Databases and Information Systems X
A. Lupeikiene et al. (Eds.)

doi:10.3233/978-1-61499-941-6-225

Morphology-Inspired Word Segmentation for Neural Machine Translation

Jānis ZUTERS [1], Gus STRAZDS and Viktorija ĻEONOVA
Faculty of Computing, University of Latvia
Raina blvd. 19, 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 requiring only minor tweaking to adapt it for any particular language, a property which makes it potentially useful for morphologically rich languages with no morphological analysers available. As a key part of the proposed algorithm we introduce the 'Root alignment' principle to extract potential sub-words from a corpus, as well as a special technique for constructing words from potential sub-words. In addition, we supplemented the algorithm with specific processing for named-entities based on transliteration. We conducted experiments with two different neural machine translation systems, training them on parallel corpora for English-Latvian and Latvian-English translation. Evaluation of translation quality showed improvements in BLEU scores when the data were pre-processed using the proposed algorithm, compared to a couple of baseline word segmentation algorithms. Although we were able to demonstrate improvements in both translation directions and for both NMT systems, they were relatively minor, and our experiments show that machine translation with inflected languages remains challenging, especially with translation direction towards a highly inflected language.

Keywords. Neural machine translation, word segmentation, named-entity processing

1. Introduction

During the last years, neural machine translation (NMT) has without a doubt become a de-facto standard for machine translation. However, it is not without fault – translation quality currently strongly varies depending on the language pairs in question. This is in no small part due to different language features, as well as the availability of good training data – morphologically rich languages, especially those with relatively little parallel training data available, present significant challenges for NMT training due to data sparseness [1].

Often, various means of pre-processing are employed in order to address data sparsity caused by the inflectedness of a language. One of the most common techniques is splitting words into segments (or sub-words) to reduce the number of unique input tokens. This works as follows: morphologically rich languages contain a high number

[1] Jānis Zuters, Faculty of Computing, University of Latvia, Raina blvd. 19, LV-1586 Riga, Latvia; E-mail: janis.zuters@lu.lv.

of lexicographically unique tokens, since each inflected form encountered for each word counts as a distinct token. Splitting these into segments allows representing them as combinations constructed from a much smaller vocabulary of sub-word tokens, thus reducing the data sparseness. This fits well with the main notion of NMT, where the text units – characters, sub-words or words – are transduced on a sequence-to-sequence basis. In this process, system perceives and processes these units as indivisible tokens. The intended outcome of the segmentation is for the system to learn to generate correct output sequences, that is, correctly inflected word forms for input sequences, independently of whether or not such sequences were present in the training data. This can be achieved by good segmentation into sub-word tokens – i.e. where words forms completely absent from the training data can nevertheless be represented as a sequence of sub-word tokens from the token vocabulary derived from the training data.

The focus of this article is a word segmentation technique based on sub-word statistics (the Prefix-Root-Postfix-Encoding algorithm, PRPE). The output resulting from such segmentation looks like words split morphologically; however, this algorithm makes no claims about making a linguistically meaningful segmentation. [2] describes an approach where well-motivated morphological splitting was performed; as such, it needed to be compared to a reference segmentation. As we are not following this approach, we avoid needing to devote considerable effort to producing a corpus with a "gold-standard" reference segmentation, and, prior to that, a linguistic model of the morphological structure of the language in question. Instead, we assess the performance of our algorithm with experiments testing whether the application of PRPE improves translation quality in comparison with baseline segmentation methods.

In addition to the base functionality of splitting words into sub-word tokens, we have also introduced a new module for the transliteration of named entities not present in the training set. By applying the segmentation algorithm to such named entities, we aim to produce viable transliterations, with the correct inflected form, of the named entities.

PRPE is by its nature almost language-independent, and can be adapted to a new language with relatively little work: changing a set of parameters to new values and adding or modifying several lines of code.

2. Related Work

The focus of this paper is a particular preprocessing approach – a segmentation algorithm for splitting the text into sub-word tokens. This approach allows tackling such problems as the inflectedness of morphologically rich languages and the sparsity of specific grammatical forms in training corpora stemming from it. In this section, we give a short overview for the most representative examples of other commonly used segmentation algorithms.

2.1. Byte Pair Encoding Based Segmentation Algorithm

In [3], the authors propose an adaptation of byte pair encoding [3] for the field of NMT. BPE is based on the principle of replacing the frequently encountered byte pairs with a new, previously unused byte. In NMT, governed by the same notion, the algorithm iteratively finds the most frequent character sequences to be used as

segments. An example of segmentation resulting from BPE application is provided in Table 1.

BPE algorithm work contains two stages: (a) a learning stage and (b) an application stage. During the former, the algorithm processes the training corpus and constructs a vocabulary of merge operations; during the latter, a particular text is segmented using the constructed vocabulary.

In the beginning of the learning stage, all words in the training corpus are split into characters and the vocabulary is initialized with these characters. Then the iterative process starts. Each iteration, the algorithm finds the most frequent pairs of neighboring symbols and (a) adds them to the vocabulary together with a new symbol, denoting the merge operation. Then these merge operations are applied to the text, replacing the corresponding input symbols. The iterations continue until a predefined number of merge operations – an algorithm parameter – is reached.

BPE algorithm effectively controls the size of vocabulary used for translation, as the number of unique tokens in the vocabulary constructed by BPE does not exceed the number of original characters in the training corpus plus the number of merge operations. Due to the nature of NMT, a bounded vocabulary is essential; thus as soon as BPE algorithm was presented in [3], the pre-processing of input text with BPE has become a de-facto industry standard.

Table 1. A segmentation example with BPE.

Language	Segmented Text
English	flu–id was rapidly accum–ulating in his brain , but all Latvian doctors would be able to do to help would be to make a bypass operation to divert excess flu–id, since a direct sur–gical operation on the t–um–our was practically unavailable .
Latvian	strauji krā–jas šķidr–ums galvas sma–dz–en–ēs , bet viss , ko Latvijā ārsti var palīdzēt , ir veikt š–un–tēšanas operāciju lai no–vadītu liek–o šķidr–umu , jo tieš–ai ķir–ur–ģ–iskai darbībai audz–ējs praktiski nav pieejams .

2.2. Morphology-Driven Splitting

Another approach to word segmentation for NMT, particularly applicable to morphologically rich languages, attempts to separate the word root from its affixes, with the notion that words stemming from the same roots would possibly have the same segments and thus that doing so would allow preserving more semantic information.

Table 2 provides a segmentation example for a language-specific morphological splitting, described in [1]. An excessive number of segments in sequences has an adverse effect on the quality of NMT, thus over-segmentation of a text should be avoided. In order to do so, the morphological splitting is performed in a limited manner; in other words, only some of the affixes are separated from the root.

In the experiments with translating between Latvian and English, the application of morphology-driven splitting has resulted in small improvements (0.5 versus 0.7 BLEU points, [4]) in comparison with BPE. A possible cause for this small improvement might be a relatively small out-of-vocabulary rate in the training data used, particularly in English.

Table 2. A segmentation example with morphology-driven splitting proposed by [1].

Language	Segmented Text
English	fluid was rapid–ly accumulat–ing in his brain , but all Latvian doctors would be able to do to help would be to make a bypass operation to divert excess fluid, since a direct surgic–al operation on the tumour was practical–ly unavail–able .
Latvian	strauj–i krāj–as šķidrum–s galv–as smadzen–ēs , bet viss , ko Latvij–ā ārst–i var palīdzē–t , ir veik–t šuntēšanas operācij–u lai novadī–tu liek–o šķidrum–u , jo tieš–ai ķirurģisk–ai darbīb–ai audz–ējs praktisk–i nav pieej–ams .

Morphology-driven splitting is typically carried out using language-specific morphological analysers. Building such analysers for inflective and agglutinative languages is more complicated than for English (see [5]). For example, for many languages morphological analysis must deal with a considerable amount of ambiguity, and therefore various disambiguation models are used ([6], [7]). As morphological analysers are typically language specific, it takes a lot of effort to build such a tool for any given language (e.g. creating morphologically annotated corpora, developing language specific routines).

2.3. Named-entity Processing for Machine Translation

In a machine translation task, translation of named entities is a particularly challenging issue. Infrequently mentioned named entities contribute to data sparsity even in languages without extensive inflective grammatical processes, but, to be translated correctly, they often require a different approach than is used for translating rare "normal" words. Examination of texts translated using neural systems shows that translation of named entities yields worse results compared to regular out-of-vocabulary words. By default, the neural system tries to translate named entities the same way it translates all other words: by segmenting their component words into sub-word tokens, which are processed by the NMT sequence-to-sequence transduction process to generate corresponding output sub-word tokens, which are then stitched back together to hopefully produce words from the target language. But this process doesn't necessarily work as well for named entities, since they can be formed using very different linguistic processes than those governing the base vocabularies of the source and target languages. Table 3 illustrates the issue of translation of named entities.

Table 3. Illustration of named-entity translation problem. Column #1 contain original English sentences, column #2 contain English sentences translated from the respective Latvian sentences.

Reference English sentence	Translated English sentence (from Latvian)
Interior Minister **Bernard Cazeneuve** is to inspect security arrangements on Saturday.	French Minister for the Interior, **Bernard Kazulvs**, will examine the security measures on Saturday.
Ryan Lochte is going to beat **Michael Phelps** in this event!	**Lohte** is going to be the **Make Felevs** in this competition.
British diver **Tom Daley**, who won bronze in the synchro platform event, also commented on the state of the pool in a Twitter post.	the British woman , **Tils Dilis**, who fought the basin of the bronze platform, was also a comment on the state of the basin .

Paper [1] proposes to modify their sub-word splitting algorithm by keeping together unknown parts of a word (i.e., without splitting them) to be able then to process them differently.

In many languages, one of the approaches used for translation of some kinds of named-entities is phonetic transliteration. In [8], a successful use of neural networks for transliteration is described.

3. PRPE Segmentation Algorithm

3.1. General Description

This section describes the basic principles of the proposed Prefix-Root-Postfix-Encoding (PRPE) algorithm[2] (see a segmentation example in Table 4) as well as a prototype of named-entity-specific processing[3]. The main motivation for the algorithm is the belief that splitting away roots from words would produce more meaningful parallel sequences for machine translation (as with morphology-driven splitting, see Section 2.2), thus increasing the quality of machine translation. But the goal of PRPE is to obtain such a segmentation based primarily on the statistics of the training data, using a bare minimum of language specific knowledge (contrast this with a language-specific morphological analyser, which would be hand-crafted using a large number of language-specific rules based on a linguistically motivated analysis of the morphological processes at work in a given language).

Table 4. A segmentation example with PRPE. Linguistically, morphological splitting is similar in Latvian and English. The two main differences for Latvian: 1) substantially more inflectedness = many more systematically varying word endings; and 2) word roots almost always end with a consonant.

Language	Segmented Text
English	fluid was rapidly accumulat–ing in his brain , but all Latvian doctors would be able to do to help would be to make a bypass operation to divert excess fluid , since a direct surgic–al operation on the tumour was practically un–avail–able .
Latvian	strauji krāj–as šķidr–ums galv–as smadzen–ēs , bet viss , ko Latvijā ārst–i var palīdzēt , ir veikt šuntēšan–as operāciju lai novad–ītu liek–o šķidr–umu , jo tieš–ai ķirurģisk–ai darbībai audz–ējs praktiski nav pieejams .

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 number of language specific heuristics. Instead of complicated probability

[2] Source code available at: https://github.com/zuters/prpe
[3] Source code available at: https://github.com/zuters/prpene

computations, in PRPE we use substring frequencies and lists of substrings specifically ranked according to frequencies.

PRPE has two phases:

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

From the algorithmic perspective, PRPE contributes two main ideas:

- The 'Root alignment' principle to extract potential roots and other sub-words in the learning phase;
- A special technique to construct words from obtained potential sub-words thus accomplishing word segmentation.

3.2. 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.

<table>
<tr><th>un</th><th>believ</th><th>abl</th><th>es</th></tr>
<tr><td rowspan="2">prefix</td><td rowspan="2">root</td><td>suffix</td><td>ending</td></tr>
<tr><td colspan="2">postfix</td></tr>
</table>

Figure 1. Illustration of the building blocks used in PRPE for the word "unbelievables".

The key idea of the algorithm is the 'Root alignment' principle (see illustrations in Figure 1 and Figure 2, and example of implementation in Figures 3, 4 and 5):

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

<table>
<tr><th>u</th><th>n</th><th>b</th><th>e</th><th>l</th><th>i</th><th>e</th><th>v</th><th>a</th><th>b</th><th>l</th><th>e</th><th>s</th></tr>
<tr><td colspan="2">prefix</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="4">potential root</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="6">potential root</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="7">potential root</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="8">potential root</td><td></td><td></td><td></td><td></td></tr>
</table>

Figure 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 (see Algorithm in Figure 3):
   a) obtain prefix statistics,
   b) 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 (see Algorithm in Figure 4):
   a) obtain postfix statistics (in a similar way as for prefixes),
   b) select endings from postfixes according predefined rules to become potential endings in segmentation;
   c) 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 (see Algorithm in Figure 5).

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

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.

```
module extract_potential_prefixes (vocab, leftstat):
  vocab – list of all words found in the text corpus
  leftstat – statistics of frequencies of left substrings
             as candidate roots
  prefstat – prefix statistics to be calculated
  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:
          # a potential root in the middle of w:
        for each substring r in w following p:
          if r is a valid root according to a hardcoded control
          and r is found in leftstat:
            prefstat[p] = prefstat[p] + leftstat[r]
  return prefstat
```

Figure 3. Prefix extraction module to algorithmically illustrate the 'Root alignment' principle: trying to locate potential roots (frequent left substrings) in the middle of a word to extract potential prefixes.

```
module extract_potential_postfixes (vocab, leftstat):
  vocab – list of all words found in the text corpus
  leftstat – statistics of frequencies of left substrings
             as candidate roots
  poststat – postfix statistics to be calculated
  suffstat – suffix statistics to be calculated
  endstat – ending statistics to be calculated
  for each word w in the vocabulary vocab:
    for each right substring p in w: # a potential postfix
      if p is a valid postfix according to a hardcoded control:
          # a potential root in the middle of w:
        for each substring r in w preceding p:
          if r is a valid root according to a hardcoded control
          and r is found in leftstat:
            poststat[p] = poststat[p] + leftstat[r]
            if p is a valid ending according to a hardcoded
control:
              endstat[p] = endstat[p] + leftstat[r]
    # extract suffixes as left parts of postfixes:
  for each postfix p in poststat:
    for each left substring s in p: # a potential suffix
      with right substring e in p where s + e == p:
        if e found in endstat:
          suffstat[s] = suffstat[s] + poststat[r]
  return suffstat, endstat, poststat
```

Figure 4. Postfix extraction module to extract potential postfixes which are split into suffixes and endings. It also exploits the 'Root alignment' principle.

```
module extract_roots (vocab, leftstat, suffstat, endstat):
  vocab – list of all words found in the text corpus
  leftstat – statistics of frequencies of left substrings
             as candidate roots
  prefstat – prefix statistics
  poststat – postfix statistics
  rootstat – root statistics to be calculated
  for each word w in the vocabulary vocab:
    for each left substring p in w where p found in prefstat:
      for each right substring pp in w: where pp found in
poststat
        with substring r in p where p + r + pp == w:
          if r is a valid root according to a hardcoded control
          and r is found in leftstat:
            rootstat[r] = rootstat [r] + leftstat[r]
  return rootstat
```

Figure 5. Root extraction module.

3.3. Segmenting Words Using Obtained Potential Segments

The 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:

$$([p]\ [p]\ r\ [s]\ [e])\ +, \tag{1}$$

where $p \epsilon$P, 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). We search for two prefixes because the two prefix case is quite common in Latvian (an example from English would be "non-re-active").

Example of segmentation candidates for word "unbelieve" ('/' marks boundary of two candidate 'root blocks'):

- un–bel–ieve
- un–bel–i / eve
- un–believ–e
- 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 the sum of ranks of individual blocks. In the example above the segmentation #2 is of two 'root blocks', i. e., out of competition.

3.4. Named-Entity Processing

To examine specific processing for named entities, an auxiliary unit has been added to the main segmentation algorithm for PRPE. The named-entity unit stands apart from the overall named-entity recognition problem. Instead, only a subset of named entities (initially those that are easiest to extract) undergoes processing, in order to explore the impact of named-entity-specific segmentation on the NMT process.

PRPE is complemented the following way (see Section 3.1.):

- The learning phase stays unchanged;
- In the application phase, segmentation is carried out by putting named entities on separate input lines split into characters.

If a named entity is recognized in the sentence,

- it is split into the main part and the ending (by distinguishing between the ending and the rest of the word we aim at transliteration while also producing a correct grammatical form for the named entity – which matters for morphologically rich languages),

- the main part is split into characters and put on a separate line above the sentence,
- the main part of the named entity in the sentence is replaced by a placeholder,
- in the translated text, the translated main part of a named entity is substituted back in to replace the placeholder.

For parallel training corpora, the described segmentation of named entities in sentences is carried out only if pairs of aligned named entities are recognized.

Tables 5 and 6 illustrate the applied approach for segmentation sentences containing named entities.

Table 5. Segmentation example of a sentence containing a named entity in an English sentence (without ending in the named entity).

Segmentation information	**Segmented Text**
Original sentence	a city tour at **Zadar** reveals remains of a Roman forum and a sea organ which plays music like the moan of a caged sea monster , through pipes set in the stone fabric of the promenade and open to the water .
Segmented named entity put before the sentence	Z–a–d–a–r–.
Segmented sentence	a city tour at <PLACEHOLDER>–. reveals remains of a Roman forum and a sea organ which plays music like the moan of a cag–ed sea monster , through pipe–s set in the stone fabric of the promen–ade and open to the water .

Table 6. Segmentation example of a sentence containing a named entity in a Latvian sentence (with a word ending in the named entity).

Segmentation information	**Segmented Text**
Original sentence	pilsētas apskatē **Zadarā** atklājas romiešu forums un jūras ērģeles , kas spēlē mūziku , kas skan kā ieslodzīta jūras briesmoņa vaids , caur akmenī iestrādātām caurulēm promenādē un atvērtām ūdenī .
Segmented named entity put before the sentence	Z–a–d–a–r–.
Segmented sentence	pilsētas apskat–ē <PLACEHOLDER>–ā–. atkl–ājas rom–iešu forums un jūras ērģeles , kas spēl–ē mūzik–u , kas skan kā ieslodz–īta jūras briesmoņ–a vaid–s , caur akmen–ī iestrād–ātām caurulēm promen–ād–ē un atvērt–ām ūden–ī .

3.5. Additional Heuristics

Several addition heuristics were used to tune the algorithm for better results.

- *The most frequently encountered words are unsegmented.* To reduce the final number of segments, a predefined number of the most frequent words stay unsegmented (see 'leave-out rate' in the results).
- *Optimization of the segmentation.* To reduce the final number of segments, several heuristics are used to join back some segments, e.g.:
  - prefixes not split away,

  - suffixes not split away between roots.
- *No segmentation candidates.* If there are no segmentations candidates (i.e., a word cannot be built using available blocks), only the best postfix is split away.
- *Uppercase marking.* A word starting with uppercase and with all remaining symbols in lowercase is converted to lowercase, and a special uppercase marker is inserted before it.

3.6. Adapting the Algorithm to a Particular Language

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

1. Add a small amount of language-specific source code (candidate word parts are additionally screened by a small number of hand-coded routines/rules);
2. Tune hyperparameters (e.g., how many prefixes should be selected as potential prefixes, minimum length of prefixes).

According to the experiments, adapting the base algorithm in this way for a particular language noticeably increases the segmentation quality.

4. Experiments and Results

The main idea for the experiments was to show that pre-processing corpora with PRPE yields better machine translation results relative to baseline segmentation schemes.

For our experiments, we used the English-Latvian dataset provided in the WMT 2017[4] shared task in news translation. The approximate size of each of the parallel corpora – 1.6M sentences. We use as a starting point the data as pre-processed (filtered, normalised, tokenised) by the authors of [9] for their experiments.

We obtained sub-word-segmented versions of both the English and Latvian texts using various configuration of PRPE, including:

1. without specialized named-entity processing;
2. with specialized named-entity processing,

as well as two baseline segmentation algorithms:

1. BPE ([3])[5];
2. Tilde's Morphologically segmented version of the same dataset, also provided to us by the authors of [1], [9].

All the non-BPE segmentations were also post-processed using BPE to better support open-vocabulary translation (by ensuring full coverage of the word vocabulary in the training data, since that is not an explicit goal/guarantee of the alternative segmentation schemes). In all cases, both languages were segmented similarly, using the same algorithm with one set of configuration parameters per experiment.

To evaluate the impact of PRPE on machine translation, we then used these various sub-word-segmented parallel corpora to train English-to-Latvian (en-lv) and Latvian-to-English (lv-en) translation models using two architecturally quite different

[4] http://www.statmt.org/wmt17/translation-task.html
[5] https://github.com/rsennrich/subword-nmt

NMT systems: Nematus ([10])[6] and ConvS2S ("Convolutional Sequence to Sequence", [11])[7].

Training even a relatively small NMT model on one or two GPUs takes a minimum of several days, so resource and time constraints precluded our doing much in the way of search over the space of potential configuration and training hyperparameters for the NMT systems we used. But since our goal was not to find optimal configurations and maximize translation BLEU scores, but instead to test for incremental benefits from using our proposed sub-word segmentation scheme, we chose an initial set of NMT configuration and training parameters (yielding reasonably good baseline results), and then used them unchanged for all subsequent experiments. We did, however, try various settings of the internal parameters of the PRPE algorithm, and found that different settings yielded best results for Nematus vs. ConvS2S. This leads to the observation that PRPE configuration should be tuned in concert with other hyperparameters when training an NMT system. (This is completely analogous to selecting the number of merge operations for BPE.) In particular, the "leave-out rate" seems to be the most important tunable parameter for PRPE.

Previous results[8] have shown that the translation direction English-to-Latvian generally yields worse scores than Latvian-to-English, and in all cases our results were consistent with this finding. This could be explained by the supposition that translation towards a morphologically richer language is a more challenging task. That's why we hoped to obtain improvements in this particular direction. Unfortunately, with Nematus, the best configuration of PRPE gave a minor (but not statistically significant[9]) improvement in BLEU score for lv-en (Latvian-to-English) translation, but in the en-lv direction produced almost identical scores to the morphologically segmented baseline (see Table 7). With ConvS2S we observed statistically significant improvements in both directions (see Table 8).

Note that the baseline scores that we obtained using ConvS2S were 3-4 BLEU points higher than the corresponding scores obtained using Nematus with the same datasets. We conjecture that this might be to a large extent because we were using a relatively basic (shallow) configuration of Nematus, with less modeling capacity than the large and deep default configuration we chose for ConvS2S. To test this conjecture – and the possibility that deeper networks might be better able to make use of more sophisticated sub-word segmentation schemes (as suggested by the bigger boost from PRPE that we saw with ConvS2S vs. Nematus) – we ran a few additional experiments using a configuration for Nematus based on training scripts provided by Edinburgh University[10] [12], which make use of Nematus features that allow for deeper network configurations in its encoder and decoder [13]. Initial results (see Table 9) seem to confirm these conjectures, but, due to time constraints, a more systematic exploration will have to await future work.

[6] https://github.com/EdinburghNLP/nematus

[7] https://github.com/facebookresearch/fairseq-py

[8] http://www.statmt.org/wmt17/results.html

[9] Statistical significance was estimated via bootstrap resampling using the script analysis/bootstrap-hypothesis-difference-significance.pl from the Moses MT system: https://github.com/moses-smt/mosesdecoder

[10] http://data.statmt.org/wmt17_systems/training

Table 7. Translation results with Nematus system using various segmentation techniques.

	BPE (BLEU)	Tilde's morph (BLEU)	PRPE (leave-out rate = 5000)	
			BLEU	p-val vs BPE
en-lv	17.05	17.15	**17.16**	0.23
lv-en	18.66	18.67	**18.90**	0.13

Table 8. Translation results with ConvS2S system using various segmentation techniques.

	BPE (BLEU)	Tilde's morph (BLEU)	PRPE (leave-out rate = 5000)	
			BLEU	p-val vs BPE
en-lv	20.30	21.26	**21.33**	0.00
lv-en	21.93	22.05	**22.61**	0.01

Table 9. Translation results using deeper Nematus models.

	BPE (BLEU)	PRPE (leave-out rate = 5000)	
		BLEU	p-val vs BPE
en-lv	19.13	**19.55**	0.06
lv-en	20.90	**21.46**	0.01

Experiments on text segmentation with named-entity-specific processing were started later, and they strongly depend on named-entity recognition capability (which lies beyond the scope of this research). At this point in time we have obtained only initial qualitative results for our proposed approach (see Table 10), which show promise, but with some technical issues that still need to be addressed. Due to limited time and compute resources, we have so far carried out such experiments only with the "shallow" base configuration of Nematus, and not yet with ConvS2S or the more powerful (but slower) "deep" Nematus configuration.

Table 10. Selected examples of machine translations from Latvian to English to illustrate the difference in output produced when using named-entity-specific segmentation (named entities highlighted).

Expected sentence	Sentence obtained without specific named-entity processing in segmentation	Sentence obtained with specific named-entity processing in segmentation
he told her : " **Joshua** isn 't breathing properly , come home right away."	he told her she said : " They 're not alarmless , even come home."	he told her : " **Josha** don 't breathe like coming , right immediately at home."
with regard to the fact that members are not satisfied with my work , up to now only one example has been mentioned , in regards to the aforementioned coach **Jakubovskis** , who has expressed his dissatisfaction.	with regard to the fact that the members are dissatisfied with my work , until now , there has been only one case , with **Julika Juliovski** , where there is something unsatisfactory .	regarding the fact that the members are dissatisfied with my work , there has been only one case so far , with the **Jakubovski** that has already been mentioned , when there is some unsatisfaction.
koushik **Chatterjee** , executive director of Tata Steel Europe , said the Indian conglomerate wants to make its steel business " more sustainable."	the Executive Director of Trainei , Tata Steel Europe 's Executive Director , said that the Indian conglomerate wants to make its steel business not " more sustainable " .	the **Chatera** , Tati Steel Europe 's executive director , said that the Indian conglomerates wanted to make the steel business more sustainable.

5. Conclusion

In this paper, we propose an algorithm for close-to-morphological word segmentation for machine translation without requiring the availability of language specific morphologically labelled data. Experimental results demonstrated that PRPE pre-processing of training data for NMT can yield small improvements in translation output, relative to pre-processing with baseline sub-word segmentation algorithms. But the results also show that machine translation with inflected languages remains a big challenge, especially with translation direction towards a highly inflected language.

The PRPE algorithm exploits the 'Root alignment' principle to extract potential sub-words, as well as a special technique to construct words from potential sub-words.

In addition, the experiments showed that fully splitting all affixes is counter-productive, in that it produces too long sequences of sub-words, and the translation quality grows worse. The best results were achieved with only compound splitting plus splitting postfixes from the end of a word, as well as leaving up to 5000 of the most frequently encountered words unsegmented.

Obtained improvements in translation quality from PRPE pre-processing were not particularly large, in some cases falling below a commonly used threshold for statistical significance, which might be a signal that the approach of autonomous (without using syntactic and semantic context) pre-processing to do sub-word segmentation might be near its limits for potential improvements.

Experiments with named-entity-specific segmentation show promise, however more advanced named-entity recognition seems to be a key necessity in order to benefit from this approach.

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, and the Faculty of Computing, University of Latvia.

References

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

[2] T. Ruokolainen, O. Kohonen, K. Sirts, A. Grönroos, M. Kurimo, S. Virpioja, A Comparative Study of Minimally Supervised Morphological Segmentation. *Computational Linguistics* **42**, issue 1 (2016), 91-120.

[3] R. Sennrich, B. Haddow, A. Birch, Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL2016)* **1**, (2016), Berlin, Germany, 1715-1725.

[4] K. Papineni, S. Roukos, T. Ward, T., Zhu, W.J. BLEU: a method for automatic evaluation of machine translation. *ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, (2002), Philadelphia, Pennsylvania, 311-318.

[5] J. Hajič, Morphological Tagging: Data vs. Dictionaries. *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (NAACL 2000)* (2000), Seattle, Washington, 94-101.

[6] P. Paikens, L. Rituma, L. Pretkalnina, Morphological analysis with limited resources: Latvian example. *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA)* (2013), Linköping, Sweden, 267-277.

[7] M. Pinnis, K. Goba, *Maximum Entropy Model for Disambiguation of Rich Morphological Tags*. International Workshop on Systems and Frameworks for Computational Morphology (2001), Springer, Berlin, Heidelberg, 14-22.

[8] Z. Li, E.S. Chng, H. Li, Named entity transliteration with sequence-to-sequence neural network, *Proceedings of the 2017 International Conference on Asian Language Processing (IALP)*, 2017.

[9] M. Pinnis, R. Krišlauks, T. Miks, D. Deksne, V. Šics, Tilde's Machine Translation Systems for WMT 2017. *Proceedings of the Second Conference on Machine Translation (WMT 2017)* (2017), Copenhagen, Denmark, 374-381.

[10] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A.V.M. Barone, J. Mokry, M. Nadejde, Nematus: a Toolkit for Neural Machine Translation. *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (2017), Valencia, Spain, 65-68.

[11] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y. Dauphin, Convolutional Sequence to Sequence Learning. *Proceedings of the 34th International Conference on Machine Learning* (2017), Sydney, Australia, 1243-1252.

[12] R. Sennrich, A. Birch, A. Currey, U. Germann, B. Haddow, K. Heafield, A.V.M. Barone, P. Williams, The University of Edinburgh's Neural MT Systems for WMT17. *Proceedings of the Second Conference on Machine Translation,* **2**: Shared Task Papers (2017). Copenhagen, Denmark, 389-399.

[13] A.V.M. Barone, J. Helcl, R. Sennrich, B. Haddow, A. Birch, Deep Architectures for Neural Machine Translation. *Proceedings of the Second Conference on Machine Translation* (2017), Copenhagen, Denmark, 99-107.