Integrating Morphological Desegmentation into Phrase-based Decoding

Mohammad Salameh[†] **Colin Cherry**[‡] **Grzegorz Kondrak**[†]

[†]Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{msalameh,gkondrak}@ualberta.ca

[‡]National Research Council Canada
1200 Montreal Road
Ottawa, ON, K1A 0R6, Canada
Colin.Cherry@nrc-cnrc.gc.ca

Abstract

When translating into a morphologically complex language, segmenting the target language can reduce data sparsity, while introducing the complication of desegmenting the system output. We present a method for decoder-integrated desegmentation, allowing features that consider the desegmented target, such as a word-level language model, to be considered throughout the entire search space. Our results on a large-scale, English to Arabic translation task show significant improvement over the 1-best desegmentation baseline.

1 Introduction

State-of-the-art systems for translating into morphologically complex languages, such as Arabic, employ morphological segmentation of the target language in order to reduce data sparsity and improve translation quality. For example, the Arabic word لدولتهم *ldwlthm* "for their country" is segmented into the prefix ل+ *l*+ "for", the stem دولة *dwlp* "country" and the suffix +هم +*hm* "their." The segmentation sometimes involves performing orthographic normalizations, such as transforming the stem-final *t* to *p*. The result is not only a reduction in the number of word types, but also better token-to-token correspondence with the source language.

Morphological segmentation is typically performed as a pre-processing step before the training phase, which results in a model that translates the source language into segmented target language. *Desegmentation* is the process of transforming the segmented output into a readable word sequence, which can be performed using a table lookup combined with a small set of rules. Desegmentation is usually applied to the 1-best output of the decoder. However, this pipeline suffers from error propagation: errors made during decoding cannot be corrected, even when desegmentation results in an illegal or extremely unlikely word. Two principal types of solutions have been proposed for this problem: *rescoring* and *phrase-table desegmentation*.

The rescoring approach desegments either an n-best list (Oflazer and Durgar El-Kahlout, 2007) or lattice (Salameh et al., 2014), and then re-ranks with features that consider the desegmented word sequence of each hypothesis. Rescoring features include the score from an unsegmented target language model and contiguity indicators that flag target words that were translated from contiguous source tokens. Rescoring widens the desegmentation pipeline, allowing desegmentation features to reduce the number of translation errors. However, these features are calculated for only a subset of the search space, and the extra rescoring step complicates the training and translation processes.

Phrase-table desegmentation (Luong et al., 2010) also translates into a segmented target language, but alters training to perform word-boundary-aware phrase extraction. The extracted phrases are constrained to contain only complete target words, without any *dangling affixes*. With this restriction in place, the phrase table can be desegmented before decoding begins, allowing the decoder to track features over both the segmented and desegmented target. This ensures that desegmentation features are integrated into the complete search space, and

Proceedings of NAACL-HLT 2016, pages 1175–1180,
San Diego, California, June 12-17, 2016. ©2016 Association for Computational Linguistics

side-steps the complications of rescoring. However, Salameh et al. (2015) show experimentally that these benefits are not worth giving up phrase-pairs with dangling affixes, which are eliminated by word-boundary-aware phrase extraction.

We present a method for decoder-integrated desegmentation that combines the strengths of these two approaches. Like a rescoring approach, it places no restrictions on what morpheme sequences can appear in the target side of a phrase pair. Like phrase-table desegmentation, its desegmentation features are integrated directly into decoding and considered throughout the entire search space. We achieve this by augmenting the decoder to desegment hypotheses on the fly, allowing the inclusion of an unsegmented language model and other features. Our results on a large-scale, NIST-data English to Arabic translation task show significant improvements over the 1-best desegmentation baseline, and match the performance of the state-of-the-art lattice desegmentation approach of Salameh et al. (2014), while eliminating the complication and cost of its rescoring step. Our approach is implemented as a single stateful feature function in Moses (Koehn et al., 2007), which we will submit back to the community.

2 Method

Our approach extends the multi-stack phrase-based decoding paradigm to enable the extraction of word-level features inside morpheme-segmented models.[1] We assume that the target side of the parallel corpus has been segmented into morphemes with prefixes and suffixes marked.[2] This allows us to define a *complete word* as a maximal morpheme sequence consisting of 0 or more prefixes, followed by at most one stem, and then 0 or more suffixes.

We also assume access to a desegmentation function that takes as input a morpheme sequence matching the above definition, and returns the corresponding word as output. Depending on the complexity of the segmentation, desegmentation can be achieved through simple concatenation, a small set of rules, a statistical table (Badr et al., 2008), or a statistical transducer (Salameh et al., 2013). El Kholy and Habash (2012) provide an extensive study on the influence of segmentation and desegmentation on English-to-Arabic SMT. In this work, we adopt the Table+Rules technique of El Kholy and Habash (2012) for English-Arabic SMT. The technique relies on a look-up table that stores mappings of segmented-unsegmented forms, and falls back on manually crafted rules for segmented sequences not found in the table. When a segmented form has multiple desegmentation options available in the table, we select the most frequent option.

The output of a phrase-based decoder is built from left to right, and at each step, a hypothesis is expanded with a phrasal translation of a previously uncovered source segment. We augment this process with *in-decoder desegmentation*, which monitors the target sequence of each translation hypothesis as it grows, detecting morpheme sequences that correspond to complete words and desegmenting them on the fly to generate new features. This is described in detail in Section 2.1.

The task of determining whether a word is complete is non-trivial. We are never sure if we will see another suffix as we expand the hypothesis, so we can only recognize a complete word as we begin the next word. For example, take hyp1 in Figure 1. This hypothesis ends with a stem *nšr*, which may end a complete word, as is the case when we expand to hyp2, or may represent a word that is still in progress, which occurs as we extend to hyp3. This means that the word-based scoring of the morpheme sequence *l+ nšr* must be delayed or approximated until we know what follows. A related challenge involves scoring phrase-pairs out of context, as is required for future-cost estimates. Take, for example, the target phrase *+h AfkAr* added by hyp3 in Figure 1. Without the context, we have insufficient information at the left boundary to score *+h* with word-based models, while *AfkAr* at the right boundary may or may not form a complete word. Here, there is no choice but to approximate. The quality of these approximations and the length of our delays will determine how effective our new features will be when incorporated into beam search.

[1] The ideas presented here could also be applied to hierarchical decoding, which would require generalizing them to account for right context as well as left.

[2] Throughout this paper, we use a token-final "+" to denote a prefix, and a token-initial "+" for a suffix.

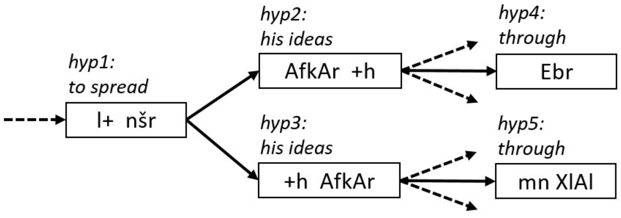

Figure 1: Decoding the Arabic translation of the phrase *"to spread his ideas through"*.

2.1 Decoder Integration

A typical phrase-based decoder represents a hypothesis with a state that contains the information to guide search and calculate features, such as the source coverage vector and the target context for the language model. Hypotheses with identical states can be recombined to improve search efficiency. We augment the state with two structures: (1) a buffer Q containing all of the morphemes that contribute to the current word in progress, represented as a queue of tokens; and (2) n-gram context C for the word-level target language model. The search's initial state begins with an empty Q and with *n-1* beginning-of-sentence tokens in *C*.

When a state is extended with a target phrase *P*, we update the in-decoder desegmentation structures Q and *C* with Algorithm 1. Tokens are appended to Q until a token t would begin a new word, at which point the tokens from Q are desegmented and the resulting word is used to calculate features and update the target context. Following the lower decoding path in Figure 1, Q would be emptied and desegmented first during hyp3 when $t =$ *AfkAr*, calculating features for $W =$ *lnšrh*.

The main cost of in-decoder desegmentation comes from maintaining the context necessary to evaluate the n-gram, word-level language model. As each desegmented word in C will correspond to at least one segmented token, the system's effective language-model order in terms of segmented tokens will frequently be much larger than n. Storing larger language-model contexts make it less likely that states will be equal to one another, which reduces the amount of recombination the system can do, and increases the number of states that must be expanded during search.

Algorithm 1 Desegmentation State Update

```
Input: State variables $Q$, $C$
Input: Extending phrase $P$
for each token $t$ in $P$ do
    if $t$ cannot continue the word in Q then
        $W$ = Desegmentation of tokens in $Q$
        Extract word-level features for $W$
         (Word-level LM score is $p(W|C)$)
        Update current feature vector
        Update $C$ with $W$
        Empty $Q$
    Append token $t$ to $Q$
```

2.2 Delayed and Optimistic Scoring

In the above approach, desegmentation and feature scoring are applied only when a complete word is formed. We refer to this as **delayed scoring** because the features for a token are not applied until other tokens have been added to the hypothesis. For example, in Figure 1, the tokens *l+ nšr* added in hyp1 are not evaluated with word-level features until hyp2 or hyp3 completes the word. This delay results in inaccurate scoring of hypotheses, as the cost from these tokens is hidden until Q is emptied. These inaccuracies can lead to poor pruning choices and search errors during beam search.

Alternatively, we can perform **optimistic scoring**, which tries to score the contents of Q as early as possible. In this case, we assume that the contents of Q form a complete word, without waiting for the next token to confirm it. With each hypothesis extension, when the last token in P is processed and added to the queue, we desegment the contents of Q and extract features, but without emptying Q. The scores of these features are cached in a variable S that does not affect recombination, as the scores are deterministic given Q, C and the model. When a later token confirms the end of the word, we subtract S from the scores derived from the actual desegmented word, to account for our earlier approximation. Note that for a Q containing only a prefix, we must still delay scoring.

2.3 Features

Three features are extracted from each desegmented form: an unsegmented language model, contiguity indicators, and a desegmented word penalty.

The unsegmented n-gram language model scores W in the context of C, as shown in Algorithm 1. This language model will heavily penalize malformed Arabic words, as they will appear as out-of-vocabulary items. Furthermore, it will evaluate well-formed Arabic words in a larger, word-level context, complementing the morpheme-level n-gram language model that is naturally included in SMT systems built over a segmented target.

We also implement the contiguity features proposed by Salameh et al. (2014). These indicators check if the desegmented form is generated from a contiguous block of source tokens, a block with 1 discontiguity, or a block with multiple discontiguities. These features enable the decoder to prefer desegmented words whose component segments were translated from contiguous or nearly contiguous source sequences. This encourages the system to select a more local, and hopefully safer, translation path when possible.

Finally, most phrase-based decoders incorporate a "word penalty" feature that counts the number of target tokens in a hypothesis. When the target language has been segmented into morphemes, this actually corresponds to a morpheme penalty. However, with in-decoder desegmentation, we now have the option to count either words or morphemes. There is reason to believe that by counting words, instead of morphemes, we will give the system greater control over the length of its output word sequence, which is particularly relevant because of BLEU's brevity penalty. We try both options in our experiments. Unfortunately, the obvious solution of including two features, a word count and a morpheme count, did not perform well during development.

2.4 Future costs

For future cost estimates, we must also provide out-of-context feature scores for each phrase-pair in our system. To do so, we ignore suffixes appearing at the beginning of a target phrase and prefixes appearing at the end. We assume that the remaining tokens form complete words, and desegment and score them to provide out-of-context scores. We also consider dangling affixes as half words, with a count of 0.5, for out-of-context scoring of the word penalty feature.

3 Experiments

We use the NIST 2012 dataset (1.49 million sentence pairs excluding UN pairs) to train an English-to-Arabic system. The system is tuned with the NIST 2004 (1353 pairs) evaluation set and tested using NIST 2005 (1056 sentences) and the newswire portion of NIST 2008 (813 pairs) and NIST 2009 (586 pairs). As there are multiple English reference translations provided for these evaluation sets, we use the first reference as our source text.

The Arabic part of the training set is morphologically segmented and tokenized by MADA 3.2 (Habash et al., 2009) using the Penn Arabic Treebank (PATB) segmentation scheme. Variants of Alif and Ya characters are uniformly normalized. We generate a desegmentation table from the Arabic side of the training data by collecting mappings of segmented forms to surface forms.

We align the parallel data with GIZA++ (Och et al., 2003), and decode with Moses (Koehn et al., 2007). The decoder's log-linear model uses a standard feature set, including four phrase table scores, six features from a lexicalized distortion model, along with a phrase penalty and a distance-based distortion penalty. KN-smoothed 5-gram language models are trained on both the segmented and unsegmented views of the target side of the parallel data. We experiment with word penalties based on either morphemes or desegmented words. The decoder uses Moses' default search parameters, except for the maximum phrase length, which is set to 8, and the translation table limit, which is set to 40. The decoder's log-linear model is tuned with MERT (Och, 2003) using unsegmented Arabic reference translations. When necessary, we desegment our 100-best-lists before MERT evaluates each hypothesis. We evaluate with BLEU (Papineni et al., 2002) measured on unsegmented Arabic, and test statistical significance with multeval (Clark et al., 2011) over 3 tuning replications.

We test four systems that differ in their desegmentation approach. The **NoSegm.** baseline involves no segmentation. The **One-best** baseline translates into segmented Arabic and desegments the decoder's 1-best output. The **Lattice** system is the lattice-desegmentation approach of Salameh et al. (2014). We implement our **in-Decoder** desegmenta-

System	WP	mt05	mt08	mt09
NoSegm.	word	33.2	18.6	25.6
One-best	morph.	33.8	19.1	26.8
Lattice	morph.	34.4	**19.7**	**27.4**
Delayed	morph.	34.1	19.4	27.0
	word	34.1	19.5	26.8
Optimistic	morph.	34.2	19.6	27.2
	word	**34.5**	**19.7**	27.2

Table 1: Evaluation of the desegmentation methods using BLEU score. Both Delayed and Optimistic refer to in-Decoder Desegmentation method used. WP shows whether Word Penalty feature is based on a complete desegmented word or a morpheme.

tion approach as a feature functions in Moses, testing scoring variants (delayed vs. optimistic), and word penalty variants (morpheme vs. word).

Table 1 shows the results on three NIST test sets, each averaged over 3 tuning replications. The lattice approach is significantly better than the 1-best system, which in turn is significantly better than the unsegmented baseline. Our **Optimistic** in-decoder approach with word penalty calculated on word tokens is significantly ($p < 0.05$) better than the 1-best approach, and effectively matches the quality of the more complex lattice approach.

All of the systems with word-level features improve over 1-best desegmentation, as their features penalize desegmentations resulting in illegal words or unlikely word sequences. We see a small, consistent benefit from optimistic scoring. Error analysis reveals that translations with many consecutive stems benefit the most from this variant, which makes sense, as our approximations would be exact in these cases. Using a word penalty calculated on word tokens appears to work slightly better on average than one calculated on morphemes.

Typically, one would hope to surpass a rescoring approach with decoder integration; however, our lattice implementation fully searches its lattice, even if composition with the word-level language model would cause the lattice to explode in size. That is, lattice desegmentation has an advantage, as it trades time-efficiency for a perfect search that ignores the complexity introduced by expanded n-gram context. A lattice beam search that dynamically calculates word-level language model scores while pruning away unlikely paths would provide a more fair, and more efficient, comparison point.

Lattice rescoring also involves many steps, requiring one to train and tune a complete segmented system with segmented references, then desegment lattices and compose them with a word LM, and then tune a lattice rescorer on unsegmented references. In contrast, our system is implemented as a single decoder feature function in Moses.[3] This one function replaces the lattice desegmentation, LM composition, and lattice rescoring steps, greatly simplifying the translation pipeline.

4 Conclusions and Future Work

We have presented a method for in-decoder desegmentation, which allows a phrase-based decoder to simultaneously consider both segmented and desegmented views of the target language. We have shown that this approach outperforms 1-best desegmentation, and matches the performance of lattice desegmentation, while eliminating the complication of its lattice transformation and rescoring steps.

We are interested in building an unsegmented, word-level language model that can provide meaningful estimates for morphological segments, which would improve scoring for out-of-context phrases and incomplete words. Also, our system currently considers only the most likely desegmentation of each segmented word. Inspired by the disambiguating desegmentation system of El Kholy and Habash (2012), we would like to extend our system to propose multiple desegmentation candidates for each word, and allow the decoder to select the correct form using its other features.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

[3] This function references a desegmentation table and an unsegmented language model, which are needed to carry out Algorithm 1. Even though it is conceptually one function, it produces a vector of feature scores, producing the various features described in Section 2.3.

References

Ibrahim Badr, Rabih Zbib, and James Glass. 2008. Segmentation for English-to-Arabic statistical machine translation. In *Proceedings of ACL*, pages 153–156.

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 ACL*, pages 176–181.

Ahmed El Kholy and Nizar Habash. 2012. Orthographic and morphological processing for English—Arabic statistical machine translation. *Machine Translation*, 26(1-2):25–45, March.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. Mada+tokan: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

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

Minh-Thang Luong, Preslav Nakov, and Min-Yen Kan. 2010. A hybrid morpheme-word representation for machine translation of morphologically rich languages. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 148–157, Cambridge, MA, October.

Franz Josef Och, Hermann Ney, Franz Josef, and Och Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29.

Franz Joseph Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Kemal Oflazer and Ilknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 25–32, Prague, Czech Republic, June.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Mohammad Salameh, Colin Cherry, and Grzegorz Kondrak. 2013. Reversing morphological tokenization in English-to-Arabic SMT. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 47–53, Atlanta, Georgia, June.

Mohammad Salameh, Colin Cherry, and Grzegorz Kondrak. 2014. Lattice desegmentation for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 100–110.

Mohammad Salameh, Colin Cherry, and Grzegorz Kondrak. 2015. What matters most in morphologically segmented smt models? In *Proceedings of the Ninth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 65–73, Denver, Colorado, USA, June. Association for Computational Linguistics.