Synthesizing Compound Words for Machine Translation

Austin Matthews **Eva Schlinger** **Alon Lavie** **Chris Dyer**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{austinma,eschling,alavie,cdyer}@cs.cmu.edu

Abstract

Most machine translation systems construct translations from a closed vocabulary of target word forms, posing problems for translating into languages that have productive compounding processes. We present a simple and effective approach that deals with this problem in two phases. First, we build a classifier that identifies spans of the input text that can be translated into a single compound word in the target language. Then, for each identified span, we generate a pool of possible compounds which are added to the translation model as "synthetic" phrase translations. Experiments reveal that (i) we can effectively predict what spans can be compounded; (ii) our compound generation model produces good compounds; and (iii) modest improvements are possible in end-to-end English–German and English–Finnish translation tasks. We additionally introduce KomposEval, a new multi-reference dataset of English phrases and their translations into German compounds.

1 Introduction

Machine translation systems make a closed-vocabulary assumption: with the exception of basic rules for copying unknown word types from the input to the output, they can produce words in the target language only from a fixed, finite vocabulary. While this is always a naïve assumption given the long-tailed distributions that characterize natural language, it is particularly challenging in languages such as German and Finnish that have productive compounding processes.

In such languages, expressing compositions of basic concepts can require an unbounded number of words. For example, English multiword phrases like *market for bananas*, *market for pears*, and *market for plums* are expressed in German with single compound words (respectively, as *Bananenmarkt*, *Birnenmarkt*, and *Pflaumenmarkt*). Second, while they are individually rare, the compounding is, on the whole, frequent in native texts (Baroni et al., 2002; Fritzinger and Fraser, 2010). Third, compounds are crucial for translation quality. Not only does generating them make the output seem more natural, but they are content-rich. Since each compound has, by definition, at least two stems, they are intuitively (at least) doubly important for translation adequacy.

Fortunately, compounding is a relatively regular process (as the above examples also illustrate), and it is amenable to modeling. In this paper we introduce a two-stage method (§2) to dynamically generate novel compound word forms given a source language input text and incorporate these as "synthetic rules" in a standard phrase-based translation system (Bhatia et al., 2014; Chahuneau et al., 2013; Tsvetkov et al., 2013). First, a binary classifier that examines each source-language sentence and labels each span therein with whether that span could become a compound word when translated into the target language. Second, we transduce the identified phrase into the target language using a word-to-character translation model. This system makes a closed vocabulary assumption, albeit at the character (rather than word) level—thereby enabling new word forms to be generated. Training data for these models is extracted from automatically aligned and compound split parallel corpora (§3).

We evaluate our approach on both intrinsic and extrinsic metrics. Since German compounds are relatively rare, their impact on the standard MT evaluation metrics (e.g., BLEU) is minimal, as we

show with an oracle experiment, and we find that our synthetic phrase approach obtains only modest improvements in overall translation quality. To better assess its merits, we commissioned a new test set, which we dub KomposEval (from the German word for a compound word, *Komposita*), consisting of a set of 1090 English phrases and their translations as German compound words by a professional English–German translator. The translator was instructed to produce as many compound-word translations as were reasonable (§4). This dataset permits us to evaluate our compound generation component directly, and we show that (i) without mechanisms for generating compound words, MT systems cannot produce the long tail of compounds; and (ii) our method is an effective method for creating correct compounds.

2 Compound Generation via Rule Synthesis

Suppose we want to translate the sentence

the market for bananas has collapsed .

from English into German. In order to produce the following (good) translation,

der bananenmarkt ist abgestürzt .

a phrase-based translation system would need to contain a rule similar to *market for bananas* $\rightarrow$ *bananenmarkt*. While it is possible that such a rule would be learned from parallel corpora using standard rule extraction techniques, it is likely that such a rule would not exist (unless the system were trained on the translation examples from this paper).

We solve the compound translation problem by "filling in" such missing rule gaps in the phrase table. The process takes place in two parts: first, identifying spans in the input that appear to be translatable as compounds (§2.1), and second, generating candidate compounds for each positively identified span (§2.2). Since synthesized rules compete along side rules which are learned using standard rule extraction techniques (and which are often quite noisy), our rule synthesis system can *overgenerate* rule candidates, a fact which we exploit in both phases.

2.1 Phase I: Classifying Compoundable Spans

Given a source sentence, we classify each span therein (up to some maximum length) as either compoundable or non-compoundable using independent binary predictions. Rather than attempting to hand-engineer features to represent phrases, we use a bidirectional LSTM to learn a fixed-length vector representation $\mathbf{h}_{i,j}$ that is computed by composing representations of the tokens $(f_i, f_{i+1}, \ldots, f_j)$ in the input sentence. The probability that a span is compoundable is then modeled as:

$$p(\text{compoundable? } | f_i, f_{i+1}, \ldots, f_j) = \sigma\left(\mathbf{w}^\top \tanh(\mathbf{V}\mathbf{h}_{i,j} + \mathbf{b}) + a\right),$$

where σ is the logistic sigmoid function, and $\mathbf{w}$, $\mathbf{V}$, $\mathbf{b}$, and a are parameters.

To represent tokens that are inputs to the LSTM, we run a POS tagger (Toutanova et al., 2003), and for each token concatenate a learned embedding of the tag and word. Figure 1 shows the architecture.

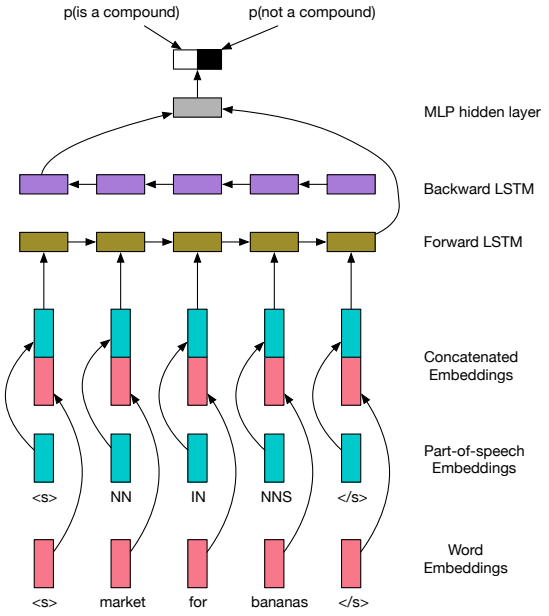

Figure 1: A graphical representation of the neural network used for classifying whether an input source phrase should or should not turn into a compound word in the target language

2.2 Phase II: Generating Compound Words

The second stage of our compound-generating pipeline is to generate hypothesis compound words for each source phrase that was identified as "compoundable" by the classifier just discussed. We do this by using a word-to-character–based machine translation system, which enables us to reuse a standard phrase-based decoder for compound generation.

2.2.1 Generation Model

The cornerstone of our generation approach is the forward and backward lexical translation tables learned by an unsupervised word aligner. We combine these two translation tables to create a word-to-character phrase table compatible with a standard decoder. This table allows our generator to know the correct translations of individual morphemes, but alone does not allow the generator to build full compound words.

To capture the small bits of "phonetic glue" (e.g., the *n* that occurs between *banane* and *markt* in the compound *bananenmarkt*) that may occur when generating compound words, we insert a special `SUF` symbol in between each pair of source words. This symbol will allow us to insert a small suffix in between the translations of source words.

Finally, we insert a special `END` symbol at the end of each source phrase. This symbol will allow the model to generate morphological variants due to suffixes indicating case, number, and agreement that only occur at the end of a whole compound word, but not in between the individual pieces. Some examples of all three types of rules are shown in Table 1.

2.2.2 Reordering and Word Dropping

We observe that in order to generate many compounds, including *bananenmarkts* from "market for bananas", a system must be able to both reorder and drop source words at will. Implemented naïvely, however, these allowances may produce invalid interleavings of source words and `SUF`/`END` tokens. For example, if we (correctly) drop the word "for" from our example, we might feed the decoder the sequence "market `SUF SUF` bananas `END`.

To disallow such bogus input sequences we disable all reordering inside the decoder, and instead encode all possible reorderings in the form of an input lattice (Dyer et al., 2008). Moreover, we allow the decoder to drop non-content words by skipping over them in the lattice. Each edge in our lattices contains a list of features, including the indices, lexical forms, and parts of speech of each word kept or dropped. Each possible sequence in the lattice also encodes features of the full path of source words kept, the full list of source words dropped, the parts of speech of the path and all dropped words, and the order of indices traversed.

With these constraints in place we can train the compound generator as though it were a normal MT system with no decode-time reordering.

3 Training

Our approach to generating compound word forms in translation has two stages. First, we build a classifier that chooses spans of source text that could produce target compounds. Second, we build a compound generator that outputs hypothesis word forms, given a source phrase. We will detail each of these steps in turn.

3.1 Extracting Compounds from Bitext

In order to learn to generate compound words we naturally require training data. Ideally we would like a large list of English phrases with their natural contexts and translations as German compounds. Of course virtually no such data exists, but it is possible to extract from parallel data, using a technique similar to that used by Tsvetkov and Wintner (2012).

To this end, we take our tokenized bitext and pass it through Dyer (2009)'s German compound splitter. We then align the segmented variant using the `fast_align` tool in both the forward and reverse directions, which produces both word alignments and lexical translation tables, which give the probability of a compound part given an English phrase. We then symmetrize the produced pair of alignments with the intersection heuristic. This results in a sparse alignment in which each target word is aligned to either 0 or 1 source words. We then undo any splits performed by the compound splitter, resulting in a corpus where the only words aligned many-to-one are precisely well-aligned compounds.

This process produces two crucially important data. First, a list of English phrase pairs that may become compound words in German on which we train our classifier. Second, the lexical translation tables, trained on compound split German data, which form the basis of our generation approach.

3.2 Training the Compoundability Classifier

The network is trained to maximize cross-entropy of its training data using the Adam optimizer (Kingma and Ba, 2014) until performance on a held-out dev set stops improving.

Due to the fact that we only need to represent the "compoundability" of each source-language word, and not its full semantics, we find that very small (10-dimensional) word and POS em-

Source	Target	Non-Zero Features
bananas	b a n a n e	$\phi_{fwd} = -0.495208$ $\phi_{rev} = -0.455368$
market	m a r k t	$\phi_{fwd} = -0.499118$ $\phi_{rev} = -0.269879$
SUF	n	$\phi_{fwd} = -3.718241$ $\phi_{\text{uses\_suf\_n}} = 1.0$
END	s	$\phi_{fwd} = -2.840721$ $\phi_{\text{uses\_end\_s}} = 1.0$

Table 1: A fragment of the word-to-character rules used in the compound generation system.

beddings work well. The recurrent part of the neural network uses two-layer LSTM (Hochreiter and Schmidhuber, 1997) cells with the hidden layer size set to 10. The final MLP's hidden layer size is also set to 10.

The training data is processed such that each span of length two to four is considered one training example, and is labeled as positive if it is well-aligned (Brown et al., 1993) to a single German compound word. Since most spans do not translate as compounds, we are faced with an extreme class imbalance problem (a ratio of about 300:1). We therefore experiment with down sampling the negative training examples to have an equal number of positive and negative examples.

3.3 Training the Compound Generation Model

As a translation model, there are two components to learning the translation system: learning the rule inventory and their features (§3.3.1) and learning the parameters of the generation model (§3.3.2).

3.3.1 Learning Word to Character Sequence Translation Rules

The possible translations of SUF and END are learned from the list of positive training examples extracted for our classifier. For each example, we find all the possible ways the source words could translate, in accordance with our translation table, into nonoverlapping substrings of the target word. Any left over letters in between pieces become possible translations of SUF, while extra letters at the end of the target string become possible translations of END. Probabilities for each translation are estimated by simply counting and normalizing the number of times each candidate was seen. See Figure 2 for an example of this splitting process.

3.3.2 Learning Generator Feature Weights

Since the generator model is encoded as a phrase-based machine translation system, we can train it using existing tools for this task. We choose to train using MIRA (Crammer and Singer, 2003), and use a 10-gram character-based language model trained on the target side of the positive training examples extracted for the classifier.

4 KomposEval Data Set

To evaluate our compound generator we needed a dataset containing English phrases that should be compounded along with their German translations. To the best of our knowledge, no substantial human-quality dataset existed, so we created one as part of this work.

We took our list of automatically extracted (English phrase, German compound) pairs and manually selected 1090 of them that should compound. We then asked a native German speaker to translate each English phrase into German compounds, and to list as many possibile compound translations as she could think of. The result is a test set consisting of 1090 English phrases, with between 1 and 5 possible German compound translations for each English phrase. This test set is published as supplementary material with this article. Some example translations are shown in Table 2.

Source phrase	Reference(s)
transitional period	Übergangsphase
	Übergangsperiode
	Übergangszeitraum
Chamber of deputies	Abgeordnetenhaus
	Abgeordnetenkammer
self-doubt	Selbstzweifel

Table 2: Examples of human-generated compounds from the KomposEval data set

5 Experiments

Before considering the problem of integrating our compound model with a full machine translation system, we perform an intrinsic evaluation of each of the two steps of our pipeline.

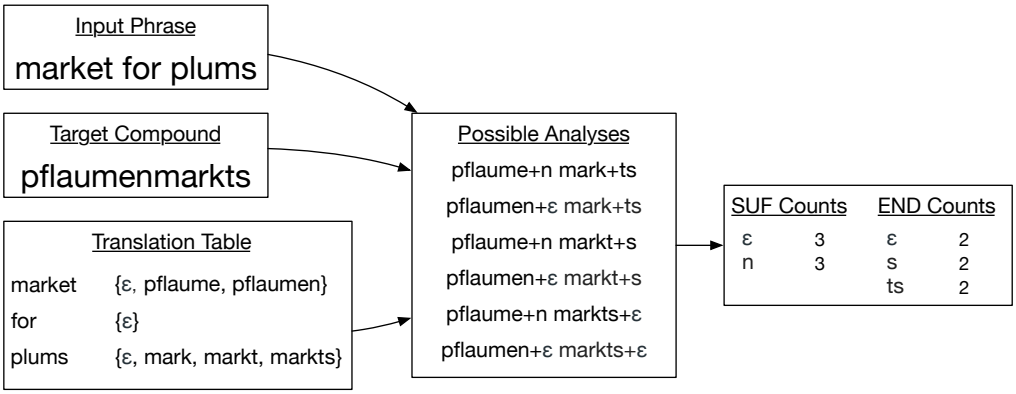

Figure 2: Decomposition of a target compound into possible analyses, given a source phrase and a morpheme-level translation table. This process allows us to learn the "phonetic glue" that can go in between morphemes, as well as the inflections that can attach to the end of a compound word.

5.1 Classifier Intrinsic Evaluation

We evaluate the effectiveness of our classifier, by measuring its precision and recall on the two held out test sets described in §2.1 taken from two language pairs: English–German and English–Finnish. Furthermore, we show results both with down-sampling (balanced data set) and without down-sampling (unbalanced data set).

Our classifier can freely trade off precision and recall by generalizing its requirement to call an example positive from $p(\text{compound} \mid \text{span}) > 0.5$ to $p(\text{compound} \mid \text{span}) > \tau$, for $\tau \in (0, 1)$, allowing us to report full ROC curves (Figure 3).

We find that our best results for the unbalanced cases come at $\tau = 0.24$ for German and $\tau = 0.29$ for Finnish, with F-scores of 20.1% and 67.8%, respectively. In the balanced case, we achieve 67.1% and 97.0% F-scores with $\tau = 0.04$ and $\tau = 0.57$ on German and Finnish respectively.

5.2 Generator Instrinsic Evaluation

To evaluate our compound generator, we fed it the source side of our newly created KomposEval corpus and had it output a 100-best list of hypotheses translations for each English phrase. From this we are able to compute many intrinsic quality metrics. We report the following metrics:

- Mean reciprocal rank (MRR); which is one divided by the average over all segments of the position that the reference translation appears in our k-best list.
- Character error rate (CER), or the average number of character-level edits that are required to turn our 1-best hypothesis into the nearest of the reference translations.
- Precision at 1, 5, and 10, which indicate what percentage of the time a reference translation can be found in the top 1, 5, or 10 hypotheses of our k-best list, respectively.

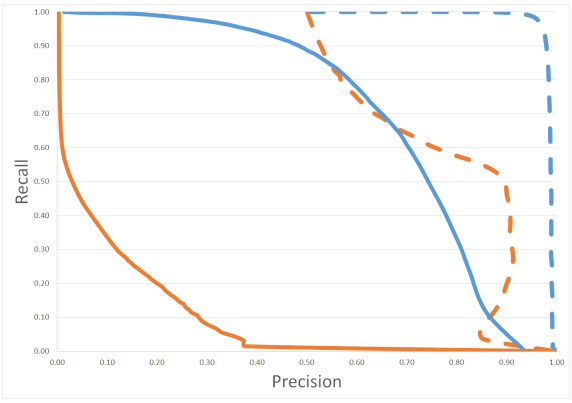

Figure 3: Receiver operating characteristic (ROC) curves for our compound classifier for two languages: German (red) and Finnish (blue). Unbalanced test set results are shown with solid lines. Balanced test set results are shown with dashed lines.

These results can be found in Table 3. We compare to a naïve baseline that is just a standard English–German phrase-based translation system with no special handling of compound word forms. We immediately see that the baseline system's is simply unable to generate most of the compound words in the test set, resulting in extraordinarily low metric scores across the board. Its one saving grace is its tolerable CER score, which shows that the system is capable of gen-

	MRR ↑	CER ↓	P@1 ↑	P@5 ↑	P@10 ↑
Baseline	<0.01	3.305	0%	0%	<0.01%
Our model	0.7004	2.506	61.38%	81.47%	84.31%

Table 3: Mean reciprocal rank, character error rate, and precision at K statistics of our baseline MT system and our compound generator.

erating the correct morphemes, but is failing to correctly adjoin them and add the phonological glue required to produce a well-formed compound word. Our system, on the other hand, is capable of reaching at least one of the five references for every single sentence in the test set, and has a reference translation in the top 5 hypotheses in its k-best list over 80% of the time.

Qualitatively, the compounds generated by our model are remarkably good, and very understandable. Major error classes include incorrect word sense, non-compositional phrases, and special non- concatenative effects at word boundaries. An example of each of these errors, along with some examples of good compound generation can be found in Table 4.

5.3 Extrinsic Translation Evaluation

Finally, we use our compound generator as part of a larger machine translation pipeline. We run our compound span classifier on each of our translation system's tune and test sets, and extract our generator's top ten hypotheses for each of the postively identified spans. These English phrases are then added to a synthetic phrase table, along with their German compound translations, and two features: the compound generator's score, and an indicator feature simply showing that the rule represents a synthetic compound. Table 5 shows some example rules of this form. The weights of these features are learned, along with the standard translation system weights, by the MIRA algorithm as part of the MT training procedure.

The underlying translation system is a standard Hiero (Chiang et al., 2005) system using the cdec (Dyer et al., 2010) decoder, trained on all constrained-track WMT English–German data as of the 2014 translation task. Tokenization was done with `cdec`'s `tokenize-anything` script. The first character of each sentence was down cased if the unigram probability of the downcased version of the first word was higher than that of the original casing. Word alignment was performed using `cdec`'s `fast_align` tool, and symmetrized using the `grow-diag` heuristic. Training is done using cdec's implementation of the MIRA algorithm. Evaluation was done using MultEval (Clark et al., 2011). A 4-gram language model was estimated using KenLM's `lmplz` tool (Heafield et al., 2013).

In addition to running our full end-to-end pipeline, we run an oracle experiment wherein we run the same pre-processing pipeline (compound splitting, bidirectionally aligning, intersecting, and de-splitting) on each test set to identify which spans do, in fact, turn into compounds, as well as their ideal translations. We then add grammar rules that allow precisely these source spans to translate into these oracle translations. This allows us to get an upper bound on the impact compound generation could have on translation quality.

The results, summarized in Table 6 and Table 7, show that adding these extra compounds has little effect on metric scores compared to our baseline system. Nevertheless, we believe that the qualitative improvements of our methods are more significant than the automatic metrics would indicate. Our method targets a very specific problem that pertains only to dense content-bearing target words that humans find very important. Moreover, BLEU is unable to reasonably evaluate improvements in these long tail phenomena, as it only captures exact lexical matches, and because we are purposely generating fewer target words than a standard translation system.

6 Related Work

Most prior work on compound generation has taken a different approach from the one advocated here, first translating the source language into a morphologically analyzed and segmented variant of the target language, and then performing morphological generation on this sequence (Cap et al., 2014; Irvine and Callison-Burch, 2013; Denkowski et al., 2013; Clifton and Sarkar, 2011; Stymne and Cancedda, 2011).

Requesting multiple translations from a translator has been used in the past, most notably to create HyTER reference lattices (Dreyer and Marcu, 2012). However, in contrast to full-sentence translations the space of possible grammatical compounds is far smaller, substantially simplifying our task.

The splitting of German compound phrases for

Input	Hypothesis	Reference	Comments
cheese specialities	Fachkäse	Käsespezialitäten	Wrong sense of "specialties"
band-aid	Band-hilfe	(Should not compound)	Idiosyncratic meaning
church towers	Kirchentürme	Kirchtürme	Extra word-internal case marking
sugar beet farmers	Zuckerrübenbauern	Zuckerrübenbauern	Perfect
tomato processing	Tomatenverarbeitung	Tomatenverarbeitung	Perfect
generation of electricity	Stromerzeugung	Stromerzeugung	Perfect, including reordering

Table 4: Examples of erroneous (top) and correct (bottom) compounds generated by our system

Source	Target	Non-Zero Features
`market for bananas`	`bananenmarkt`	$\phi_{Compound} = 1$ $\phi_{Score} = -38.9818$
`market for bananas`	`bananenmarktes`	$\phi_{Compound} = 1$ $\phi_{Score} = -49.8976$
`market for bananas`	`marktordnung`	$\phi_{Compound} = 1$ $\phi_{Score} = -53.2197$
`market for bananas`	`bananenmarkts`	$\phi_{Compound} = 1$ $\phi_{Score} = -54.4962$
`market for bananas`	`binnenmarkt`	$\phi_{Compound} = 1$ $\phi_{Score} = -57.6816$

Table 5: Examples synthetic rules dynamically added to our system to translate the phrase "market for bananas" into a German compound word. Note that we correctly generate both the nominative form (with no suffix) and the genitive forms (with the -s and -es suffixes).

		BLEU ↑	METR ↑	TER ↓	Len
WMT2012	Baseline	16.2	34.5	64.8	94.1
	+Our Compounds	16.3	34.6	64.9	94.2
	+Oracle Compounds	16.9	35.2	64.6	95.5
WMT2013*	Baseline	18.8	37.3	62.1	93.6
	+Our Compounds	18.9	37.5	62.3	96.7
	+Oracle Compounds	19.7	38.2	61.9	97.6
WMT2014	Baseline	19.6	38.9	64.3	103.5
	+Compounds	19.6	39.0	64.5	103.9
	+Oracle Compounds	21.7	40.9	61.1	100.6

Table 6: Improvements in English–German translation quality using our method of compound generation on WMT 2012, 2013, and 2014. * indicates the set used for tuning the MT system.

		BLEU ↑	METR ↑	TER ↓	Len
Dev*	Baseline	12.3	29.0	72.7	96.5
	+Our Compounds	12.3	29.1	72.8	96.8
DevTest	Baseline	11.4	29.9	71.6	96.2
	+Our Compounds	11.6	30.1	71.5	96.4
Test	Baseline	10.8	28.4	73.4	96.7
	+Our Compounds	10.9	28.5	73.3	96.9

Table 7: Improvements in English–Finnish translation quality using our method of compound generation on WMT 2014 tuning, devtest, and test sets. * indicates the set used for tuning the MT system.

translation from German into English has been addressed by Koehn and Knight (2001) and Dyer (2009). They elegantly solve the problem of having a large, open vocabulary on the source side by splitting compound words into their constituent morphemes and translating German into English at the morpheme level. Their approach works excellently when translating *out* of a compounding language, but is unable to generate novel compound words in the target language without some sort of post processing.

Dynamic generation of compounds in a target language using such post processing has been examined in the past by Cap et al. (2014) and Clifton and Sarkar (2011). Both perform compound splitting on their parallel data, train a morpheme-based translation system, and then stitch compound words back together using different models. While effective, their approach runs into difficulties if the morphemes that should compound get separated by the reordering model during the translation process. Both address this using more complicated models, whereas our holistic approach handles this problem seamlessly.

Stymne (2012) gives an excellent taxonomy of compound types in Germanic languages, and discusses many different strategies that have been used to split and merge them for the purposes of machine translation. She identifies several difficulties with the split-translate-merge approach and points out some key subtleties, such as handling

of bound morphemes that never occur outside of compounds, that one must bear in mind when doing translation to or from compounding languages.

The idea of using entirely character-based translation systems was introduced by Vilar et al. (2007). While their letter-level translation system alone did not outperform standard phrase-based MT on a Spanish–Catalan task, they demonstrated substantial BLEU gains when combining phrase- and character-based translation models, particularly in low resource scenarios.

7 Conclusion

In this paper we have presented a technique for generating compound words for target languages with open vocabularies by dynamically introducing synthetic translation options that allow spans of source text to translate as a single compound word. Our method for generating such synthetic rules decomposes into two steps. First an RNN classifier detects compoundable spans in the source sentence. Second, a word-to-character machine translation system translates the span of text into a compound word.

By dynamically adding compound words to our translation grammars in this way we allow the decoder, which is in turn informed by the language model, to determine which, if any, of our hypothesized compounds look good in context. Our approach does away with the need for post processing, and avoids complications caused by reordering of morphemes in previous approaches. However, this technique relies heavily on a strong target language model. Therefore, one important extension of our work is to further study the interaction between our model and the underlying language model.

In addition to our generation technique we have presented a new human-quality data set that specifically targets compounding and use it to demonstrate tremendous improvements in our translation system's ability to correctly generalize from compound words found in parallel text to match human translations of unseen compoundable phrases.

References

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Predicting the components of german nominal compounds. In *ECAI*, pages 470–474.

Archna Bhatia, Chu-Cheng Lin, Nathan Schneider, Yulia Tsvetkov, Fatima Talib Al-Raisi, Laleh Roostapour, Jordan Bender, Abhimanu Kumar, Lori Levin, Mandy Simons, et al. 2014. Automatic classification of communicative functions of definiteness. Association for Computational Linguistics.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Fabienne Cap, Alexander Fraser, Marion Weller, and Aoife Cahill. 2014. How to produce unseen teddy bears: Improved morphological processing of compounds in SMT. In *Proc. EACL*.

Victor Chahuneau, Eva Schlinger, Noah A Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases.

David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. 2005. The hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 779–786. Association for Computational Linguistics.

Jonathan H Clark, 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 Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 32–42. Association for Computational Linguistics.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

Waleed Ammar Victor Chahuneau Michael Denkowski, Greg Hanneman, Wang Ling Austin Matthews Kenton Murray, Nicola Segall Yulia Tsvetkov, and Alon Lavie Chris Dyer. 2013. The cmu machine translation systems at wmt 2013: Syntax, synthetic translation options, and pseudo-references. In *8th Workshop on Statistical Machine Translation*, page 70.

Markus Dreyer and Daniel Marcu. 2012. Hyter: Meaning-equivalent semantics for translation evaluation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 162–171. Association for Computational Linguistics.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. Technical report, DTIC Document.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL*.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 406–414. Association for Computational Linguistics.

Fabienne Fritzinger 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. Association for Computational Linguistics.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *HLT-NAACL*, pages 518–523.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Kevin Knight. 2001. Knowledge sources for word-level translation models. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 27–35.

Sara Stymne and Nicola Cancedda. 2011. Productive generation of compound words in statistical machine translation. In *Proc. WMT*.

Sara Stymne. 2012. Text harmonization strategies for phrase-based statistical machine translation.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Yulia Tsvetkov and Shuly Wintner. 2012. Extraction of multi-word expressions from small parallel corpora. *Natural Language Engineering*, 18(04):549–573.

Yulia Tsvetkov, Chris Dyer, Lori Levin, and Archna Bhatia. 2013. Generating English determiners in phrase-based translation with synthetic translation options. In *Proc. WMT*.

David Vilar, Jan-T Peter, and Hermann Ney. 2007. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39. Association for Computational Linguistics.