-model there are very small differences in number of merges between the methods based on symbol and POS-match, although this difference is much larger without the EPOS-model. For instance, the difference is 68 merges between symbol and POS-match with the EPOS-model, whereas without the EPOS-model the difference is 277. This difference shows that the EPOS-model clearly helps in improving coalescence.

The choice of merging method thus has a large impact on the final translation results. For merging to be successful we found that some knowledge source that is passed through the translation process, such as EPOS-tags, is needed. The pure word-list–based method performed the worst of all systems in most cases. On the automatic metrics, the POS-match and symbol methods gave about the same results as the baseline; we believe there are other strengths to this method as compared to the baseline system, however, such as the production of a higher number of compounds. A further advantage of the POS-match strategy is that it can form novel compounds, which is not the case for either the baseline or the word-list–based methods.

8.5.2 Merging as Sequence Labeling. Finally, we performed experiments where we compared our suggestion of a sequence labeling merging method to heuristic merging. We chose two heuristic strategies, the POS-match strategy, which gave the overall best results, and compound-list+, which gave the best results of the word-list based merging strategies without any POS or symbol information. We also found that the intersection of these two strategies worked well on some of our corpora, and decided to include that as well. In addition, we used a reference-based heuristic for creating the CRF training data, which identified merges resulting in compounds that are present in the reference data. Such a heuristic cannot be used at translation-time, however, because there is no reference data then. In this section we will refer to these heuristics as **list** for the compound-list+ heuristic, **POS** for the POS-match strategy, and **ref** for the reference-based heuristic.

For these experiments we used three data sets, as summarized in Table 21. For the training data for the sequence labeler we translated all of the training data, except

Table 21
Overview of the experimental settings for comparing heuristic and sequence labeling merging. Training sentences are taken from translated SMT training data, except extra data, which was not used for SMT training.

Corpus	Europarl German	Auto Swedish	Auto Danish
Compounds split	N, V, Adj	N, V, Adj	N
POS tagsets	POS	POS	RPOS
Training sentences CRF	249,388	317,398	164,702
Extra training sentences CRF	–	–	163,201

one-word sentences, which were filtered out, for the smaller automotive corpora. For the Europarl experiment, we limited the training data to the 250,000 first sentences of the training data, and again filtered out one-word sentences. For the Danish corpora we also had access to additional data, which we used to control for the effect of reusing SMT training data. For the machine learning we wanted to use separate development and test sets. We chose to use the test sets from Table 4 for development testing, where we used the first 1,000 sentences for German Europarl, and picked new unseen 1,000 sentence test sets for the final evaluation. For frequency calculations of compounds and compound parts that were needed for compound splitting and some of the compound merging strategies, we used the target side of the SMT training data.

In these experiments we used the unmarked representation scheme. For German Europarl we used the unmarked scheme as it is with a binary sequence labeler. Reverse normalization can be performed afterwards as for the other strategies. For the automotive corpus we used the unmarked scheme without normalization.

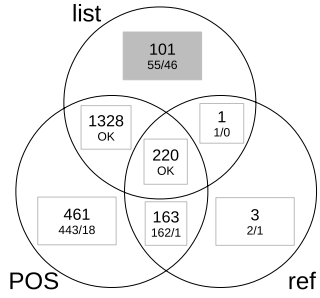
We compared alternative combinations of the heuristics on our three validation data sets (see Figure 4). In order to estimate the amount of false negatives for all three heuristics, we inspected the first 100 sentences of each validation set, looking for words that should be merged, but were not marked by any of the heuristics. In no case could we find any such words, so we thus assumed that between them, the heuristics can find the overwhelming majority of all compounds to be merged. The differences across domains, Europarl and automotive, is quite striking, with a much higher number of compounds found by using the reference-based heuristic for the automotive corpus.

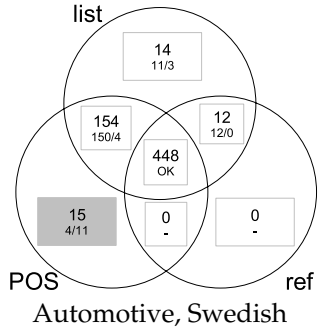
We conducted a round of preliminary experiments to identify the best combination of the heuristics available at training time (compound-list+, POS-match, and reference-based) to use to create automatically the training data for the CRF. The best results on the validation data are obtained by different combinations of heuristics for the three data sets, as could be expected by the different distribution of errors in Figure 4. In the following experiments we trained the CRF using for each data set the combination of heuristics corresponding to leaving out the gray portions of the Venn diagrams. This sort of preliminary optimization requires hand-labeling a certain amount of data. Based on our experiments, skipping this optimization and just using ref∨POS (the optimal configuration for the German–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. Results are largely insensitive to variations in these hyper-parameters, especially to the CRF regularization parameter.

For the Danish automotive corpus we had access to training data that had not been used to train the SMT system, so we could test the performance of the CRF when trained on that data as compared with training on the possibly biased data that was used to train the SMT system. Table 22 shows the results with the two types of training data. For the smallest training size, 20,000 sentences, the unseen training data is significantly better on recall and F-score. For the other sizes, there are no significant differences when the same amount of training data is used. When we added the unseen data to the SMT data, we only saw small non-significant improvements. This indicates that it is feasible to use translated SMT training data for the sequence labeler. We still decided to use all available training data for our subsequent experiments on Danish Europarl.

The overall merging results of the heuristics and the best sequence labeler are shown in Table 23. Notice how the list and POS heuristics have complementary sets of false negatives: when merging on the union of the two heuristics, the number of false





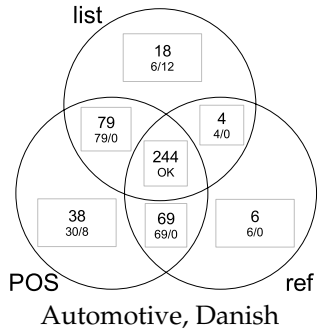


Figure 4
Evaluation of the different heuristics on validation files from the three corpora. The number in each region of the Venn diagrams indicates the number of times a certain combination of heuristics fired (i.e., the number of positives for that combination). The two smaller numbers below indicate the number of true and false positive, respectively. Venn diagram regions corresponding to unreliable combinations of heuristics have corresponding figures on a gray background. OK means that a large fraction of the Venn cell was inspected, and no false positive was found.

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 either heuristic alone in all cases on test data, and in several cases on validation data. The list heuristic alone performs reasonably well on the Swedish data set, but has a very low recall on the German and Danish data sets. In all four cases the SMT training data has been used for the list used by the heuristic, so this is unexpected, especially considering the fact that

Table 22
Experimental results for CRF on Danish Automotive, with different training data and sizes. SMT is the same data that was used to train the SMT system, and new is additional data.

Data	Size	Precision	Recall	F-score
SMT	20K	.977	.970	.974
SMT	50K	.979	.977	.978
SMT	100K	.977	.980	.978
SMT	165K	.977	.986	.982
new	20K	.977	.984	.981
new	50K	.975	.986	.981
new	100K	.978	.991	.984
new	163K	.978	.991	.984
all SMT + new	165K + 20K	.978	.991	.984
all SMT + new	165K + 50K	.982	.993	.988
all SMT + new	165K + 100K	.980	.991	.985
all SMT + new	165K + 163K	.978	.993	.985

Table 23
Precision, Recall, and F-score for compound merging methods based on heuristics or sequence labeling on validation data and on held-out test data. The superscripts mark the systems that are significantly worse than the system in question (l-list, p-POS, lp-list∨POS, c-best CRF configuration).

	Validation data			Test data		
	Precision	Recall	F-score	Precision	Recall	F-score
			German Europarl			
list	.967	.724	.828	.952	.717	.818
POS	.991^{l,lp}	.978^{l}	.984^{l}	.997^{l,lp}	.980^{l}	.989^{l}
list∨POS	.967	.999^{l,p,c}	.983^{l}	.963^{l}	1^{l,p,c}	.981^{l,p}
CRF (ref∨POS)	.982^{l,lp}	.990^{l,p}	.986^{l,p}	.983^{l,lp}	.996^{l,p}	.989^{l,lp}
			Swedish auto			
list	.989^{p,lp}	.994^{p}	.991^{p}	.990	.977	.984
POS	.976	.963	.969	.992^{lp}	.974	.983
list∨POS	.972	1^{p}	.986^{p}	.982	.998^{l,p,c}	.990^{l,p}
CRF (ref∨list)	.987^{p,lp}	.998^{p}	.993^{p,lp}	.987	.987	.987
			Danish auto			
list	.925	.760	.835	.991^{lp}	.764	.863
POS	.981^{l,lp}	.964^{l}	.972^{l,lp}	.978	.929^{l}	.954^{l}
list∨POS	.925	.986^{l,p}	.955^{l}	.976	.988^{l,p,c}	.982^{l,p,c}
CRF (ref∨list∨POS)	.978^{l,lp}	.993^{l,p}	.985^{l,p,lp}	.978	.966^{l,p}	.972^{l,p}

the Danish data set is in the same domain as one of the Swedish data sets. 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 on the other data sets 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 four cases, than the best heuristic, the combination heuristic, which is positive (because erroneously merged compounds are usually more disturbing for a reader or post-editor than non-merged compounds).

The differences between the systems on MT metrics are generally small and mostly not significant, as could be expected, since the differences were small on the intrinsic evaluation. For German Europarl only 10–739 out of 29,786 words are actually different between any pair of systems. The only noticeable differences between the systems are that the list system is worse than the other systems for Danish and German. The list strategy is competitive for Swedish automotive, which is much bigger than Danish, and in one of our previous experiments, with a large Swedish Europarl corpus (Stymne and Cancedda 2011), confirming our intuition that the list strategy tends to work better with large corpora. Even though we do not see much effect on MT metrics, we still think the small differences on the intrinsic evaluation are important. Compound merging is a postprocessing process, which operates directly on the actual translation results. This is quite different from compound splitting, as described in Section 8.4, which is performed as part of the preprocessing and where we cannot be sure on the impact of the preprocessing on the translation process.

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 some 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*) for Danish, and *kvarts sekel* (*quarter century*) and *bostad(s)+ersättning* (*housing grant*) for Swedish. This confirms that the sequence labeler, trained from automatically labeled data based on heuristics, can learn to merge new compounds that the heuristics themselves cannot find.

8.6 Effects of Corpus Size and Domain

Most of the previous experiments were performed on relatively small data sets, and it has been shown that effects of preprocessing strategies sometimes are larger on small corpora (e.g., Nießen and Ney 2004; Popović and Ney 2006). To investigate this issue, we first present a scaling experiment for small data sets from German Europarl, and then report results on two large data sets for translation from English to German on Europarl and news data.

Table 24 shows the result for baseline and EPOS systems trained on 100K and 440K sentences. In the smaller data condition the results are similar for the baseline and the EPOS system. With 440K data the EPOS system is significantly better than the baseline

Table 24
Results for translation on German Europarl with small training data sets. Significant improvements over the respective baseline+POS are marked **(1% level).

System	Bleu	NIST	Meteor
100K baseline+POS	16.8 (0.0)	5.37 (0.02)	24.6 (0.1)
100K EPOS	16.9 (0.1)	5.39 (0.02)	24.8 (0.1)
440K baseline+POS	19.3 (0.1)	5.76 (0.05)	26.7 (0.2)
440K EPOS	19.7 (0.2)**	5.83 (0.03)**	27.0 (0.2)**

Table 25
Overview of the corpus for large evaluation sets, and the corpus sizes in sentences. The names of the corpora refer to those used for the WMT evaluations. For the tuning data only a subset of the full data was used.

	WMT08		WMT10	
	Name/type	Size	Name/type	Size
Translation models	Europarl	1,054,688	Europarl+NewsComm	1,462,401
Language model 1	Europarl	1,412,546	Europarl+NewsComm	1,885,872
Language model 2	–		News	17,449,589
Tuning data	dev2006	600	news-test2008	1,025
Test Set Europarl	test2007	2,000	–	
Test Set News	nc-test2007	2,007	news-test2009	2,525

Table 26
Results on WMT08 data on Europarl and News test sets. Significantly different results from the baseline are marked with *(5% level) and **(1% level), significantly different results from baseline+POS are marked similarly with #.

	Europarl			News		
System	**Bleu**	**NIST**	**Meteor**	**Bleu**	**NIST**	**Meteor**
Baseline	19.9 (0.1)	5.89 (0.01)	27.6 (0.0)	14.3 (0.3)	5.61 (0.09)	24.0 (0.5)
Baseline+POS	20.1 (0.1)*	5.90 (0.01)	27.8 (0.2)	14.4 (0.3)*	5.60 (0.09)	23.9 (0.4)
EPOS-unmarked	20.1 (0.0)	5.95 (0.01)*#	27.6 (0.2)	14.7 (0.4)**#	5.60 (0.07)	24.1 (0.6)*

on all three metrics. We thus see that our compound processing method works well on medium-sized data sets as well, and is competitive to the baseline on very small data sets.

In our final experiments we used larger data sets from the workshop of statistical machine transition, year 2008 and 2010, which we will refer to as WMT08 and WMT10.[7] The corpora used are presented in Table 25. In these experiments we also investigated the effects of in- and out-of-domain data. For WMT08 we trained and tuned on only Europarl data, and tested both in-domain, on Europarl and out-of-domain, on News. For WMT10 we trained on a mix of data, and tested on News. For these experiments we used morphologically enriched POS-tagsets. For WMT08 we used a commercial dependency parser for the morphology (Tapanainen and Järvinen 1997), and for WMT10 we used RFTagger (Schmid and Laws 2008). For the systems with split compounds we used the EPOS-tagset based on morphological tags. We used the POS-match merging algorithm.

Table 26 shows the results on the WMT08 evaluation. On Europarl the system with compound processing and an EPOS-model is significantly better than both baselines on NIST, and on the News domain it is significantly better than both baselines on Bleu. On the other metrics the EPOS system is on par with the baseline for both domains. The differences between the two baselines are mostly small. Overall, the scores are much lower on the out-of-domain data. Table 27 shows the results for the WMT10 evaluation, where more data was used to train the systems. The EPOS system is significantly better

7 http://www.statmt.org/wmt08/, http://www.statmt.org/wmt10/.

Table 27
Results on WMT10 data on News test set. Significance shown as in Table 26.

System	Bleu	NIST	Meteor
Baseline	13.5 (0.1)	5.29 (0.04)	21.0 (0.2)
Baseline+POS	13.8 (0.1)*	5.35 (0.01)	21.3 (0.2)
EPOS-unmarked	14.1 (0.3)**##	5.35 (0.07)**	21.4 (0.1)*

than the baseline without POS on all metrics, and also significantly better than the baseline with POS on Bleu. This shows that the suggested compound processing is effective also on larger data sets and out-of-domain data.

9. Conclusion

In this article, we have investigated several methods for splitting and merging compounds when translating into languages with closed compounds, with the goal of generating novel compounds. We have described methods for promoting coalescence and for deciding if and how to merge words that are either competitive with, or superior to, any previously published method. A common trait of all these methods is that they promote coalescence and allow formation of novel compounds by using customized *POS-tagsets. This shows that the integration of light-weight linguistic information through POS-tags is useful for compound processing into compounding languages. We believe that some of the techniques, such as customized POS-sequence models, could be used to target morphological phenomena other than compounding as well. We also believe that the suggested methods are likely to be useful for other compounding languages such as Finnish or Japanese.

For promoting compound coalescence we introduced additional LMs based on customized POS-tagsets, and with dedicated SMT model features counting the number of sequences known a priori to be desirable and undesirable. Experiments showed consistently good results when using the extended tagset EPOS in sequence models. The other tagsets were less successful, but on an automotive corpora a restricted tagset, RPOS worked well both in sequence models and using count features.

For compound splitting, previously proposed strategies for the opposite translation direction could be modified to improve performance when translating into German. We also confirmed previous results that showed no correlations between intrinsic evaluations of compound splitting and translation results in the other translation direction.

For merging we designed a heuristic algorithm based on part-of-speech matching, which gave consistently good results. We also improved existing heuristics based on frequency lists, which worked well on some data sets. We furthermore cast the compound merging problem as a sequence labeling 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, even trained on automatically labeled data.

Our goal was to identify methods that can generate novel compounds. Depending on the choice of the features, the sequence labeling approach has the potential to be truly productive, that is, to form new compounds in an unrestricted way. The list-based heuristic is not productive: It can only form a compound if this was already observed as such. The POS-match heuristic yields some productivity. Because it uses special

POS-tags for compound modifiers, it can form a compound provided its head has been seen either alone or as a head, and its modifier(s) have been seen elsewhere, possibly separately, as modifier(s) of compounds. The sequence labeling approach can decide to merge two consecutive words even if neither was ever seen before in a compound.

In the merging step we only considered the order of words as they appear in the 1-best output. Other options would include also attempting reordering of words in the output in addition to merging, or to use lattice or *n*-best output and perform compound merging and reranking in combination. However, we believe that the use of sequence models for coalescence gives sufficiently good results and minimizes the potential impact of more costly options.

In this work we mainly viewed the translation system as a black box. Our modifications to the PBSMT pipeline were based on pre- and postprocessing and on the use of *POS-sequence models. It is possible that further improvements can be had by integrating compound processing tighter into the decoder.

The rate of novel compounds is relatively low in the data sets used in our experiments, mainly due to the fact that most experiments are carried out with in-domain test data. However, the best methods tend to vary between the domain-restricted automotive corpora and Europarl. The importance of effective compound splitting and merging can be expected to grow in domain adaptation settings, outside the scope of this work.

Overall we have shown that systems with compound processing give consistently good results when they use POS-based language models and use either the POS-match heuristic or sequence labeling for compound merging. The difference between a baseline using a POS-model but no compound splitting and the split-merge EPOS-models is small in all experiments and does not seem to increase with the size of the training corpus. However, only the latter can produce novel compounds.

Acknowledgments

We would like to thank Maria Holmqvist and Tamás Gaál for valuable discussions about different aspects of the work in this article. We also thank the anonymous reviewers for their valuable comments on an earlier draft. The work presented in the article was mostly conducted while S. Stymne was at Linköping University and Xerox Research Centre Europe.

References

Badr, Ibrahim, Rabih Zbib, and James Glass. 2008. Segmentation for English-to-Arabic statistical machine translation. In *Proceedings of the 46th Annual Meeting of the ACL: Human Language Technologies, Short papers*, pages 153–156, Columbus, OH.

Banerjee, Satanjeev and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at ACL'05*, pages 65–72, Ann Arbor, MI.

Baroni, Marco, Johannes Matiasek, and Harald Trost. 2002. Predicting the components of German nominal compounds. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI)*, pages 470–474, Amsterdam.

Berton, André, Pablo Fetter, and Peter Regel-Brietzmann. 1996. Compound words in large-vocabulary German speech recognition systems. In *Proceedings of the Fourth International Conference on Spoken Language Processing*, pages 1,165–1,168, Philadelphia, PA.

Birch, Alexandra, Miles Osborne, and Philipp Koehn. 2008. Predicting success in machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 745–754, Honolulu, HI.

Botha, Jan A., Chris Dyer, and Phil Blunsom. 2012. Bayesian language modelling of German compounds. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 341–356, Mumbai.

Brodda, Benny. 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.
Brown, Ralf D. 2002. Corpus-driven splitting of compound words. In *Proceedings of the 9th International Conference of Theoretical and Methodological Issues in Machine Translation*, pages 12–21, Keihanna.
Carlberger, Johan, Rickard Domeij, Viggo Kann, and Ola Knutsson. 2005. The development and performance of a grammar checker for Swedish: A language engineering perspective. In Ola Knutsson, editor, *Developing and Evaluating Language Tools for Writers and Learners of Swedish*. Ph.D. thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.
Carlberger, Johan and Viggo Kann. 1999. Implementing an efficient part-of-speech tagger. *Software Practice and Experience*, 29:815–832.
Cherry, Colin. 2008. Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of the 46th Annual Meeting of the ACL: Human Language Technologies*, pages 72–80, Columbus, OH.
Clark, Jonathan H., Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*, pages 176–181, Portland, OR.
Collins, Michael. 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*, pages 1–8, Philadelphia, PA.
Collins, Michael and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.
Cutting, Doug, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 133–140, Trento.
Doddington, George. 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, CA.
Dyer, Chris. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the NAACL*, pages 406–414, Boulder, CO.
Dyer, Chris. 2010. *A Formal Model of Ambiguity and its Applications in Machine Translation*. Ph.D. thesis, University of Maryland, USA.
El-Kahlout, İlknur Durgar 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, NY.
El Kholy, Ahmed and Nizar Habash. 2010. Techniques for Arabic morphological detokenization and orthographic denormalization. In *LREC 2010 Workshop on Language Resources and Human Language Technology for Semitic Languages*, pages 45–51, Valletta.
Fraser, Alexander. 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.
Friberg, Karin. 2007. Decomposing Swedish compounds using memory-based learning. In *Proceedings of the 16th Nordic Conference on Computational Linguistics (NODALIDA'07)*, pages 224–230, Tartu.
Fritzinger, Fabienne and Alexander Fraser. 2010. How to avoid burning ducks: Combining linguistic analysis and corpus statistics for German compound processing. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 224–234, Uppsala.
Hedlund, Turid. 2002. Compounds in dictionary-based cross-language information retrieval. *Information Research*, 7(2). Available at `http://InformationR.net/ir/7-2/paper128.html`. Accessed January 29, 2013.
Holmqvist, Maria, Sara Stymne, and Lars Ahrenberg. 2007. Getting to know Moses: Initial experiments on German–English factored translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 181–184, Prague.
Holz, Florian and Chris Biemann. 2008. Unsupervised and knowledge-free learning of compound splits and periphrases. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING)*, pages 117–127, Haifa.
Institut für Deutsche Sprache. 1998. Rechtschreibreform (Aktualisierte Ausgabe). IDS Sprachreport,

Extra-Ausgabe Dezember 1998. Mannheim, Germany.
Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit X*, pages 79–86, Phuket.
Koehn, Philipp, Abhishek Arun, and Hieu Hoang. 2008. Towards better machine translation quality for the German–English language pairs. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 139–142, Columbus, OH.
Koehn, Philipp 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.
Koehn, Philipp, 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, Demo and Poster Sessions*, pages 177–180, Prague.
Koehn, Philipp and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th Conference of the EACL*, pages 187–193, Budapest.
Kokkinakis, Dimitros. 2001. *A Framework for the Acquisition of Lexical Knowledge: Description and Applications*. Ph.D. thesis, Göteborg University, Sweden.
Kürschner, Sebastian. 2003. Von Volk-s-musik und Sport-ø-geist im Lemming-ø-land – Af folk-e-musik og sport-s-ånd i lemming-e-landet: Fugenelemente im Deutschen und Dänischen – eine kontrastive Studie zu einem Grenzfall der Morphologie. Master's thesis, Albert-Ludwigs-Universität, Freiburg, Germany.
Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, MA.
Langer, Stefan. 1998. Zur Morphologie und Semantik von Nominalkomposita. In *Tagungsband der 4. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS)*, pages 83–97, Bonn.
Larson, Martha, 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.
Lavie, Alon and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague.
Macherey, Klaus, Andrew Dai, David Talbot, Ashok Popat, and Franz Och. 2011. Language-independent compound splitting with morphological operations. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*, pages 1,395–1,404, Portland, OR.
Nießen, Sonja and Hermann Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 1,081–1,085, Saarbrücken.
Nießen, Sonja and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.
Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 160–167, Sapporo.
Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
Papineni, Kishore, 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, PA.
Popović, Maja and Hermann Ney. 2006. Statistical machine translation with a small amount of bilingual training data. In *5th LREC SALTMIL Workshop on Minority Languages*, pages 25–29, Genoa.
Popović, Maja, 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.

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

Riezler, Stefan 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, MI.

Sarawagi, Sunita and William W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 1,185–1,192, Cambridge, MA.

Schiller, Anne. 2005. German compound analysis with WFSC. In *Proceedings of the Finite State Methods and Natural Language Processing*, pages 239–246, Helsinki.

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester.

Schmid, Helmut and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 777–784, Manchester.

Simard, Michel, 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.

Sjöbergh, Jonas 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)*, pages 899–902, Lisbon.

Stolcke, Andreas. 2002. SRILM—an extensible language modeling toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904, Denver, CO.

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

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

Stymne, Sara and Nicola Cancedda. 2011. Productive generation of compound words in statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 250–260, Edinburgh.

Stymne, Sara 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.

Stymne, Sara, Maria Holmqvist, and Lars Ahrenberg. 2008. Effects of morphological analysis in translation between German and English. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 135–138, Columbus, OH.

Tapanainen, Paso and Timo Järvinen. 1997. A nonprojective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington, DC.

Taskar, Ben, Carlos Guestrin, and Daphne Koller. 2003. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS)*, pages 25–32, Vancouver.

Thorell, Olof. 1981. *Svensk ordbildningslära*. Esselte Studium, Stockholm, Sweden.

Tsochantaridis, Ioannis, Thorsten Joachims, Thomas Hofmann, and Altun Yasemin. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1,453–1,484.

Virpioja, Sami, 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.