SIGMORPHON 2022 Shared Task on Morpheme Segmentation Submission Description: Sequence Labelling for Word-Level Morpheme Segmentation

Leander Girrbach
University of Tübingen, Germany
leander.girrbach@student.uni-tuebingen.de

Abstract

We propose a sequence labelling approach to word-level morpheme segmentation. Segmentation labels are edit operations derived from a modified minimum edit distance alignment. We show that sequence labelling performs well for "shallow segmentation" and "canonical segmentation", achieving 96.06 f1 score (macro-averaged over all languages in the shared task) and ranking 3rd among all participating teams. Therefore, we conclude that sequence labelling is a promising approach to morpheme segmentation.

1 Introduction

This paper describes our participation in the SIGMORPHON 2022 Shared Task on Morpheme Segmentation (Batsuren et al., 2022a). Building on previous work on word segmentation and transliteration by Hellwig and Nehrdich (2018), we propose a sequence labelling approach to morpheme segmentation.

The shared task consists of 2 tracks: Word-level morpheme segmentation and sentence-level morpheme segmentation. Data for this shared task was taken from (Batsuren et al., 2021) and (Batsuren et al., 2022b). Although our approach is applicable to both word-level and sentence-level morpheme segmentation, we focus on word-level segmentation. We only evaluate the zero-shot performance of our word-only segmentation models on sentence-level morpheme segmentation.

Sequence labelling approaches can claim several advantages over the main alternative, namely (neural) encoder-decoder approaches: Sequence labelling does not require beam search for inference, may allow for smaller models, and defines a direct alignment between the input and predictions. The latter property may make models more interpretable and help with error analysis. However, sequence labelling is less flexible than encoder-decoder approaches and requires special handling of cases where the input and target sequences are of different length. However, due to the local structure of morphology, sequence labelling may be sufficient to model morpheme segmentation despite being less expressive than encoder-decoder approaches.

2 Related Work

Morpheme segmentation is a well-established task in computational linguistics (cf. Mager et al. (2020)). Recently, two definitions of morpheme segmentations have emerged: "Shallow segmentation" and "canonical segmentation" (Kann et al., 2016). "Shallow" segmentation means segmenting the input word surface string into morphemic substrings. This kind of segmentation is called "shallow", because no orthographic restoration of morphemes to their "canonical" form is performed (Cotterell et al., 2016). "Canonical segmentation", instead, attempts to restore a standardised form of morphemes. As noted by Kann et al. (2016), this is necessary for synthetic languages where multiple morphemes may be merged. Another source of morpheme merging may arise from phonological or orthographic constraints of the language. The present shared task features both shallow segmentation data (e.g. Czech, Latin), and canonical segmentation (e.g. Italian, English). Since canonical segmentation is a strict generalisation of shallow segmentation, methods that work for all languages in this shared task have to be able to perform canonical segmentation.

However, shallow segmentation allows for a conceptually easier approach, namely sequence labelling (Ruokolainen et al., 2013; Sorokin, 2019). Canonical segmentation has hitherto been defined as a sequence-to-sequence task (Kann et al., 2016; Mager et al., 2020). Of course, various improvements for the sequence-to-sequence setup have been proposed, for example reranking of output hypotheses (Kann et al., 2016), multi task learn-

19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 124 - 130

ing (Kann et al., 2018), pointer-generator networks (Sharma et al., 2018), and imitation learning (Makarov and Clematide, 2018).

In fact, Sorokin (2019) explicitly doubts that canonical segmentation can be approached as a sequence labelling task. However, other approaches have already worked towards approaching canonical segmentation as a sequence labelling task: Cotterell et al. (2016) take a middle ground by allowing only for a maximum number of insertions. Ribeiro et al. (2018) train a model to first predict insertion positions in the input sequence. Then, they use a sequence labelling model on the augmented input string to predict the labels. While similar to our approach, we augment the labels instead of the input string. Therefore, our method remains end-to-end trainable. Finally, Hellwig and Nehrdich (2018) propose a sequence labelling approach to Sanskrit word segmentation, which includes restoring original forms that have been merged due to a phonological process called Sandhi.

Therefore, our work extends the method proposed by Hellwig and Nehrdich and thereby shows that canonical morpheme segmentation can be approached effectively as a sequence labelling task.

3 Method

3.1 Data preprocessing

We propose an adaption of the Sanskrit word segmentation method by Hellwig and Nehrdich (2018) for word-level morpheme segmentation. The main idea is to redefine morpheme segmentation as a sequence labelling task. In particular, for each character in the input word, we predict an edit operation. Edit operations can be copying, deletion, or substitution. Here, insertion is a special case of substitution. An example is in Table 1.

In order to redefine morpheme segmentation as a sequence labelling task, we need alignments of input words and the segmented morphemes. We propose to align words and morphemes by the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) with the following parameters: Only equal characters can be matched, and we set the gap cost to 0. Here, we treat all morphemes as one sequence of characters. From all alignments with maximum score according to the Needleman-Wunsch algorithm (i.e. minimum edit distance), we choose the alignment with the maximum sum of squared lengths of contiguous aligned segments. The idea is to copy longer morphemes directly from the input word and insert shorter morphemes. Furthermore, we want to avoid splitting predicted morphemes. Instead, we want to copy as many complete morphemes from the input word as possible. An example is in Table 2.

After having aligned words to their respective morphemes, we obtain data for sequence labelling in the following way: Word characters that are aligned to corresponding characters in the morpheme string are copied. Word characters that are aligned to gaps in the morpheme string are deleted. Morpheme separation characters and possible following characters to complete a morpheme are aligned to gaps in the input word. We prepend these to the label of the input word character following the gap. Remaining morpheme string characters (which do not appear behind a morpheme separation character) that are aligned to gaps in the input word are appended to the label of the next input word character before the gap. In Table 3, we show the resulting labels for the English word “entabulates”. Note that our eventual labelling makes more use of copying than the simple edit operation example given in Table 1.

3.2 Models

For sequence labelling, we use a plain 2-layer BiLSTM model. For each position of the input sequence, the model predicts exactly one edit operation. Ground-truth labels for supervised training are derived as explained in Section 3.1.

Our submission is produced by single models (i.e. no ensembling) trained in a supervised fashion. Models have 2 layers with 256 hidden units each. We apply dropout with probability 0.1 after the first BiLSTM layer. We use the AdamW optimizer (Loshchilov and Hutter, 2019) with initial learning rate 0.001 and weight decay 0.001. We divide the learning rate by 2 after 3 epochs without improvement of word error rate (WER) on the development set. Note that WER is a stricter metric than f1 score and edit distance, which are the shared task’s official evaluation metric. Each model is trained for 50 epochs with batch size 32, but we only keep the checkpoint with lowest WER on the development set.

3.3 Zero-shot sentence-level segmentation

For sentence-level segmentation, we proceed in the following way: Since all sentence-level languages (Czech, English, Mongolian) are also part of the word-level track, we can use our models from the

e	n			t	a	b	u	l			a	t	e			s		
e	n	␣	@	@t	a	b		le	␣	@	@a	t	e	␣	@	@	s	
C	C			S	C	C	D	S			S	C	C			S		

Table 1: Edit operation to transduce "entabulates" to its morphemic segment string "en␣@@table␣@@ate␣@@s". "␣@@" is the morpheme separation symbol in the given data, "S" means substitution, "C" means copy, and "D" means deletion.

m	a	m	m	a	␣	@	@	a	r	e	␣	@	@	e	r	a	n	n	o
m	a	m	m											e	r	a	n	n	o
m	a	m	m							e					r	a	n	n	o

Table 2: Example for different alignments of the Italian word "mammare" to its morpheme segmentation string "mamma␣@@are␣@@eranno". The upper alignment is preferred, because it contains longer contiguous aligned subsequences.

word-level track for sentence-level segmentation. We retrieve all space-separated tokens from the sentence-level test data and segment each token individually, thus creating a dictionary mapping tokens to their word-level segmentation. Then, we replace each token in the sentence by the segmentation according to the word-only dictionary. Tokens that only consist of punctuation are copied directly from the input sentence without any segmentation.

This method obviously ignores all sentence-level information that could help with disambiguating multiple possible segmentations. However, we still find it interesting to see how well a word-level-only segmentation model performs on the sentence level for the different languages.

4 Results

Word-level segmentation Official test set results[1] for word-level segmentation are in Table 4. f1 score is greater than 0.9 for all languages. In terms of macro-averaged f1 score, our submission ranks 3rd out of 5 participating teams (excluding baseline) who submit predictions for all languages.

In our results, we do not see any trends regarding a relationship between number of generated labels and performance. The language with weakest performance, English, has the 2nd highest number of generated labels, but the language with highest number of generated labels, Russian, is the language with second best performance. Czech, the number with the lowest number of generated labels, is the language with 3rd worst performance, but Latin, the language with 2nd lowest number of generated labels, is the language with best performance. This suggests that our data preprocessing method does not obscure the segmentation difficulty inherent in a language.

Remember that differences in the amounts of labels is due to different annotation approaches in the data: For Czech and Latin, only "shallow" morpheme boundaries are annotated, i.e. where morpheme boundaries are in the input string. For other languages, restored morphemes are annotated that are contracted when forming the word. For example, the English word "entabulates" is segmented as "en␣@@table␣@@ate␣@@s" where "u" is inserted to form the word, but the "e" in "table" is deleted.

Sentence-level segmentation Official test set results for sentence-level segmentation are in Table 5. Sentence-level performance is worse than word-level performance for all languages. While the decrease in performance is still moderate for English and Czech, we see a very high decrease in performance for Mongolian. This suggests that the number of ambiguous tokens in English and Czech is relatively not very high, while a lot of ambiguous words exist in Mongolian.

5 Error Analysis

Frequent Errors As claimed in Section 1, our proposed sequence labelling method allows for direct comparison of the predicted labels to labels created by our preprocessing. Here, we provide a short analysis of the most frequent errors made by our English word segmentation model. To this end, we apply the preprocessing method described in Section 3.1 to the test set released by the shared

[1] Taken from https://github.com/sigmorphon/2022SegmentationST/tree/main/results

e	n	t	a	b	u	l	a	t	e	s
C	C	␣@@+C	C	C	D	C+e	␣@@+C	C	C	␣@@+C

Table 3: Labels generated by our data preprocessing method for English word "entabulates" with morpheme segment string "en␣@@table␣@@ate␣@@s". Our labels allow for special symbols (C = copy, D = delete) and arbitrary string insertions. Labels do not have to contain special symbols. "+" here means concatenation and is not to be read as part of the label.

Lang.	Dis.	P	R	F1	# Lbls
ces	0.18	93.95	92.81	93.38	2
eng	0.25	90.51	90.52	90.51	1740
fra	0.28	93.56	93.96	93.76	1275
hun	0.11	98.21	98.97	98.59	442
spa	0.11	97.88	97.98	97.93	1311
ita	0.20	95.50	95.97	95.73	850
lat	0.01	99.35	99.39	99.37	4
rus	0.15	98.16	98.26	98.21	1809
mon	0.10	96.91	97.13	97.02	442
Avg.	0.15	96.00	96.11	96.06	

Table 4: Official word-level results for our system (all languages). Dis is edit distance, P is precision, R is recall, and F1 is f1 score. # Lbls is the number of labels generated by our data preprocessing method (see Section 3.1).

Lang.	Dis.	P	R	F1
ces	2.50	89.52	88.42	88.97
eng	1.78	87.83	89.58	88.69
mon	9.85	69.59	67.55	68.55

Table 5: Official sentence-level results for our system (all languages). Dis is edit distance, P is precision, R is recall, and F1 is f1 score.

b	i	o	m	e
C	C	C	␣@@o+C	C
C	C+o	␣@@+C	C	C

Table 6: An example where different labels result in the same (correct) segmentation: "biome" ⟶ "bio␣@@ome".

task organisers after the submission deadline. Then, we calculate a confusion matrix of the labels predicted by our model and the labels created by the preprocessing method.

First, however, we want to note that in few cases even incorrect predictions may lead to correct segmentations. This is due to ambiguity in the alignments. For example, consider the test item "biome" with ground truth segmentation "bio @@ome". In Table 6 we show that our model's prediction differs from the generated alignment, but the resulting segmentations are identical. In the English test set, this is the case for 118 words, so we do not think this is a problem for our subsequent error analysis. In total, there are 8615 words ($\approx$ 15% of all test words) with incorrect segmentation.

The most common errors are predicting morpheme boundaries where actually no morpheme boundaries are, i.e. predicting "␣@@ + C" instead of "C", which happens 3820 times, and missing to predict morpheme boundaries, i.e. predicting "C" instead of "␣@@ + C", which happens 3786 times. An example is "lemming": Our models predicts "lem␣@@ing" instead of "lemming". A morpheme boundary was overlooked in "sanity": Our model predicts "sanity" instead of "sane␣@@ity".

The next most frequent errors are missing to insert an "e", i.e. predicting "C" instead of "C+e", which happens 499 times, and inserting a superfluous "e", i.e. predicting "C + e" instead of "C", which happens 414 times. For example, our model predicts "wok␣@@ism" instead of "woke␣@@ism" for "wokism" and "ominouse␣@@ity" instead of "ominous␣@@ity" for "ominosity".

The last error type in the top 5 most frequent errors is not deleting an input character, i.e. predicting "C" instead of "D", which happens 448 times. For example, our model predicts "charr␣@@y" instead of "char␣@@y" for "charry".

Please note that there can be multiple errors in the predictions for a single input word. However, in most cases (5873), there is only one incorrectly predicted label. In 2008 cases, there are 2 incorrectly predicted labels. The extreme case is "alsakharovite", for which 9 of the predicted labels are incorrect: Our model simply predicts to copy each character, but the ground truth is given as "Aleksey␣@@ite". This shows that our model struggles with proper names, which is not surprising.

In conclusion, this analysis shows that the largest gains in performance can be expected from improving the shallow part of segmentation, while there are fewer individual morpheme reconstruction errors. Therefore, a possible future extension of our proposed model is switching to a multi task setting, where one task is to predict morpheme boundaries and the other task is to reconstruct partial or missing morphemes. In the setting evaluated here, these tasks are approached jointly.

Label Embeddings Additionally, we inspect the learned (English) label embeddings and try to see whether any patterns emerge. To this end, we retrieve the 50 most frequent labels (accounting for 98% of all labels in the dataset excluding the simple copy label) and their label embeddings (columns in the final linear prediction layer). We cluster the label embeddings by affinity propagation. The advantage of affinity propagation is that we do not have to specify the number of clusters. We use the scikit-learn implementation of affinity propagation (Pedregosa et al., 2011) with default parameters. The discovered clusters are in Table 7.

In total, the clustering generates 7 clusters, of which 4 clusters contain multiple labels and 3 clusters contain only 1 label. No cluster is completely pure, but we can observe the following trends: Cluster 0 mostly represents morpheme boundary labels followed by a copy operation, i.e. "insert morpheme boundary before this character". Cluster 2 mostly represents substitutions and Clusters 1 and 3 mostly represent insertions. We cannot say anything definite about the 1 element clusters.

From these observations we conclude that the model learns to distinguish different edit operations (insertion, substitution) and also learns to distinguish inserting morpheme boundaries from other edit operations. This provides further evidence that changing our approach to a multi task setting may be worth exploring.

Generalisation to Unseen Substitutions Finally, we want to address the problem that the finite number of labels generated by our data preprocessing method (see Section 3.1) may not allow the model to generalise to substitutions not seen in the training data.[2] To collect evidence concerning this problem, again for the English test set, we find all words that cannot be generated by our model because generating them would require labels that were not generated from the training data. In total, we find 35 such words (of 57755 test words in total). Furthermore, upon manual inspection, we find many of these cases either caused by proper names with irregular or non-English segmentation, for example "Staffie" is segmented as "Stafford␣@@shire␣@@ie" or "Lebos" is segmented as "Lebanon␣@@ese␣@@o␣@@s", or annotation errors, for example "unlid" is segmented (in the gold data) as "un#Etymology_2␣@@lid" or "perfosfamide" is segmented as "hydroperoxy␣@@fosfamide". However, we also discover a genuine problem of our model, namely that it does not have any labels to generate hyphens ("-"). This proves that the problem can be substantial, if there had been more hyphenated words in the test data.

On the other hand, hyphens do not appear as character in any train set segmentation, so it is generally hard to anticipate peculiarities of the test set. The case of proper names could perhaps be handled by external resources, but this does not scale well.

Summing up, we acknowledge that this is an issue not solved entirely by our approach. However, it does not cause many problems for this shared task. This being said, this shared task provides a lot of data for the featured languages, so the missing label problem may become more serious for low resource languages or settings.

One step towards approaching this issue could be not only generating labels from one alignment of the word to its morpheme segmentation string (see Section 3.1), but from multiple alignments. This could potentially also regularise the model or allow for different training strategies than the standard supervised training. Another possibility could be to equip the model with the ability to generate new

[2]We thank the reviewers for pointing out this problem.

Cluster ID	Labels
0	“C”, “␣@@ + C”, “␣@@e + C”, “␣@@a + C”, “␣@@i + C”, “␣@@o + C”, “C + t”, “␣@@l + C”, “D + n”, “␣@@ + C + e”, “C + ␣@@s”, “C + te”, “C + ic”, “␣@@is + C”, “␣@@i + C + e”, “C + m”, “␣@@en + C”, “␣@@a + C + e”, “C + l”, “␣@@t + C”
1	“D + y”, “D + ␣@@y”, “C + um”, “D + e”, “C + s”, “D + e␣@@y”, “D + o”, “D + a”, “D + d”
2	“C + e”, “C + y”, “C + a”, “C + o”, “C + us”, “C + on”, “D + sis”, “C + ion”, “C + ous”, “D + p”
3	“D”, “D + i”, “C + ␣@@y”, “D + is”, “D + x”, “C + n”, “D + s”
4	“C + ne”
5	“D + ce”
6	“D + de”

Table 7: Clusters of labels discovered by affinity propagation clustering of the label embeddings of the 50 most frequent English labels.

labels, for example by predicting a fixed number of label subsymbols from each input character. Symbols could be blanks, unigrams, or ngrams. This would relax the constraint that labels have to be entirely known beforehand, while maintaining a sequence labelling setup.

6 Conclusion

We presented a sequence labelling approach for word-level morpheme segmentation. Models trained with this approach yield strong performance on all languages (word-level) despite of not using ensembling and using a simple BiLSTM encoder. Error analysis for English reveals that the model is often only wrong in 1 single place, struggles with proper names, and most frequently errors are caused by incorrect prediction of morpheme boundaries.

Acknowledgements

We thank Çağrı Çöltekin for providing access to computation resources and giving feedback on a draft version of this paper. We thank the task organisers for organising this shared task. Finally, we thank the reviewers for their helpful comments and suggestions.

References

Khuyagbaatar Batsuren, Gábor Bella, and Fausto Giunchiglia. 2021. MorphyNet: a large multilingual database of derivational and inflectional morphology. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 39–48, Online. Association for Computational Linguistics.

Khuyagbaatar Batsuren, Gábor Bella, Aryaman Arora, Viktor Martinović, Kyle Gorman, Zdeněk Žabokrtský, Amarsanaa Ganbold, Šárka Dohnalová, Magda Ševčíková, Kateřina Pelegrinová, Fausto Giunchiglia, Ryan Cotterell, and Ekaterina Vylomova. 2022a. The sigmorphon 2022 shared task on morpheme segmentation. In *19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*.

Khuyagbaatar Batsuren, Omer Goldman, Salam Khalifa, Nizar Habash, Witold Kieraś, Gábor Bella, Brian Leonard, Garrett Nicolai, Kyle Gorman, Yustinus Ghanggo Ate, et al. 2022b. Unimorph 4.0: Universal morphology. *arXiv preprint arXiv:2205.03608*.

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and morphological segmentation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 664–669, San Diego, California. Association for Computational Linguistics.

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2754–2763, Brussels, Belgium. Association for Computational Linguistics.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 961–967, Austin, Texas. Association for Computational Linguistics.

Katharina Kann, Jesus Manuel Mager Hois, Ivan Vladimir Meza-Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 47–57, New Orleans, Louisiana. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Manuel Mager, Özlem Çetinoğlu, and Katharina Kann. 2020. Tackling the low-resource challenge for canonical segmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5237–5250, Online. Association for Computational Linguistics.

Peter Makarov and Simon Clematide. 2018. Neural transition-based string transduction for limited-resource setting in morphology. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 83–93, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Joana Ribeiro, Shashi Narayan, Shay B. Cohen, and Xavier Carreras. 2018. Local string transduction as sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1360–1371, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 29–37, Sofia, Bulgaria. Association for Computational Linguistics.

Abhishek Sharma, Ganesh Katrapati, and Dipti Misra Sharma. 2018. IIT(BHU)–IIITH at CoNLL–SIGMORPHON 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 105–111, Brussels. Association for Computational Linguistics.

Alexey Sorokin. 2019. Convolutional neural networks for low-resource morpheme segmentation: baseline or state-of-the-art? In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 154–159, Florence, Italy. Association for Computational Linguistics.