Subword-Delimited Downsampling for Better Character-Level Translation

Lukas Edman Antonio Toral Gertjan van Noord
Center for Language and Cognition
University of Groningen
(j.l.edman, a.toral.ruiz, g.j.m.van.noord)@rug.nl

Abstract

Subword-level models have been the dominant paradigm in NLP. However, character-level models have the benefit of seeing each character individually, providing the model with more detailed information that ultimately could lead to better models. Recent works have shown character-level models to be competitive with subword models, but costly in terms of time and computation. Character-level models with a downsampling component alleviate this, but at the cost of quality, particularly for machine translation. This work analyzes the problems of previous downsampling methods and introduces a novel downsampling method which is informed by subwords. This new downsampling method not only outperforms existing downsampling methods, showing that downsampling characters can be done without sacrificing quality, but also leads to promising performance compared to subword models for translation.

1 Introduction

Character-level models (henceforth character models) have recently sparked interest in their potential applicability across a wide range of NLP tasks. They promise a tokenization-free approach, while also potentially allowing the model to quickly recognize similarities between words based on their spelling. However, as of yet, character models have not stood up to the level of subword models, mainly due to their similar performance while being significantly slower and more expensive to train due to their longer input sequences (Xue et al., 2022).

To alleviate the problem of training time, several methods have been proposed to initially downsample characters into shorter sequences, which are then fed into the encoder or decoder. For discriminative tasks, these can be applied without any loss in performance (Tay et al., 2021), however for generative tasks like NMT, the performance is either untested or lacking when compared to character models without downsampling (Libovický et al., 2021).

Seeing as subword tokenization is essentially a form of downsampling and performs quite well, the idea of downsampling is not inherently flawed. However, attempts to downsample within the neural network have not achieved similar performance for translation. This begs the question, why are current neural downsampling methods underperforming for translation?

In this work, we make three main contributions:

1. We analyze the existing downsampling methods based on their position, length, and morpheme consistency.
2. We introduce a novel neural downsampling method based on subwords that outperforms existing downsampling methods.
3. We make the necessary modifications to allow for variable-length downsampling and upsampling in an encoder-decoder architecture.

We start by providing an overview of the prior work in character-level NLP in Section 2. We then compare the various downsampling methods, noting the 3 main advantages of downsampling based on subwords in Section 3. Next, we cover the modifications necessary to do variable-length downsampling for translation in Section 4. We follow this with experimental details (Section 5), results (Section 6), analysis (Section 7), discussion (Section 8), conclusion (Section 9), and finally the limitations (Section 10).

2 Related Work

Character-level models have been of interest for several years, notably used for character-level translation prior to the advent of Transformer models with some success (Costa-Jussa and Fonollosa, 2016; Lee et al., 2017).

Lee et al. (2017) raises the issue that the length of the sequences require the models to potentially
Table 1: Summary of previous work on character-level models compared to ours. Note that many of the works have tested multiple models, so we only include their main additions which are unique in research.

<table>
<thead>
<tr>
<th>Work</th>
<th>Encoder</th>
<th>Decoder</th>
<th>Downsampler</th>
<th>Encoder</th>
<th>Decoder</th>
<th>Upsampler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lee et al. (2017)</td>
<td>Fixed</td>
<td>—</td>
<td>CNN</td>
<td>LSTM</td>
<td>LSTM</td>
<td>LSTM</td>
</tr>
<tr>
<td>Boukkouri et al. (2020)</td>
<td>Word</td>
<td>—</td>
<td>CNN</td>
<td>Transformer</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Tay et al. (2021)</td>
<td>Fixed</td>
<td>—</td>
<td>GBST</td>
<td>Transformer</td>
<td>Transformer</td>
<td>—</td>
</tr>
<tr>
<td>Libovický et al. (2021)</td>
<td>Fixed</td>
<td>Fixed</td>
<td>CNN</td>
<td>Transformer</td>
<td>Transformer</td>
<td>LSTM</td>
</tr>
<tr>
<td>Xue et al. (2022)</td>
<td>—</td>
<td>—</td>
<td>GBST</td>
<td>Transformer</td>
<td>Transformer</td>
<td>LSTM</td>
</tr>
<tr>
<td>Edman et al. (2022)</td>
<td>Fixed</td>
<td>Subword</td>
<td>CNN</td>
<td>Transformer</td>
<td>Transformer</td>
<td>LSTM</td>
</tr>
<tr>
<td>Ours</td>
<td>Subword</td>
<td>Subword</td>
<td>CNN</td>
<td>Transformer</td>
<td>Transformer</td>
<td>LSTM</td>
</tr>
</tbody>
</table>
as a third method, buffered fixed-size downsampling (“Buffered Fixed”), which we introduce to better understand the importance of position and morpheme consistency. Table 2 shows an example of these downsampling methods.

Positional Consistency The first factor we consider is the importance of positional consistency, that is, where a word begins within a downsampling block. For example, consider the following two sentences, with alternating colors denoting the chunking of character sequences when using a fixed-size downsampling factor of 4:

He is going to the store.
She is going to the store.

Words such as “is” or “the” end up in 2 tokens for some sentences, while only in 1 for others. Meanwhile, longer words such as “going” and “store” can be split several different ways, leading to several different potential representations depending on the sentence. This positional inconsistency introduces an extra level of difficulty to the model, which we expect results in worse performance. Of the 4 methods tested, Fixed is the only one that suffers from this positional inconsistency.

Length Consistency The consistency of the lengths of downsampling blocks is also important, particularly in the case of longer words. In the CNN downsampler, the max pooling acts as a bottleneck, making it more difficult for the model to learn a complete representation for words with many characters. For example, in Table 2, we see that the WDD method downsamples “Characters” into a single block, which means the max pooling downsamples that word by a factor of 10. Furthermore, the LSTM in the upsampling module may have difficulty decoding a long sequence of characters from a single hidden representation.

This consistency mainly affects the WDD method. There is also a small amount of inconsistency in the SDD method, but this can be greatly minimized by setting a maximum subword token length (see details in Section 4.2).

Morpheme Consistency The third and final benefit of SDD is its creation of more morphologically consistent tokens. When splitting words into multiple subword tokens, it may be better for a model to split along the morpheme boundaries rather than every 4 characters, as the importance of characters can vary. Observe the effect of fixed-size splitting on various verbs:

He is going to the store.
She is going to the store.

The “ing” ending can suffer a sort of positional inconsistency within the word itself. Additionally, these fixed-size splits can be detrimental due to the imbalance of information in the resulting downsampled tokens. Referring back to our example in Table 2, even if tokens are corrected positionally, the word “great” is split into two tokens, with the last character getting its own dedicated token while carrying minimal information.

The Buffered Fixed method offers positional consistency and length consistency, but not morpheme consistency, so comparing this to SDD should tell us the importance of this third factor.

4 Architecture

We now explain the architecture used in our experiments. We build off of the previous work by using the CNN downsampling architecture followed by the Transformer and using Libovický et al. (2021)’s two-step decoding with an LSTM for upsampling. This previous work was only applied to
fixed-length downsampling and upsampling, however the aforementioned WDD and SDD methods require variable-length downsampling and upsampling. Thus, we explain how this is accomplished in the next two subsections. Figure 1 shows the architectures used in our translation experiments.1

4.1 Variable-length Downsampling

The downsampling module for WDD and SDD is identical to that of the fixed-size model, with the exception that the max pooling is computed over all characters in a word or subword. On the decoder side, we additionally require the lengths of each word or subword token in order to create a causal mask which allows the association of characters within the same block, while preventing the association of characters from future blocks.

4.2 Variable-length Upsampling

While the downsampling module ensures that the Transformer receives word or subword-level tokens, we still require a method for upsampling back to characters. Libovický et al. (2021) introduced an effective two-step decoder, consisting of the Transformer followed by an LSTM which takes as input the hidden representation of the Transformer decoder, the character embedding of the previous character, and the previous LSTM hidden state. This has previously only been applied to methods with a fixed-size downsampling, and as such we need to make some modifications to allow it to work with variable-length sequences.

The top blocks of figure 1 show the original and modified versions of the two-step decoder. The input to the LSTM is first modified. In the original case, with a downsampling factor of 4, the hidden representation is repeated 4 times, and each is concatenated with individually learned character embeddings for the block of 4 characters. The LSTM then must predict the next block, which effectively means each character generated is conditioned on the character 4 steps back.

With the modified version, the hidden representation is repeated the same number of times as the length of the next block plus one, as we add in an end-of-word token for each block to the character embeddings and labels.2 Each hidden representation is concatenated with the character embeddings, shifted over by 1. Although the character embeddings fed to the LSTM are no longer associated with the respective hidden block, since the embeddings are individually learned, no information from future blocks is accessible (thus avoiding the leaking issue described in Edman et al. (2022)).

Since LSTMs are known to struggle as the lengths of sequences get longer, we also limit the lengths of each subword,3 which prevents the joining of subwords beyond a specified character length. This minimizes the length inconsistency previously mentioned in Section 3.

---

1Our encoder-only experiments use the same encoder architecture (including the downsampler) with a linear output layer added.

2The end-of-word token is added to stop the LSTM’s generation of a word at generation time.

3We use the argument from SentencePiece max_sentencepiece_length to achieve this.
Table 3: Comparison of the 4 different downsampling methods, with an ablation of positional consistency (Buf. Fixed - Fixed), length consistency (SDD - WDD), and morpheme consistency (SDD - Buf. Fixed) using two evaluation metrics (BLEU and COMET).

5 Experimental Setup

Our code is made available on GitHub.4 We experiment with translation, using the encoder-decoder, as well as two encoder-only tasks: NLI and review classification.5 While we focus mainly on improving translation, we also include these encoder-only tasks to test the importance of the choice of downsampling method for non-generative tasks.

We compare several models, including the 4 downsampling methods (Fixed, Buffered Fixed, WDD, and SDD), as well as a subword-level model and a character-level model,6 both of which use the standard Transformer architecture, requiring no downsampling or upsampling module.

For translation, we experiment with 3 language pairs: English–Arabic, English–German, and English–Turkish. We chose these language pairs as they exhibit different levels of linguistic similarity and morphological richness. We evaluate our models with BLEU (Papineni et al., 2002) and COMET (Rei et al., 2020).

The full details of the datasets used for all tasks can be found in Appendix A. To keep our research eco-friendly and to allow for faster iteration on our models, we train our models on smaller translation datasets, consisting of roughly 200 thousand sentence pairs per language pair, and roughly 500 thousand and 5 million sentences for NLI and review classification, respectively. We discuss the potential for these models in the high-resource setting in Section 8.

For SDD, we change the value of the max_sentencepiece_length argument in SentencePiece to achieve an effective downsampling factor as close to our fixed-size downsampling counterparts as possible.7 To maintain a comparability to a strict downsampling of 4, we set the max_sentencepiece_length and the vocabulary size such that the average downsampling factor is around 4. We achieve this by first setting the vocabulary size to the size recommended by VOLT (Xu et al., 2020), then lowering the max_sentencepiece_length until the average downsampling factor is close to 4.8 We chose 4 as it is roughly equivalent to the ratio of the number of bytes per subword token when comparing the ByT5 and mT5 models across all the languages in the mC4 corpus (Xue et al., 2022).

Our parameter setup for models, vocabulary, and training are noted in Appendix B.

6 Results

We start by comparing the 4 downsampling methods, discerning the importance of the positional, length, and morpheme consistency. We then compare our best-performing downsampling method to our two standard baselines: the subword-level and character-level models. Third, we compare these models in their ability to generalize to out-of-domain datasets. Lastly, we look at the encoder-only tasks, comparing all methods thus far.

6.1 Comparing Downsampling Methods

First comparing the various downsampling methods Table 3, we see that as expected from Section 3 our novel SDD method performs best overall.

Of the three factors, positional consistency appears most important, making up the greatest portion of the performance increase. Given that

---

4https://github.com/Leukas/SDD
5Review classification is classifying product reviews from 1 to 5 stars based on the review title and content.
6By character-level we in fact mean byte-level, aligning with previous work. This applies to our downsampling models as well.
7By default, it is set to 16, which inhibits the performance of the LSTM in the decoder.
8The average downsampling factor is calculated by averaging the number of bytes per subword token in the training data.
Table 4: Translation results of traditional subword models and character models without downsampling compared to character models with subword-delimited downsampling (SDD). Green (*) and red (†) denote a significant positive or negative difference (p < 0.05) with respect to the Subword model.

<table>
<thead>
<tr>
<th>Method</th>
<th>de-en</th>
<th>en-de</th>
<th>ar-en</th>
<th>en-ar</th>
<th>tr-en</th>
<th>en-tr</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subword</td>
<td>27.23</td>
<td>24.08</td>
<td>25.59</td>
<td>11.22</td>
<td>14.73</td>
<td>11.30</td>
<td>19.02</td>
</tr>
<tr>
<td>Char</td>
<td>27.37</td>
<td>24.32</td>
<td>26.34</td>
<td>11.07</td>
<td>15.52</td>
<td>11.97</td>
<td>19.03</td>
</tr>
<tr>
<td>SDD</td>
<td>27.78</td>
<td>25.12</td>
<td>27.74</td>
<td>11.95</td>
<td>15.99</td>
<td>11.31</td>
<td>19.98</td>
</tr>
</tbody>
</table>

positional inconsistency means there is often no subword-like structure to the downsampled tokens, it is understandable that the model has the most difficulty translating in this scenario.

Length consistency contributes some improvement as well, showing that there is indeed a bottleneck effect when downsampling longer words, and that there is merit in the model splitting such words into multiple tokens prior to the Transformer.

Morpheme consistency is least important; however it seems to have a larger impact on the more morphological languages of Arabic and Turkish. This gives evidence towards the idea that the subword splitting of SentencePiece is likely more conducive to translating morphemes than translating 4-character chunks. Although we use SentencePiece due to its ubiquity, there are arguably better subword tokenizers with respect to adherence to morphology, such as LMVR (Ataman et al., 2017). These tokenizers are fully compatible with SDD, and they may further increase the disparity we see in morpheme consistency.

6.2 Comparing to Subword and Character Models

We now compare the performance of SDD to subword and character models in Table 4.

Overall, SDD outperforms both subword and character models on both BLEU and COMET. It performs significantly better than the subword model in 4 and 3 cases according to BLEU and COMET, respectively. Only in 1 case in terms of COMET does SDD lead to significantly worse performance.

The Arabic–English language pair shows the largest improvement over the baselines. The reason for this is unclear, though it is the only language pair with an entirely different character set between the source and target (save for numbers, symbols, and some proper nouns).

As SDD was originally intended to shore up the weaknesses of the previous downsampling methods, the aim was to perform on par with the subword and character models. The slight increase in performance shows that there is some benefit in using a combination of the two, namely using character-level input while operating on a subword-level within the Transformer encoder-decoder.

6.3 Out-of-domain Generalization

Boukkouri et al. (2020) found that with CharacterBERT, the models generalized better to out-of-domain encoder-only tasks such as classification of medical data. They argue that because CharacterBERT does not have a strict vocabulary, and it learns more general properties of language which can be useful for unseen words. As we see evidence of the same in our embedding analysis (see Section 7), we similarly test our models on two out-of-domain translation datasets. For all languages, we test on FLoRes (Goyal et al., 2022), which consists of Wikipedia data. For English–German, we additionally test on the WMT21 Biomedical Shared Task test set.

The FLoRes and biomedical results are shown in Tables 5 and 6, respectively. Here we see the SDD model and character model both outperforming the subword model, as expected. The results on FLoRes favor the SDD model, while the biomedical results favor the character model. The biomedical data is arguably “more out-of-domain” than FLoRes, since it contains a large amount of medical terminology unlikely to appear in the training data. This may indicate that the character downsampling is somewhat sensitive to the vocabulary it is trained on, and as such it generalises better than the subword model but not as well as the character model.

Since the SDD model only uses a subword vocabulary to determine the lengths for downsampling, it is perhaps possible to use a different vocabulary when switching domains, namely one that offers better downsampling for the out-of-domain words. We leave this for future research.
6.4 Encoder-only Tasks

The accuracies achieved in the encoder-only tasks are shown in Table 7. The character-level model has surprisingly low accuracies. Given that the main difference between the character model and the other models is the longer sequence length fed into the Transformer, we expect the complexity of the self-attention patterns necessary for these tasks is more difficult to learn when data is limited.

<table>
<thead>
<tr>
<th>Method</th>
<th>NLI (%)</th>
<th>RC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subword</td>
<td>80.14</td>
<td>73.90</td>
</tr>
<tr>
<td>Char</td>
<td>60.75</td>
<td>68.86</td>
</tr>
<tr>
<td>Fixed</td>
<td>81.01</td>
<td>73.28</td>
</tr>
<tr>
<td>Buf. Fixed</td>
<td>81.89</td>
<td>73.69</td>
</tr>
<tr>
<td>WDD</td>
<td>81.89</td>
<td>73.69</td>
</tr>
<tr>
<td>SDD</td>
<td>81.30</td>
<td>74.22</td>
</tr>
</tbody>
</table>

Table 7: Results of NLI and review classification (RC).

Our SDD method outperforms both subword and character baselines on both tasks. Unlike with translation, the downsampling models show little difference in performance. As SDD is intended to help by providing a consistent tokenization, it seems that this consistency is less important for sequence classification tasks. This is probably because there is no need for character or word recovery: the model does not need to reconstruct any of its input, so it can potentially lose some character information while still learning to correctly classify.

A token classification task such as part-of-speech tagging may be more difficult for the fixed-size downsampling model, although it is not clear how to apply such a model to a token classification task, given the mismatch in downsampling blocks and token labels. The SDD model can however be applied to token classification in the same manner a subword model would be applied.

Another reason for there being more of a gap in the performance for translation might be that the cross-attention is what benefits most from the three token consistency factors. Inconsistent tokens on both the source and target side likely make learning what to attend to quite difficult.

7 Embedding Analysis

The sole difference in the architecture of the encoder-only models is their embeddings, or the hidden representations prior to being fed into the Transformer. As such, we extract these embeddings for words to analyze their differences. We compare the models in their word embedding similarity (i.e. cosine similarity) for pairs of words. We generate 4 test sets:

1. Grammatical pairs - Pairs which share a common lemma.
3. Far pairs - Pairs with a max Levenshtein distance.
4. Far Synonyms - Synonyms with a max Levenshtein distance.

The first two sets are meant to distinguish beneficial character-level similarity (e.g. “take” and “takes”) from detrimental similarity (e.g. “pour” and “tour”). The latter two sets are meant to distinguish the opposite, when a lack of similarity may be appropriate (e.g. “bulb” and “faster”) or inappropriate (e.g. “camp” and “tent”). We limit the vocabulary used in the test sets to words in the subword vocabulary that only require a single token. We also filter out tokens which have no synonyms, effectively also removing non-words and non-English words. Lemmas and synonyms are gathered using WordNet (Miller, 1995).

We split the test sets additionally into 3 subcategories: “seen”, where both words in the pair have been seen during training, “half-seen”, where only one word is seen in training, and “unseen”,
Table 8: Model average z-scores for the respective test sets, with a 95% confidence interval.

<table>
<thead>
<tr>
<th></th>
<th>Grammatical</th>
<th>Close Spell</th>
<th>Far Spell</th>
<th>Far Synonym</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subword</td>
<td>µ: 0.01</td>
<td>σ: 0.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seen</td>
<td>1.03 ± 0.08</td>
<td>0.17 ± 0.06</td>
<td>0.05 ± 0.05</td>
<td>0.18 ± 0.05</td>
</tr>
<tr>
<td>Half-seen</td>
<td>-0.09 ± 0.07</td>
<td>-0.09 ± 0.02</td>
<td>-0.12 ± 0.02</td>
<td>-0.12 ± 0.02</td>
</tr>
<tr>
<td>Unseen</td>
<td>-0.12 ± 0.12</td>
<td>-0.10 ± 0.05</td>
<td>-0.10 ± 0.02</td>
<td>-0.06 ± 0.09</td>
</tr>
<tr>
<td>SDD</td>
<td>µ: 0.31</td>
<td>σ: 0.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seen</td>
<td>5.75 ± 0.05</td>
<td>4.68 ± 0.06</td>
<td>-0.52 ± 0.04</td>
<td>-0.49 ± 0.04</td>
</tr>
<tr>
<td>Half-seen</td>
<td>6.09 ± 0.09</td>
<td>5.17 ± 0.06</td>
<td>-0.43 ± 0.04</td>
<td>-0.40 ± 0.04</td>
</tr>
<tr>
<td>Unseen</td>
<td>6.06 ± 0.14</td>
<td>5.82 ± 0.11</td>
<td>-0.27 ± 0.04</td>
<td>-0.21 ± 0.13</td>
</tr>
</tbody>
</table>

where neither is seen in training. The sizes of each split are shown in Table 12 in Appendix A. The results are shown in Table 8.

As expected, the subword model has near 0 similarity for half-seen and unseen words since it has no mechanism for developing their embeddings during training. It does recognize grammatical seen words as similar, but it has a more muted response to seen words with similar spelling. In contrast, SDD has a strong response to both seen and unseen words, as the character models can develop embeddings for unseen words according to character-level patterns. We see that both models appear capable of distinguishing between grammatical pairs and pairs with similar spelling but not necessarily similar meaning, though only in the case where both words are seen for the subword model.

There is a stark difference between the z-scores of the Close Spell and Far Spell sets for the SDD model. As for the Far Synonyms, there is a small difference for the subword model, and none for the SDD model. It is possible that these words are distinguished in later layers of the model.

8 Practicality Discussion

While this paper takes a more theoretical approach, using smaller, domain-specific datasets for training NMT, the practical usage is worth considering. Given the prior work of ByT5, Charformer, and CharacterBERT, the consensus appears that with large, pretrained models using character information, the performance on standard metrics is similar, and the outperformance is on out-of-domain data (Boukkouri et al., 2020) or data with character-level corrupted input (Xue et al., 2022). As such, we expect the same is true for SDD, however the downsampling module used (based on Libovický et al. (2021)) appears to be a limiting factor. We explain this in further detail in Appendix C, where we train and test the models on the larger WMT14 German–English dataset, finding that the performance of the subword model deteriorates when the same two-step decoding method is added.

Conversely, we expect such models to be useful in lower-resource settings. In such an scenarios, we show that the inclusion of character-level information improves performance beyond that of subword models (Appendix D).

Table 9: Training time ratios with respect to the subword model in Table 9. While the SDD model is considerably slower than the subword model in terms of training time, it still performs better than the character model, particularly for encoder-only tasks, suggesting that decoding is the main source of the slower training times.

In terms of training time, we report the training time ratios with respect to the subword model in Table 9. While the SDD model is considerably slower than the subword model in terms of training time, it still performs better than the character model, particularly for encoder-only tasks, suggesting that decoding is the main source of the slower training times.

One potential use of the SDD model that is yet unexplored comes in the form of adapting existing subword models to take character input. Since pre-training is a costly endeavor, and since there are beneficial characteristics of character-level models such as out-of-domain generalization, it may be possible to improve a subword model by adapting it to use the SDD downsampling module rather than its own subword embeddings. Previous work in adaptation has shown success in adapting models to different tasks (Üstün et al., 2020; Pfeiffer et al., 2020) and languages (Bapna et al., 2019; Artetxe et al., 2020).

9 For these experiments, we used separate models trained using the same initial vocabulary as T5-Small, in order to get a larger number of single-token words to evaluate on. This is why there exist words in the vocabulary that are not seen during training. The performance of these models is similar to those reported in Table 7.

10 All experiments were conducted on a single Nvidia V100 GPU.
et al., 2019; Üstün et al., 2021), so a similar approach may be useful here.

9 Conclusion

Previous work has casted doubt on the usefulness of character-level NMT models, due to their lack of improvement over subword models despite using more fine-grained information while being also slower. Downsampling modules added to the character models have previously been proposed but have always come at the cost of accuracy.

We show that it is possible to downsample without sacrificing accuracy, by downsampling based on the lengths of subwords. This novel downsampling outperforms the previous downsampling methods, as well as character and subword models on the majority of language pairs tested.

There are several avenues for future research. While much work has been done on optimizing a vocabulary for a subword model, finding the optimal lengths for subword-delimited downsampling is still an open problem. The most promising may in fact be adapting any of the numerous pretrained subword models to use characters as input. Overall, character-level models show promise that has yet to be fully realized.

10 Limitations

As we mention in Sections 5 and 8, our largest limitation is that the majority of our testing is done on smaller datasets, consisting of roughly 200 thousand sentence pairs per language pair. While we do test on a larger dataset for German→English, it is limited to only a single language pair, and single direction, so it is still an open question whether our method works well in general in higher-resource settings.

Additionally, we only test on 3 language pairs in our main results, all of which have English on the source or target side. It is possible that our method only works well in circumstances where English is present in the language pair, or it only works well where the other language is either German, Arabic, or Turkish. Notably, languages such as Chinese can have characters that hold the same meaning as a word in English, and as such subword tokenizers like SentencePiece may be less useful. Subsequently, SDD may be less useful for these types of languages.

In terms of the tasks tested, we mainly focus on translation, so we can make no claims about the performance on other generative tasks. We test on two discriminative tasks, NLI, and review classification, however both are tested without following the popular pretrain-then-finetune paradigm, making the results difficult to compare to existing work. The scope of these tests is also limited to English only.

In terms of parameters tested, we mainly follow previous work, so it is possible that our method does not perform as well (or possibly performs better) under different parameter settings. To keep our carbon impact minimal, we opted for using VOLT to determine an optimal vocabulary size (which requires no training of NMT models), rather than the standard grid search approach. VOLT does not guarantee an optimal vocabulary size however, and this may have an impact on our results, be it favorably or unfavorably.

Finally, as we note in Appendix B.3, our method is considerably slower than the subword model. However, it is also considerably faster than the character model while also performing better on the majority of the tasks tested. As this work is exploring new territory, it is very likely that our implementation is not as efficient as it could be.

11 Acknowledgements

We thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster.

References


Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated


### A Datasets

The datasets used for translation are shown in Table 10. In summary, we use IWSLT2017, SE-TIMES, and datasets provided by WMT shared tasks.

The datasets used for NLI and review classification (RC) are shown in Table 11. For NLI, we use the SNLI corpus (Bowman et al., 2015), and for RC we use the Amazon Customer Reviews Dataset. For SNLI we use the predefined train/dev/test splits, and for RC we take a 99/1 split of the Books_v1_00 subset for training and validation, and use 1% of the Books_v1_02 subset for testing (amounting to roughly 22k test sentences).

Concerning the evaluation sets we generated for Section 7, we show the sizes of the generated sets in Table 12. We limited the size of the test sets to 2000 word pairs, randomly selecting from those generated.

---

11.[https://sites.google.com/site/iwsltevaluation2017/TED-tasks](https://sites.google.com/site/iwsltevaluation2017/TED-tasks)

12.[https://opus.nlpl.eu/SE-TIMES.php](https://opus.nlpl.eu/SE-TIMES.php)

13.[https://statmt.org/wmt21/translation-task.html](https://statmt.org/wmt21/translation-task.html)

Table 10: Datasets used for translation, showing the names of the training and evaluation sets, the size of the training data (in sentence pairs), and the domain.

<table>
<thead>
<tr>
<th>Task</th>
<th>NLI</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>SNLI train</td>
<td>Books_v1.00</td>
</tr>
<tr>
<td>Size (k)</td>
<td>547</td>
<td>5711</td>
</tr>
<tr>
<td>Eval</td>
<td>SNLI dev/test</td>
<td>Books_v1.00</td>
</tr>
<tr>
<td>Domain</td>
<td>Image captions</td>
<td>Book reviews</td>
</tr>
</tbody>
</table>

Table 11: Datasets used for encoder-only tasks, showing the names of the training and evaluation sets, the size of the training data (in premise/hypothesis pairs or reviews), and the domain.

<table>
<thead>
<tr>
<th>Grammatical</th>
<th>Close spell</th>
<th>Far spell</th>
<th>Far Synonym</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen</td>
<td>1873</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>Half-seen</td>
<td>273</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>Unseen</td>
<td>92</td>
<td>525</td>
<td>2000</td>
</tr>
</tbody>
</table>

Table 12: Size of generated datasets for evaluating the embeddings of word pairs.

Table 13: Number of parameters (in millions) for each model trained.

<table>
<thead>
<tr>
<th>Lang pair</th>
<th>Main</th>
<th>Low-Resource</th>
<th>High-Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>en++de</td>
<td>16</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>en++ar</td>
<td>5</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>en++tr</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>xh++zu</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>de++en</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 14: Vocabulary sizes and max token lengths for the languages tested.

As noted in Section 5, we use VOLT to determine the optimal vocabulary sizes. We specify these, along with the max token length that most closely matches an average downsampling factor of 4 in Table 14. It should be noted that the max token length argument in SentencePiece is determined by the maximum character length, however SDD operates on the byte level. So for English–Arabic, although the max token length is set to 4, the average downsampling factor is in fact above 4, due to the length of the Arabic characters in UTF-8 being typically 2-3 bytes.

B.2 Vocabulary

We use an effective batch size of 128, accumulated over 4 iterations. We use the AdamW optimizer (Loshchilov and Hutter, 2017), with betas 0.9 and 0.999, a learning rate of 2e-4, and a linear scheduler with a warmup of 10000 steps. We apply an early stopping with a patience of 10, using the BLEU score on the validation set as the stopping criterion.

B.3 Training

We use an effective batch size of 128, accumulated over 4 iterations. We use the AdamW optimizer (Loshchilov and Hutter, 2017), with betas 0.9 and 0.999, a learning rate of 2e-4, and a linear scheduler with a warmup of 10000 steps. We apply an early stopping with a patience of 10, using the BLEU score on the validation set as the stopping criterion.

C Higher-Resource Translation

We also run experiments in the higher resource setting of the WMT14 German→English data. Fol-
lowing the principles of Ott et al. (2018), we use larger batch sizes of 50k and 240k tokens for the subword and character models, respectively, both of which average to about 2000 sentences per batch. We also increase the learning rate to 5e-4 and use mixed-precision for the training. We include an additional model, which is the subword model with the two-step decoding as used in the models with downsampling. In other words, the subword model is given the same LSTM upsampling head, but the upsampling is simply 1-to-1. We test this to ablate any effect of the upsampler on the results.¹⁵

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
<th>COMET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subword</td>
<td>28.62</td>
<td>0.431</td>
</tr>
<tr>
<td>Two-step subword</td>
<td>27.49</td>
<td>0.360</td>
</tr>
<tr>
<td>SDD</td>
<td>27.27</td>
<td>0.364</td>
</tr>
</tbody>
</table>

Table 15: Translation results on the WMT14 DE→EN dataset.

Our results are shown in Table 15. We can see that the standard subword model performs best, however the addition of the two-step decoder hurts performance significantly. The subword model with the two-step decoder still performs on par with our SDD model, confirming our expectations. We conclude that SDD is competitive in higher-resource settings, since it achieves similar scores to the subword variant that also has a two-step decoder. This two-step decoder, while effective on lower-resource settings, does not scale well to higher-resource settings without any modification. This raises the question as to whether an upsampling method exists that has better scaling ability. We leave this for future research to explore.

D Lower-Resource Translation

To analyze our models on lower-resource translation, we chose to train and evaluate on Xhosa–Zulu, specifically the data provided by the WMT2021 Shared Task. Typically, in lower-resource scenarios, other techniques such as multilingual transfer learning and back-translation are applied to improve the models. We chose this language pair as the performance gain from employing these techniques is less substantial (Wei et al., 2021).

The results show that the character-level model without downsampling performs best, with SDD as a close second. Xhosa and Zulu, being closely related languages, likely would benefit greatly from character-level translation regardless of the amount of data, given that many of the differences between the two are on the character level. Of course, more thoroughly evaluating SDD on low-resource settings would require a pretrain-then-finetune approach, which we reserve for future research.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
<th>COMET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subword</td>
<td>4.59</td>
<td>5.89</td>
</tr>
<tr>
<td>Char</td>
<td>6.51</td>
<td>6.42</td>
</tr>
<tr>
<td>SDD</td>
<td>6.35</td>
<td>6.32</td>
</tr>
</tbody>
</table>

Table 16: Translation Results for Xhosa–Zulu.

¹⁵We do not include this in our main results because we did not see a noticeable difference in the performances there.