Predictive text for agglutinative and polysynthetic languages

Anonymous ACL submission

Abstract

This paper presents a set of experiments in the area of morphological modelling and prediction. We examine the tasks of segmentation and predictive text entry for two underresourced and indigenous languages, K’iche’ and Chukchi. We use different segmentation methods to make datasets for language modelling and then train models of different types: single-way segmented, which are trained using data from one segmentor; two-way segmented, which are trained using concatenated data from two segmentors; and finetuned, which are trained on two datasets from different segmentors. We measure word and character level perplexities of the language models and find that single-way segmented models trained using morphologically segmented data and finetuned models work the best. Finally, we test the language models on the task of predictive text entry using gold standard data and measure the average number of clicks per character and keystroke savings rate. We find that the models trained using morphologically segmented data work better, although with substantial room for improvement. At last, we propose the usage of morphological segmentation in order to improve the end-user experience while using predictive text and we plan on testing this assumption by training other models and experimenting on more languages.

1 Introduction

Nowadays text prediction is widely used in different cases such as autocomplete, smart keyboards, etc. The underlying models are limited by resources, so they save only the top-N highest frequency words, which may work well with analytic languages, but when it comes to the synthetic languages the out-of-vocabulary (OOV) problem becomes more and more noticeable. In order to deal with it, words are usually segmented in constituent parts, so that more of them can be saved in the model vocabulary.

The segmentation task is not new, there are many algorithms with BPE (Gage, 1994) being the most known and used for segmentation. Such methods do not lean on linguistics but only on statistics. In this paper, we tested whether morphological segmentation can improve language modelling and whether it can compete against statistical segmentation methods in predictive text entry task.

We have a particular interest in developing text prediction that is both effective and ergonomic. By ergonomic we mean that made predictions should be linguistically sound and intelligible for the end user. For example, imagine an English word antidiesestablishmentarianism. An ergonomic segmentation would split the word into its constituent morphs [anti, dis, establish, ment, arian, ism], or an alternative [anti, dis, establishment, arianism]. An unergonomic segmentation might be [antid, isestab, lishmentarianism] or [an, tidises, tablishm, entarianism]. One of the issues with many current methods is that while they can produce segments that are meaningful units, in many cases the segments are not linguistically meaningful. We argue that for the task of predictive text entry producing non-linguistic units creates more cognitive load and so would result in slower text entry than predicting the same amount (or a greater number of) linguistic units.

The remainder of the paper is laid out as follows: in Section 2 we overview the languages we experiment on, in Section 3 we discuss the works that were an inspiration for this paper, in Section 4 we describe the experiments we are doing, in Section 5 we review used segmentation methods, choose the best morphological segmentor and do the segmentation, in Section 6 we provide results of language modelling, in Section 7 we speak about language modelling evaluation task, in Section 8 we discuss our thoughts on the results, in Section 9 we announce the planned future experiments. Examples in this paper will be mostly given in K’iche’ and En-
We performed the experiments using two languages: K’iche’ (ISO-639: quc), a Mayan language of Guatemala that is of the agglutinating type, and Chukchi (ISO-639: ckt), a Chukotko-Kamchatkan language of Siberia of the polysynthetic type. Both of these types are characterised by words consisting of a large number of individual morphs, surface representations of morphemes.

The following examples in K’iche’ (1) and Chukchi (2) demonstrate this tendency.1

1. X-in-e’-ki-k’am-a’
   CR-B1SG-MOV-A3PL-RECEIVE-DEP
   ‘They went to take me’

   Both languages exhibit polypersonal agreement (both the subject and object arguments of transitive verbs are encoded on the verb), and Chukchi, in addition, exhibits noun incorporation. As it can be seen in example 2, the object mane/wan’la/’money’ is incorporated, rendering intransitive the transitive root ван’ла/’wan’la/’ask’.

   nemæqej na-mane-wan’la-sqew-э-ены-т
   ‘They also came to ask for money’

Languages of these types are widespread across the Americas but infrequent in Europe and, as a result, were less researched in terms of predictive text input.

### 2.1 Data

As K’iche’ and Chukchi are low-resource languages, the availability of large corpora is limited. We used data annotated for morphological segments and unannotated text as well. For Chukchi, the annotated data came from the ChukLang corpus, a disbalanced ratio like 80 percent/20 percent can lead to unreliable results. The automatically segmented corpus was then used for language modelling, while the test split of annotated data was used for predictive text.

### 3 Related work

Being one of the latest works (Schwartz et al., 2020) on language modelling of indigenous languages, this paper proposed the usage of morphological segmentation in order to improve metrics of language modelling. They compared different segmentation methods, such as single words, dividing into characters, BPE, Morfessor, Finite-state transducers (FST). Even though FST is a good segmentation method used for lots of languages (Mittal, 2010; Hlaing and Mikami, 2014) and there are even ones for K’iche’ (Richardson and Tyers, 2021) and Chukchi, we chose the usage of morphological segmentation to improve metrics. For our experiment, we used the automatically segmented corpus and the test split of annotated data for predictive text.

### Table 1: Dataset sizes for the two languages measured in sentences and words

<table>
<thead>
<tr>
<th>Language</th>
<th>Unannotated</th>
<th>Annotated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sents</td>
<td>Words</td>
</tr>
<tr>
<td>K’iche’</td>
<td>24,254</td>
<td>275,265</td>
</tr>
<tr>
<td>Chukchi</td>
<td>33,322</td>
<td>151,585</td>
</tr>
</tbody>
</table>


2https://chuklang.ru/
and Chukchi (Andriyanets and Tyers, 2018), we decided that we will not use them because the coverage for Chukchi is too low and it is hard to do disambiguation with FST because it requires a huge tagged corpus. Unfortunately, the authors could not do the end-task evaluation of the trained models but suggested doing predictive text. While also having no access to native speakers we decided to emulate the user input in order to evaluate the models. It still is not as good as end-user testing though it is better than nothing.

Another work (Boudreau et al., 2020) that gave us ideas on how to approach the language modelling task was devoted to Mi’kmaq language modelling evaluation. Mi’kmaq (ISO-639: mic), an Eastern Algonquian low-resource polysynthetic language, is spoken primarily in Eastern Canada and has around 8700 speakers. Not only did the authors work with indigenous language, but they also did the keystroke savings evaluation, which is pretty similar to the idea of predictive text evaluation described in the previous work.

There are other works (Suhartono et al., 2014; Yu et al., 2017) that describe keystroke savings evaluation. What is more important, the authors worked with agglutinative languages, Bahasa (ISO-639: ind), the official language of Indonesia, and Korean (ISO-639: kor), official and national language of both North Korea and South Korea (originally Korea). Though we do not want to use the same language modelling technics as were described in the papers, we still find it inspiring there are works dedicated to this task.

As we mentioned before, we assume that the usage of morphs while doing text prediction will make it both effective and ergonomic. However, there was a research (Lane and Bird, 2020) on Kunwinjku, a polysynthetic language of northern Australia, and Turkish, which states that morph-based auto-complete for polysynthetic languages can be troublesome due to long words and sparse vocabularies of such languages. Moreover, dialectal variations and dealing with input errors using edit distance makes the next-morpheme predictioning even harder, so, as it is shown in the paper, Turkish may be a more attractive language for morph-based predictioning than Kunwinjku.

4 Tasks

As mentioned previously, our experiments are split into four distinct tasks, from the more fundamental to the more application-specific. In the following sections we describe the methodology for these tasks and the results obtained.

Morphological segmentation We train morphological segmentation models based on the corpora outlined in Table 1 and evaluate them. Here to be a correct segmentation models must match the reference sentence in the test set. The evaluation measure is \( F_1 \) score. \( F_1 \) score is defined using precision and recall:

\[
\text{precision} = \frac{\text{correct boundaries found}}{\text{total boundaries found}} \quad (1)
\]

\[
\text{recall} = \frac{\text{number of found}}{\text{total correct boundaries}} \quad (2)
\]

\[
F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (3)
\]

Language modelling We take the best morphological segmentation model and the statistical segmentation ones in order to do language modelling. We do 10-fold cross-validation in order to train models for end-task evaluation. We also do experiments using one fold investigating how the data volume influences the model training results. The evaluation metric is word and character level perplexity. Although the model we chose allows both character and word level training, in this paper we do word level training with subwords serving as words.

Predictive text entry We take the trained models from the former task and compare their performance in the predictive text task. The task is to predict the next linguistic unit of output for a given input looking at the top-3 predictions. The evaluation measure is average number of clicks per character and keystroke savings rate. The fewer clicks per character the less the end-user has to type. It’s important to mention that the first segment of each word is always typed character by character; this is caused by the model not having token \(<\text{bos}>\) (beginning of the sentence) in its design and the fact that we are doing word level training. As mentioned above, we use the cross-validation models for this task.

Significance testing As all of the tasks are done using cross-validation, we have sets of results for each model. These results can be tested in order to say if some models are significantly better than
the others. To implement this, first, we do the one-way ANOVA\(^3\) with the null hypothesis being “all the means are the same”. In case the null hypothesis is rejected, we do pairwise Least Significant Difference test (LSD-test)\(^4\) to group the models so that we can find the best performing ones which are not significantly different from each other. The LSD values for all the tasks are given in the appendix.

5 Segmentation

The idea to compare statistical and morphological segmentation was already tested (Pan et al., 2020); the results show that the usage of morphological segmentation significantly improves the BLEU and ChrF3 metrics in neural machine translation (NMT).

In this paper we tried six segmentation models. Four of these are unsupervised: Byte-pair encoding (BPE; Gage, 1994), which was popularised by (Sennrich et al., 2016), Unigram (Kudo, 2018), WordPiece (Schuster and Nakajima, 2012) and Morfessor (Virpioja et al., 2013). The other two are supervised: NeuralMorphemeSegmentation (NMS; Sorokin and Kravtsova, 2018) and NCRF++ (Yang and Zhang, 2018).

Morfessor, NMS and NCRF++ are morphological segmentation models and they have to be trained and evaluated against a test subset. The remaining three do not require evaluation although they can be evaluated on a hold-out dataset.

As an output format, we decided to use one of those in the mentioned work (Pan et al., 2020): we modified the stem with singular suffix strategy, so that all of the subwords are treated the same way: single-morpheme words remain unchanged, in composite words every morpheme except the last one ends with #, the last morpheme ends with $.

5.1 Systems

As we decided to compare statistical and morphological segmentation, we also wanted to compare the models within each type. For this reason we chose BPE, Unigram and Wordpiece as statistical segmentation models. While choosing morphological segmentation models, we were looking for the ones that were tested not only on English and also

<table>
<thead>
<tr>
<th></th>
<th>Chukchi</th>
<th>K'iche'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morfessor</td>
<td>0.610</td>
<td>0.618</td>
</tr>
<tr>
<td>NMS</td>
<td>0.840</td>
<td>0.907</td>
</tr>
<tr>
<td>NCRF++</td>
<td>0.821</td>
<td>0.874</td>
</tr>
</tbody>
</table>

Table 2: Best F\(_1\)-score for morphological segmentation. The neural morphological segmentation (NMS) model outperforms both Morfessor and NCRF++ for both languages.

had F\(_1\)-score as a computed metric. Thus, alongside Morfessor, the best-known morphological segmentor, we also chose NMS and NCRF++.

5.2 Results

We decided to choose the best segmentation model out of Morfessor, NMS and NCRF+++, because the paper is not dedicated entirely to defining the best morphological segmentation model being rather examining if it is better than statistical segmentation in specific tasks. The results of morphological segmentation are presented in Table 2 and the models hyperparameters are included in the appendix.

While being designed for Russian, an inflective language, NMS outperformed Morfessor and NCRF++. Though, since the small size of the available data the metrics are not stable, fluctuating between \(n - 0.3\) and \(n + 0.3\), where \(n\) is the F\(_1\)-score on evaluation data. Thus the second-best model, NCRF++, was chosen to segment the whole corpus and to be used while evaluating the language model.

Hence, 3 segmentation methods were used to make datasets for the following experiments – NCRF++, Unigram and Wordpiece. In order to do Unigram segmentation we used a package made by Google (Kudo and Richardson, 2018), for Wordpiece we used BertWordPieceTokenizer model from tokenizers package (Moi, 2021). Table 3 shows the same sentence being segmented with different models.

6 Language modelling

In order to do the text prediction we decided to choose the model that achieved state-of-the-art word level perplexities on Penn treebank and WikiText-2 (Merity et al., 2017). This model was applied (Schwartz et al., 2020) to several indigenous languages, including Chukchi, and showed good performance. This model trains fast, allows to be trained both on character level and word level, and

---


also is good dealing with overfitting, which is essen-
tial while working with low-resource languages.

Although BERT (Devlin et al., 2019) has
been successfully used for low-resource languages
(Ngoc Le and Sadat, 2020; Wang et al., 2020), mod-
els based on BERT models usually have hundreds of
millions of parameters and as such are not efficient
enough in terms of space for existing mobile phones.
This is not suitable for us as our main goal is to use
the model for a phone keyboard in order to do pre-
dictive text.

The data for language modelling was at first split
into modelling (80 percent) and test (20 percent)
subsets. Then for the 10-fold cross-validation the
modelling subset was split into train (75 percent)
and validation (25 percent) subsets. The folds were
made using ShuffleSplit6 with the same seed as the
one used while language modelling. The dictionar-
es for the embeddings consist of all the subwords
of train dataset plus the <unk> token; the validation
subset is used to calculate perplexity in the end of
each epoch. The models were trained until 5 epochs
without perplexity improvement on a validation sub-
set.

In order to investigate how the volume of data in-
fluences the model perplexity we decided to do lan-
guage modelling increasing the data volume from
10 percent to 100 percent with a 10 percent step.
The data is shuffled using the same seed as in other
tasks and then we take the first \( n \) percent of lines.
Thus we can be sure that two-way segmented mod-
els won’t get the data only from one segmentor. We
did these experiments using only one fold as we did
not intend to show that a certain amount of data lets
us train a better model.

The training hyperparameters are included in the
appendix.

### 6.1 Modelling type

All the models we trained can be divided into three
types: single-way segmented, two-way segmented
and finetuned models.

In order to distinguish a language model from a
segmentation method the model names will be given in
**bold** e.g. Unigram is a segmentation model while
**Unigram** is a model trained on data processed by
the corresponding segmentation model.

#### 6.1.1 Single-way segmented

Models of this type — **Morph. segm., BPE, Unigram, Wordpiece** — were trained using single
datasets we got in Section 5.

#### 6.1.2 Two-way segmented

Models of this type — **MS+BPE, MS+Unigram, MS+Wordpiece** — were trained using two datasets
we got in Section 5 combined together. The idea
behind this modelling type is that we want to see if having data processed by different segmentation
methods can help us solve both evaluation tasks on
a high level.

#### 6.1.3 Finetuning

As it was proposed in one of the related works
(Boudreau et al., 2020), pretrained embeddings can
be used in order to improve the performance of
the language models. We decided to try finetuning
though we chose to pretrain not only embeddings
but also RNN layers.

Models of this type — **BPE2MS, Unigram2MS, Wordpiece2MS** — were at first trained using the
Unigram/Wordpiece data and then morphologically
segmented data was used to finetune the trained
model. It’s important to mention that while train-
ing models of this type either datasets volumes were
step-by-step increased by 10 percent e.g. the first
**Unigram2MS** model was trained on 10 percent of
Unigram data and then on 10 percent of morpholo-
gically segmented data. Looking ahead we should
also mention that it turned out there is no need to
lower the learning rate of the model while finetun-
ing it — the perplexity of the model in the end of
training is the same while epoch count (and, accord-
ingly, the training time) becomes approximately 10
times higher.

<table>
<thead>
<tr>
<th>Variant</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input text</td>
<td>Xke’e’ ri nukinaq’</td>
</tr>
<tr>
<td>Canonical</td>
<td>x# ke’e’ x# ri nu# kinaq’S</td>
</tr>
<tr>
<td>Morph. segmen.</td>
<td>x# ke’e’ x# ri nu# kinaq’S</td>
</tr>
<tr>
<td>BPE</td>
<td>xke’# ’# x# ri nukina# q’S</td>
</tr>
<tr>
<td>Unigram</td>
<td>xke’# ’# x# ri nuki# na# q’S</td>
</tr>
<tr>
<td>WordPiece</td>
<td>xke’# ’x# ri nuk# inaq’S</td>
</tr>
</tbody>
</table>

Table 3: Segmentation variants for the K’iche’ sen-
tence Xke’e’ ri nukinaq’ “My beans were ground”.

The canonical segmentation corresponds to /CP-grind.PASS5
the poss.1sg-bean/. The hash symbol, #, indicates that
there is a segment after the current one and the dollar
symbol, $, indicates the last segment in a multi-segment
word.

---

5https://scikit-learn.org/0.24/modules/generated/sklearn.model_selection.ShuffleSplit. html
6.2 Results

As we can see in Table 4, the best models for K’iche’ and Chukchi according to perplexity are Morph. segm. and finetuning models.

<table>
<thead>
<tr>
<th></th>
<th>K’iche’</th>
<th>Chukchi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wd</td>
<td>Ch</td>
</tr>
<tr>
<td>MS</td>
<td>32.59</td>
<td>7.57</td>
</tr>
<tr>
<td>BPE</td>
<td>38.53</td>
<td>8.95</td>
</tr>
<tr>
<td>Uni</td>
<td>35.29</td>
<td>8.20</td>
</tr>
<tr>
<td>WP</td>
<td>148.24</td>
<td>34.45</td>
</tr>
<tr>
<td>BPE2MS</td>
<td>34.03</td>
<td>7.91</td>
</tr>
<tr>
<td>Uni2MS</td>
<td>34.32</td>
<td>7.97</td>
</tr>
<tr>
<td>WP2MS</td>
<td>32.06</td>
<td>7.45</td>
</tr>
<tr>
<td>MS+Uni</td>
<td>35.46</td>
<td>8.24</td>
</tr>
<tr>
<td>MS+WP</td>
<td>34.10</td>
<td>7.92</td>
</tr>
<tr>
<td></td>
<td>54.27</td>
<td>12.61</td>
</tr>
</tbody>
</table>

Table 4: Word (Wd) level and character (Ch) level perplexities for the models (mean scores of 10-fold cross-validation). MS stands for Morph. segm., Uni stands for Unigram, WP stands for Wordpiece. We do not give subword level perplexities as they are not comparable. The best scores are in bold being significantly better according to ANOVA than the others but not outperform each other.

The two-way segmented models show lower scores than Morph. segm. ones, though they are better than the models trained on data of the second origin (BPE, Unigram, Wordpiece segmentors).

It does seem like the usage of morphologically segmented data allows us to improve the performance of the models.

It is worth saying that perplexity scores for different segmentations can’t be compared to each other as is due to the dictionary sizes of all the models being different. This is why we computed the word and character perplexities using the subword ones (Mielke, 2019). Basically, it is just a normalisation of metrics in order to be able to compare them correctly. To do that, we computed the negative log-likelihood of the strings:

\[ nll = \log ppl^{sw} * (C_{sw} + k) \]  

where \( nll \) is negative log-likelihood, \( ppl^{sw} \) is the computed subword level perplexity, \( C_{sw} \) is the total count of subwords in the set and \( k \) is the total count of lines in the set that stands for the count of \(<eos>\) tokens which the model also predicted.

Then we computed word level and character level perplexities using the negative log-likelihood we got on a previous step:

\[ ppl^w = \exp \frac{nll}{C_w + k} \]

\[ ppl^c = \exp \frac{nll}{C_c + k} \]

where \( ppl^w \) is word level perplexity, \( ppl^c \) is character level perplexity, \( nll \) is negative log-likelihood, \( C_w \) is the total count of words in the set, \( C_c \) is the total count of characters in the set and \( k \) is the total count of lines in the set.

As we also did the modelling of one fold using different data volumes, we decided to look at the dependency of the finetuning metrics on the data volume and to compare them to the results of Morph. segm.. As we can see at Figure 1, 40 percent of data is a threshold at which the most models epoch count starts getting lower and the Morph. segm. model perplexity starts becoming smooth; the perplexity is rising due to the growing count of tokens in data. Moreover, the epoch count while doing the finetuning is lower than when training the Morph. segm., while the perplexity scores are being close to each other after the 40 percent threshold, so we can confirm our conclusion that finetuning works well.

7 Predictive text input

Another task to evaluate language models is predictive text input. The idea is that we emulate a person using a smart keyboard while it is offering some predictions, which have to be meaningful. The meaningfulness is important because we assume that the typing person would like to choose from real words/morphs and not some artificial subwords that make at best no sense and in a worst case scenario they may mean something totally wrong (3). The example is given in Turkish because it illustrates the problem really good.

(3) a. araba-m-a
    ‘into my car’

b. arab-am-a
    ‘arab into *vulgar word*’

While evaluating, we look through top 3 model predictions and compare them to the subword we are currently predicting. If they are similar, that prediction is chosen, otherwise we look at the next one. If none of the predictions were correct, we consider...
that the user will have to input the word to end character by character. The input for the model is built up using the remaining morphemes.

7.1 Results

As we can see in Table 5, for K’iche’ the best model is MS+Wordpiece and for Chuckhi the best ones are Morph. segm, BPE2MS, Wordpiece2MS and MS+Unigram. We also decided to include the keystroke savings rate used in the Mi’kmaq paper (Boudreau et al., 2020) in order to be able to compare the results. The group of models which are the best for Chuckhi is the second best for K’iche’. Predictive text metrics do correlate with language-modelling metrics; even though MS+Wordpiece performs the best for K’iche’, the group of Morph. segm, BPE2MS and Wordpiece2MS has both best perplexity and clicks per character scores. We suppose that the models connected with morphological segmentation perform better in this task because the language model, while not being trained on the evaluation data, got resembling training data.

The results for Chuckhi are worse than the results for K’iche’. The reason may be that gold standard for Chuckhi is in telqep Chuckhi, while the corpus used for training is in standard Chuckhi. Another reason may be that words in K’iche’ evaluation data are shorter both segmentwise and characterwise than the Chuckhi words. In case a model can not predict a correct morph, we penalize it by making the whole word be typed character-by-character, so the longer the word is, the more significant mistakes become.

Another issue that may influence the results is that the first subword of each word is typed character by character. We plan to get rid of it in the future in order to be able to evaluate the results better.

8 Discussion

As we can see, the evaluation shows that different models are good at different tasks which paves the way for a discussion if we can say that one model is better than another or not. While not being able to tell the correct answer for this question, we would recommend to try morphological segmentation as it can be used with a statistical one (alongside or by finetuning).

Morphological segmentation can also improve the model performance in predictive text task and other tasks, which were not discussed in this pa-
per. What is important to mention is that there is no need in training models using morphologically segmented data from scratch, the existing models can be finetuned and the results will not differ significantly from the ones of Morph. segm. while the training time will be much lower as shown on Figures 1c and 1a.

As we can see, in all the tasks K’iche’ models have better performance than Chukchi models. While we do not know the particular reason for this, we assume that the polysynthetic language complexity may be hindering the model from training. In the mentioned above paper (Lane and Bird, 2020) the authors also reported that polysynthetic languages have their special challenges such as high word length, complexity, etc.

As we referenced the Mi’kmaq (Boudreau et al., 2020), it seems reasonable to compare the results of their experiments with the results of ours. The results of the evaluation cannot be compared easily because our task was to predict linguistic units, not any kind of units, while in the Mi’kmaq paper words and BPE segments were being predicted; though if we do compare the results, we can see that the best KSR score for Mi’kmaq is 3.81, while the best score for K’iche’ is 4.26. At the same time, the best Chukchi KSR is much worse that the Mi’kmaq score being only 0.79.

Alongside the metrics computed while experimenting there is also a metric which cannot be measured without end-user testing – the sanity check. As mentioned before, the issue with statistical segmentation is that subwords predicted and offered to the user may have no sense for the user or, what is much worse, may carry the wrong meaning. We do suppose that this alone can be a reason to choose morphological segmentation over the regular one because segmentation task is not done in isolation – it serves a purpose in a larger scheme of things. We think that in case the language model will be used in predictive text settings, where the user experience and user reaction is highly relevant, morphological segmentation should be chosen as a subword tokenization method, while statistical segmentation may be chosen while doing machine translation, for example.

9 Future work

We are planning to test several other language models and language modelling metrics in order to find out what correlates best with text prediction scores.

We find it reasonable to experiment on other languages, for example, Turkish, Nahuatl and Yupik, in order to get a better understanding when the use of morphological segmentation is reasonable.

Another task to do is to run an end-user evaluation of multiple segmentations and determine which units are preferred. In order to do this, we would also need to solve the problem of predictive text evaluation that the user has to input the first word character by character. In order to do this, we will possibly have to combine word level and character level based models.

Acknowledgements

Removed for review

References


A Hyperparameters

Here we provide hyperparameter values for the various models to aid in reproduction of the results.

A.1 Morphological segmentation

In this section we describe the best hyperparameter settings that we found for the various tasks.

A.1.1 Morfessor

The best results for both K’iche’ and Chukchi were achieved with this hyperparameters:

A.1.2 NeuralMorphemeSegmentation

The best results for K’iche’ were achieved with this hyperparameters:

A simple and language independent subword tokenizer and detokenizer for neural text processing.

Subword regularization: Improving neural network translation models with multiple subword candidates.

Can you compare perplexity across different segmentations?

Automatic sanskrit segmentizer using finite state transducers. pages 85–90.

Tokenizers.

Revitalization of indigenous languages through pre-processing and neural machine translation: The case of Inuktitut.

Morphological word segmentation on agglutinative languages for neural machine translation.

Morphological word segmentation on agglutinative languages for neural translation.

A morphological analyser for k’iche’.

Morfessor 2.0: Python implementation and extensions for morfessor baseline. D4 julkaistu kehittämis- tai tutkimusraportti tai -selvitys.

Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing.


Syllable-level neural language model for agglutinative language.

Bert: Pre-training of deep bidirectional transformers for language understanding.


The best results for both K’iche’ and Chukchi were achieved with this hyperparameters:

Can you compare perplexity across different segmentations?

Automatic sanskrit segmentizer using finite state transducers. pages 85–90.

Tokenizers.

Revitalization of indigenous languages through pre-processing and neural machine translation: The case of Inuktitut.

Morphological word segmentation on agglutinative languages for neural machine translation.

Morphological word segmentation on agglutinative languages for neural translation.

A morphological analyser for k’iche’.

Morfessor 2.0: Python implementation and extensions for morfessor baseline. D4 julkaistu kehittämis- tai tutkimusraportti tai -selvitys.

Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing.


Syllable-level neural language model for agglutinative language.

Bert: Pre-training of deep bidirectional transformers for language understanding.


The best results for both K’iche’ and Chukchi were achieved with this hyperparameters:
The best results for Chukchi were achieved with this hyperparameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning algorithm</td>
<td>recursive</td>
</tr>
<tr>
<td>training type</td>
<td>type based</td>
</tr>
</tbody>
</table>

Table 6: Morfessor hyperparameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>convolutional layers</td>
<td>3</td>
</tr>
<tr>
<td>window size</td>
<td>3 – 4</td>
</tr>
<tr>
<td>filters</td>
<td>96</td>
</tr>
<tr>
<td>dense output users</td>
<td>64</td>
</tr>
<tr>
<td>context dropout</td>
<td>0.3</td>
</tr>
<tr>
<td>memorize morphemes</td>
<td>no</td>
</tr>
<tr>
<td>memorize ngram counts</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 7: NMS hyperparameters (K’iche’).

The best results for Chukchi were achieved with this hyperparameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>convolutional layers</td>
<td>3</td>
</tr>
<tr>
<td>window size</td>
<td>4–6</td>
</tr>
<tr>
<td>filters</td>
<td>96</td>
</tr>
<tr>
<td>dense output users</td>
<td>20</td>
</tr>
<tr>
<td>context dropout</td>
<td>0.3</td>
</tr>
<tr>
<td>memorize morphemes</td>
<td>no</td>
</tr>
<tr>
<td>memorize ngram counts</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 8: NMS hyperparameters (Chukchi).

A.1.3 NCRF++

The best results for K’iche’ were achieved with this hyperparameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>char. embedding dim</td>
<td>200</td>
</tr>
<tr>
<td>char. hidden vector dim</td>
<td>200</td>
</tr>
<tr>
<td>optimizer</td>
<td>Adagrad</td>
</tr>
<tr>
<td>convolutional layers</td>
<td>4</td>
</tr>
<tr>
<td>use CRF layer</td>
<td>yes</td>
</tr>
<tr>
<td>use char. sequence layer</td>
<td>yes</td>
</tr>
<tr>
<td>use CNN to train for chars</td>
<td>yes</td>
</tr>
<tr>
<td>use CNN to train for words</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 9: NCRF++ hyperparameters (K’iche’).

A.2 Least Significant Deviation values

The LSD-test results for language modelling and predictive text tasks:

<table>
<thead>
<tr>
<th>Task</th>
<th>K’iche’</th>
<th>Chukchi</th>
</tr>
</thead>
<tbody>
<tr>
<td>language modelling</td>
<td>1.494</td>
<td>17.806</td>
</tr>
<tr>
<td>predictive text</td>
<td>14.22e-4</td>
<td>6.779e-4</td>
</tr>
</tbody>
</table>

Table 11: LSD values

A.3 Language modelling

All the models were built results for Chukchi were achieved with this hyperparameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM layers</td>
<td>3</td>
</tr>
<tr>
<td>embedding dim</td>
<td>256</td>
</tr>
<tr>
<td>hidden units per layer</td>
<td>3000</td>
</tr>
<tr>
<td>use regularization</td>
<td>no</td>
</tr>
<tr>
<td>layers dropout</td>
<td>0.4</td>
</tr>
<tr>
<td>RNN layers dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>embeddings dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>remove words from embeddings dropout</td>
<td>0.0</td>
</tr>
<tr>
<td>sequence length</td>
<td>100</td>
</tr>
<tr>
<td>optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>learning rate</td>
<td>1e-3</td>
</tr>
<tr>
<td>weight decay</td>
<td>1.2e-6</td>
</tr>
<tr>
<td>seed</td>
<td>1111</td>
</tr>
</tbody>
</table>

Table 12: Awd-lstm hyperparameters.
<table>
<thead>
<tr>
<th>Case</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>naqaj at k’o wi chi wech</td>
</tr>
<tr>
<td>Morph. Segm.</td>
<td>naqaj at k’o wi chi wech</td>
</tr>
<tr>
<td>Unigram</td>
<td>naqaj at k’o wi chi wech</td>
</tr>
<tr>
<td>Wordpiece</td>
<td>naqaj at k’o wi chi wech</td>
</tr>
</tbody>
</table>

| Raw          | qonojel wa’ pa q’ab’ la oj k’o wi nudyos    |
| Morph. Segm. | rini'mal xutaq loq jun taq’om wuj chi we   |
| Unigram      | rini'mal xutaq loq jun taq’om wuj chi we   |
| Wordpiece    | rini’mal xutaq loq jun taq’om wuj chi we   |

| Raw          | xarumal keta’m konojel ri winaq kkiquumuj le ja’ |
| Morph. Segm. | xarumal keta’m konojel ri winaq kkiquumuj le ja’ |
| Unigram      | xarumal keta’m konojel ri winaq kkiquumuj le ja’ |
| Wordpiece    | xarumal keta’m konojel ri winaq kkiquumuj le ja’ |