# Semi-Automatic Quasi-Morphological Word Segmentation for Neural Machine Translation

Jānis Zuters, Guntis V. Strazds, and Kārlis Immers

University of Latvia ­ Faculty of Computing

19 Raina blvd., LV-1586 Riga ­ Latvia

**Abstract.** This paper proposes the Prefix-Root-Postfix-Encoding (PRPE) algorithm, which performs close-to-morphological segmentation of words as part of text pre-processing in machine translation. PRPE is a cross-language algorithm with only few efforts needed to adapt and tune to a particular language, so it can be potentially useful for morphologically rich languages with no morphological analysers available. The proposed algorithm has yielded improvements for English-Latvian and Latvian-English translation.

**Keywords:** neural machine translation, translation of rare words, morphologically rich languages, quasi-morphological word segmentation

## Introduction

In recent years neural machine translation (NMT) has become indisputable default approach for machine translation. Still the quality of translation is strongly dependents from language pairs – for morphologically rich languages with limited resources of training data it remains a challenge due to data sparseness [tilderich2017].

Among the approaches to overcome inflectedness of a language, data pre-processing should be named, and one of the methods is splitting words into segments (or sub-words) in order to decrease the amount of unique text units thus reducing data sparseness. This is important because of the main paradigm of NMT – sequence to sequence of text units (characters, sub-words, or words) translation.

This article focuses on word segmentation implicitly based on sub-word statistics (Prefix-Root-Postfix-Encoding algorithm, PRPE). The obtained text resembles morphologically segmented text however without any claims for correct morphological splitting. Thus the output of the proposed segmentation method was not compared against some reference segmentation (like in [morphoseg2016]). Instead, it was showed, how the segmentation improves translation quality. Unlike morphological segmentators for particular languages, PRPE is almost language independent, requiring comparatively small work (programming and parameter tuning) to adapt to a new language.

## Related Work

This paper focuses on a particular approach of text pre-processing for NMT to overcome inflectedness of languages and the problem of rare words – segmentation of text into subword units.

### Byte Pair Encoding Based Segmentation Algorithm

Byte pair encoding based segmentation algorithm (BPE), proposed in [bpe2016], utilizes the principle of iteratively finding the most frequent character sequences of the text to become potential segments.

The algorithm consists of two phases: (a) the learning phase, in which the vocabulary of merge operations is obtained, (b) the applying phase, in which a certain text is segmented using the vocabulary.

The learning phase starts with all the words in the text represented as sequences of characters. Then through an iterative process the most frequent pairs of neighbouring symbols (initially, characters) are merged together and these pairs (or ‘merge operations’) written to a special vocabulary. At each iteration, (a) the chosen merge operation is added to the vocabulary, (b) the merge operation is applied to the text. The process is continued until a predefined number of merge operations is reached.

The applying phase transforms a text into a segmented text according to the vocabulary of merge operations.

BPE allows to effectively control the size of vocabulary for the translation as it is equal to the number of unique characters plus the number of merge operations. A bounded vocabulary is essential for NMT approach. Since appeared, the algorithm is often regarded a benchmark algorithm of text segmentation for NMT.

|  |  |
| --- | --- |
| English | you need to know exactly what you want to im@@ mor@@ tal@@ ise during the photo session , and be able to tell the photo@@ grap@@ her about it . |
| Latvian | ir jāsaprot , ko tieši tu vē@@ lies ie@@ mūž@@ ināt foto@@ sesijas laikā un jāpa@@ stāsta par to fotogrāf@@ am . |

Table 1: A segmentation example with BPE. Many of the words (especially in English) are not segmented at all.

### Morphology-Driven Splitting

One of ideas for word segmentation for NMT is trying to separate root hoping that separating roots from affixes (especially suffixes in morphologically rich languages) will preserve more semantic information (words with common roots would also have the same segments).

In [tilderich2017] a language-specific morphological splitting approach is described. To avoid over-segmentation of the text, morphological splitting is performed in a limited manner, i.e., not all affixes are separated (too many segments in a sequence reduces the quality of NMT).

|  |  |
| --- | --- |
| English | you need to know exact@@ ly what you want to im@@ mor@@ tal@@ ise during the photo session , and be able to tell the photo@@ grap@@ her about it . |
| Latvian | ir jā@@ saprot , ko tieš@@ i tu vēl@@ ies ie@@ mūž@@ inā@@ t foto@@ sesij@@ as laik@@ ā un jā@@ pastāst@@ a par to foto@@ grāf@@ am . |

Table 2: A segmentation example with morphology-driven splitting proposed by [tilderich2017] (postprocessed with BPE to support open vocabulary).

It is reported for morphology-driven splitting to have small improvement on translation quality (0.5-0.7 BLEU points, [bleu2002]). The small improvement could be explained by small out-of-vocabulary rate (especially in English).

### Morfessor

About Morfessor here.

|  |  |
| --- | --- |
| English | you need to know exact@@ ly what you want to im@@ mortal@@ ise dur@@ ing the photo session , and be able to tell the photograph@@ er about it . |
| Latvian | ir jā@@ saprot , ko tieši tu vēl@@ ies ie@@ mūž@@ inā@@ t foto@@ sesijas laikā un jāpa@@ stāsta par to foto@@ grāf@@ a@@ m . |

Table 3: A segmentation example with ‘morfessor’ segmentation proposed by [flatcat2014] (postprocessed with BPE to support open vocabulary).

...

## PRPE Segmentation Algorithm

This chapter describes the basic principles of the Prefix-Root-Postfix-Encoding (PRPE) algorithm, proposed by this paper[[1]](#footnote-1).

The basic principle underlying PRPE comes from the BPE algorithm – to learn the most frequent character sequences and then use them to segment words in a text. The main idea added is to take the most frequent left and right substrings of words instead of any character sequences regarding left substrings as potential prefixes and roots, but right substrings as potential postfixes. Then these potential building blocks (prefixes, roots, postfixes) are combined together in a special way to constitute words thus performing segmentation. As a result, a close-to-morphological segmentation is obtained. For better results, the PRPE algorithm should be complemented with a small language specific part.

PRPE has two phases:

1. The **learning phase**, in which ranked lists of main building blocks (potential prefixes, roots and postfixes) are obtained;
2. The **application phase**, in which segmentation is performed using obtained building blocks.

The construction of PRPE algorithm is backed up by the belief that having preprocessed input text through morphological segmentation (especially, with roots separated away) would increase the quality of machine translation.



Fig. 1: Illustration of the building blocks used in PRPE with word “unbelievables”.

### Obtaining Potential Segments

The main goal of the learning phase of PRPE is to obtain lists of potential prefixes, roots and postfixes (suffixes and endings) from a single-language corpus.

The key idea of the algorithm is ***the*** ‘***Root alignment’ principle*** (see illustration in Fig. 2 and example of implementation in Fig. 3):

* Left substrings of words are considered potential roots;
* Aligning potential roots with the middle parts of words allows extracting potential prefixes and postfixes.



Fig. 2: The illustration of the ‘Root alignment’ principle in word “unbelievables”: potential roots aligned with the middle part of the word to collect statistics for prefix “un”.

Obtaining potential segments is carried out in four steps:

1. Collecting frequency statistics of left and right substrings of words. For instance, among the most frequent left substrings in English we can found “the”, “ther”, “re”, “commis”, but among the most popular right substrings – “s”, “es”, “tion”, “ation”.;
2. Extracting potential prefixes from left substrings through aligning other left substrings as potential roots with the middle part of word:
   1. obtain prefix statistics,
   2. select the most frequent prefixes to become potential prefixes in segmentation;
3. Extracting potential postfixes from right substrings through aligning other left substrings as potential roots with the middle part of word:
   1. obtain postfix statistics (in a similar way as for prefixes),
   2. select endings from postfixes according predefined rules to become potential endings in segmentation;
   3. extract and select the most frequent suffixes from postfixes by splitting away collected endings – to become potential suffixes in segmentation;
4. Extracting potential roots from left substrings through aligning them with the middle part of word considering already collected prefixes and postfixes. Here longer roots are also assigned bigger weight coefficients to better compete with smaller roots in the segmentation phase.

All the obtained lists of potential subwords are ranked, and the predefined hyper-parameters determine how many of the respective subwords will become final building blocks. Ranking numbers (1, 2, 3, etc.) will be then used to calculate the best segmentation.

|  |
| --- |
| **module** extract\_potential\_prefixes (*vocab*, *leftstat*):  *vocab* – list of all words found in the text corpus  *leftstat* – statistics of frequencies of left substrings; obtained in step 1 of the learning phase  *prefstat* – prefix statistics to be calculated by this module  **for each** word ***w*** in the vocabulary *vocab*:  **for each** left substring ***p*** **in** ***w***: *# a potential prefix*  **if** ***p*** is a valid prefix according to a hardcoded control:  **for each** substring **r** **in** **w** just after **p**: # a potential root in the middle of w  **if** ***r*** is a valid root according to a hardcoded control:  **and** ***r*** is found **in** *leftstat*:  prefstat[**p**] = prefstat[**p**] + leftstat[**r**]  **return** ordered best prefixes from *prefstat* |

Fig. 3: Implementation of the ‘Root alignment’ principle to extract prefixes. Extracting postfixes and roots is designed in a similar way.

As postfixes are split into suffixes and endings (which is not so important for English, but matters for morphologically rich languages), the output of the learning phase consists of four ranked lists: prefixes, roots, suffixes and endings.

### Segmenting Words Using Obtained Potential Segments

Segmentation phase uses ranked lists (prefixes, roots, suffixes and endings) to segment words. Ranking numbers are used to calculate the best segmentation candidate.

Segmenting a word is carried out in the following way:

1. All possible segmentations for the word are obtained;
2. The highest ranked candidate segmentation wins.

**Collecting all possible segmentations.**

Four ranked lists of potential segments available (P: prefixes, R: roots, S: suffixes and E: endings) for segmentation.

Each candidate segmentation is built in the following form:

**([p] [p] r [s] [e])+**,

where pϵP, rϵR, sϵS, eϵE.

This means that one segmentation is one or more ‘root blocks’ (as root is the only mandatory block in the big block).

Example of segmentation candidates for word “unbelieve” (‘/’ marks boundary of two ‘root blocks’):

1. un – bel – ieve
2. un – bel – i / eve
3. un – believ – e
4. un – believe

**Calculating the best segmentation.**

The best segmentation is the highest ranked segmentation from those with the smallest number of ‘root blocks’, and the rank of the segment is sum of ranks of individual blocks. In the example above the segmentation #2 is of two ‘root blocks’, i.e., out of competition.

**Optimization of the segmentation.**

To reduce the final number of segments several heuristics are used to join back some segments.

**No segmentation candidates.**

If there is no segmentations candidates (i.e., a words cannot be built using available blocks), only the best postfix is split away.

### Adapting the Algorithm to a Particular Language

As the algorithm is not fully language-independent, there should be some adaptation activities carried out for a particular language:

1. Add small language specific source code (validity of word parts is additionally checked by hardcoded routines);
2. Tune hyperparameters (e.g., how many prefixes should be selected as potential prefixes, minimum length of prefixes).

According to the experiments, adapting to a particular language significantly increases the segmentation quality.

## Experimental Work and Results

In our experiments, we use the English-Latvian dataset provided in the WMT 2017 (http://www.statmt.org/wmt17/translation-task.html) shared task in news translation. The data was pre-processed (filtered, normalised, tokenised, etc.) by the authors of [tildewmt2017] in their experiments. The approximate size of each corpus – 250M.

To evaluate the impact of PRPE on machine translation, we segmented text using several methods (both Latvian and English corpora):

1. BPE ([bpe2016]);
2. Tilde’s Morphologically segmented, provided by the authors of [tilderich2017], [tildewmt2017];
3. Segmented using Morfessor ([flatcat2014]);
4. many configurations of PRPE.

Al the non-BPE segmentations were postprocessed using BPE to support open-vocabulary.

Then the segmented corpora were used to build English-Latvian and Latvian-English translation models in two NMT systems:

1. Nematus ([nematus2017])
2. fairseq[[2]](#footnote-2).

## Conclusion

F

S.

Acknowledgements

The research has been supported by the European Regional Development Fund within the research project “Neural Network Modelling for Inflected Natural Languages” No. 1.1.1.1/16/A/215.

References

[tildewmt2017] Pinnis, M., Krišlauks, R., Miks, T., Deksne, D., & Šics, V. (2017). Tilde’s Machine Translation Systems for WMT 2017. In Proceedings of the Second Conference on Machine Translation (WMT 2017), Volume 2: Shared Task Papers (pp. 374–381). Copenhagen, Denmark: Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/W17-4737 (2017)

[tilderich2017] Pinnis M., Krišlauks R., Deksne D., Miks T. Neural Machine Translation for Morphologically Rich Languages with Improved Sub-word Units and Synthetic Data. In: Ekštein K., Matoušek V. (eds) Text, Speech, and Dialogue. TSD 2017. Lecture Notes in Computer Science, vol. 10415. Springer, Cham (2017)

[morphoseg2016] Ruokolainen T., Kohonen O., Sirts K., Grönroos A., Kurimo M., Virpioja S. A Comparative Study of Minimally Supervised Morphological Segmentation. In: Computational Linguistics, vol. 42, issue 1, pp. 91-120. MIT Press (2016)

[flatcat2014] Grönroos A., Virpioja S, Smit P., Kurimo M. Morfessor FlatCat: An HMM-Based Method for Unsupervised and Semi-Supervised Learning of Morphology. In: The 25th International Conference on Computational Linguistics (COLING 2014); Dublin, Ireland, August 23-29, pp. 1177-1185. Dublin City University and Association for Computational Linguistics (2014)

[bpe2016] Sennrich R., Haddow B., Birch A. Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL2016). Berlin, Germany (2016)

[bleu2002] Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. J. BLEU: a method for automatic evaluation of machine translation. ACL-2002: 40th Annual meeting of the Association for Computational Linguistics. pp. 311–318.

[nematus2017] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry and Maria Nadejde (2017): Nematus: a Toolkit for Neural Machine Translation. In Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, pp. 65-68.

1. source code available at: https://github.com/zuters/prpe [↑](#footnote-ref-1)
2. https://github.com/facebookresearch/fairseq [↑](#footnote-ref-2)