Semitic Root Encoding: Tokenization Based on the Templatic Morphology of Semitic Languages in NMT

Brendan T. Hatch **Stephen D. Richardson**
Brigham Young University
Provo, UT
{hatch5o6, srichardson}@byu.edu

Abstract

The morphological structure of Semitic languages, such as Arabic, is based on non-concatenative roots and templates. This complex word structure used by humans is obscured to neural models that employ traditional tokenization algorithms, such as byte-pair encoding (BPE) (Sennrich et al., 2016; Gage, 1994). In this work, we present and evaluate Semitic Root Encoding (SRE), a tokenization method that represents both concatenative and non-concatenative structures in Semitic words with sequences of root, template stem, and BPE tokens. We apply the method to neural machine translation (NMT) and find that SRE tokenization yields an average increase of 1.15 BLEU over the baseline. SRE tokenization is also robust against generating combinations of roots with template stems that do not occur in nature. Finally, we compare the performance of SRE to tokenization based on non-linguistic root and template structures and tokenization based on stems, providing evidence that NMT models are capable of leveraging tokens based on non-concatenative Semitic morphology.

1 Introduction

1.1 Overview

Byte-pair encoding (BPE) (Sennrich et al., 2016; Gage, 1994) and unigram language modeling (Kudo, 2018) are commonly used approaches for sub-word segmentation in language models. Segmenting words into sub-words with these methods often allows models to learn ***concatenative*** word structures, such as prefixation, suffixation, and compounding, making them especially desirable for modeling languages with rich concatenative morphology. However, since these approaches only segment on continuous strings, they cannot account for the templatic morphology of Semitic languages like Arabic and Hebrew, which is based on ***non-concatenative*** root and template paradigms.

In this work, we present a sub-word segmentation method called Semitic Root Encoding (SRE) which represents word stems with two tokens: a root token and a template stem token.

We evaluate the impact of SRE tokenization on neural machine translation (NMT), assessing general translation quality and examining dubious word stems generated (*i.e.*, stems created by root + template stems combinations that do not occur in nature). We make the following contributions:

1. We show that SRE yields small improvements in general translation quality compared to BPE.

2. We show that models trained with SRE rarely generate dubious root + template stem combinations.

3. We provide further evidence that NMT models can learn Semitic non-concatenative morphology, leveraging root tokens and template stem tokens.

1.2 The Templatic Morphology of Semitic Languages

The morphology of Semitic languages is based on non-concatenative root and template paradigms. The principles of Semitic templatic morphology are explained here with examples from Modern Standard Arabic. While Arabic, like many languages, employs concatenative word structures, it also famously exhibits a non-concatenative root and template schematic to create word stems. Most roots consist of three consonants (though this does vary), known as radicals, which are inserted into various templates to form words. While the data used in this research is in the original Arabic script, throughout this paper, example words will be provided in Latin transliterations where roots will be represented with capital letters and templates will

Proceedings of The Third Arabic Natural Language Processing Conference, pages 26–41

Root	Template	Template function	Word	Gloss
K-T-B	y**123**	*verb*	y**KTB**	*he writes*
S-K-N	y**123**	*verb*	y**SKN**	*he lives/resides (in)*
K-T-B	m**12**u**3**	*passive participle*	m**KT**u**B**	*is written*
S-K-N	m**12**u**3**	*passive participle*	m**SK**u**N**	*is haunted/lived (in)*
K-T-B	**1**a**23**	*active participle*	**K**a**TB**	*writer, is writing*
S-K-N	**1**a**23**	*active participle*	**S**a**KN**	*is living/residing (in), resident*
K-T-B	**12**a**3**	*plural active participle*	**KT**a**B**	*writers (plural of KaTB)*
S-K-N	**12**a**3**	*plural active participle*	**SK**a**N**	*residents, population (plural of SaKN)*

Table 1: Example templates and their functions. Roots in the first column are inserted into templates in the second column to produce words in the fourth column.

be represented with lowercase letters and the numbers 1, 2, and 3 that act as placeholders for the first, second, and third radicals. It should also be noted that short vowels in Arabic, represented with diacritics, are usually omitted, and therefore, only long vowels will be represented in the examples provided. For example, the verb *y**KTB*** (يكتب) consists of the root K-T-B and the template *y**123***, where K is in slot 1, T is in slot 2, and B is in slot 3. Words that share a root are usually closely related semantically. Table 1 shows a few words made with the roots K-T-B and S-K-N and four different templates, and demonstrates that words with the root K-T-B relate to writing while words with root S-K-N relate to residence. Additionally we see that each template connects each root meaning with a grammatical function.

As seen in Table 1, roots are not always a continuous sequence of characters, but are broken up in several different ways depending on the template they are inserted into. Non-continuous portions of templates can also be a single unit that serves a special grammatical function. Because sub-word segmentation methods like BPE and unigram can only represent words as a concatenative series of sub-words, they obscure non-concatenative word structures to translation models, even though the non-concatenative structures are transparent and useful to humans. In attempt to remedy this weakness, the SRE method represents word stems as a root token followed by a template stem token, operating on the hypothesis that this will allow models to make generalizations about root meanings and template functions in ways that are impossible with traditional sub-word segmentation methods.

In this work, the term *stem* will refer to the substring ranging from the first radical of a word to the last radical. The term *template stem* will similarly refer to the substring ranging from the first placeholder of the template to the last placeholder. For example, in the word *alm**KTuB**h* (المكتوبة), the stem is ***KTuB***. In the corresponding template *alm**12u3**h*, the template stem is ***12u3***.

Often, prefixes, suffixes, and clitics are appended to these stems. Additionally, some words, such as those borrowed from other languages, do not have stems with the templatic structure described, but still may have affixes. For these reasons, SRE is designed to account for both the concatenative and non-concatenative/templatic word structures of Semitic languages.

2 Related Works

BPE (Sennrich et al., 2016; Gage, 1994) and unigram language modeling (Kudo, 2018) are common strategies for handling morphological complexity in language models. Toolkits like MADAMIRA (Pasha et al., 2014), Farasa (Abdelali et al., 2016), and CAMeL Tools (Obeid et al., 2020), provide, among other capabilities, Arabic morphological sub-word segmentation functions, a problem also tackled by Almuhareb et al. (2019), who propose a bi-directional long short-term memory system. Chaudhary et al. (2018) train named entity recognition (NER) and machine translation (MT) systems on both morphemic and phonemic sub-words of various languages; Alkaoud and Syed (2020) train traditional and contextual Arabic word embedding models on morphemic sub-words; and Guzmán et al. (2016) use embeddings of Arabic lexical and morpho-syntactic units in the evaluation of MT. Shapiro and Duh (2018) create Arabic word embeddings that capture the whole word as well as the lemma, and Salama et al. (2018) train Arabic lemma-based embeddings as well as whole word embeddings that incorporated morphological

annotations. Additionally, Alyafeai et al. (2023) compare six tokenizing strategies on four Arabic text classification datasets, revealing that the best approach is task-dependent.

Semitic root extraction has been addressed in various works. De Roeck and Al-Fares (2000) propose a clustering algorithm, Taghva et al. (2005) a rule-based system, Sakakini et al. (2017) an unsupervised learning method, and El-Kishky et al. (2019) a constrained seq2seq model.

Few works, however, fully tackle challenges of non-concatenative morphology on language generation tasks, and traditional sub-word segmentation methods may not be optimal for it. Amrhein and Sennrich (2021), for instance, though not addressing Semitic root and template morphology, demonstrate that BPE underperforms for other kinds of non-concatenative morphology like vowel harmony. That said, El-Kishky et al. (2019), like we do, present a sub-word segmentation approach to represent the non-concatenative word structure of Arabic, though it *only* segments non-concatenative structures and also does not limit the total vocabulary size. Their work also differs in the tasks they apply the scheme to, being word analogy, word similarity, and LSTM language modeling. In this work, we present SRE, which represents both concatenative and non-concatenative word structures in Arabic textual data while controlling for vocabulary size, and evaluate it as applied to NMT.

3 Sub-word Segmentation Methods

In this section, we describe all sub-word segmentation approaches employed in our experiments, which include SRE, BPE, Fake-SRE, and Stem-SRE.

3.1 SRE

SRE sub-word segmentation accounts for both the *non-concatenative* and *concatenative* morphology in each word. The first step to accomplish this task is *SRE Preprocessing*, a method for converting non-concatenative Semitic structures into a concatenative representation.

SRE Preprocessing. *SRE Preprocessing* requires a morphological analyzer to extract the root and template from a given word. We use the morphological analyzer[1] provided in the CAMeL Tools toolkit (Obeid et al., 2020)[2], using the *calima-msa-r13* database. *SRE Preprocessing* for a sentence works as follows: The sentence is first split into words using the CAMeL Tools word tokenizer[3]. For each word in the sentence, the root and template are extracted using the morphological analyzer. The word is then reformatted to be a string consisting of the root wrapped in angle brackets, followed by the template. For example, the word *alm**KTuBh*** (المكتوبة) would be reformatted to the string *'<KTB>alm12u3h'*. If the morphological analyzer detects no Semitic root or template, then the word is left as is in the reformatting process. Afterwards, the reformatted words are concatenated into a complete preprocessed sentence. See Figure 1 for an example of *SRE Preprocessing*.

SRE Preprocessing is then used in two separate pipelines: (1) Training a special BPE model called SRE BPE and (2) SRE sub-word segmentation itself.

Training SRE BPE. SRE BPE is a special SentencePiece (Kudo and Richardson, 2018)[4] BPE model trained on a dataset of *SRE Preprocessed* sentences (see Section 3.2 for more details on the BPE implementation). Prior to training this BPE model, a cache of roots and templates, called *RootCache*, was created by running the morphological analyzer on a large dataset that included training, validation, and test data (discussed in Appendix A.1). All roots, wrapped in angle brackets (*e.g.*, '<KTB>'), and template stems (*e.g.*, '12u3') from *RootCache* are provided as *user_defined_symbols* to the SentencePiece module. For vocabulary items provided as *user_defined_symbols*, the SentencePiece module always extracts these as one piece.

SRE Sub-word Segmentation. To segment a sentence into sub-words, *SRE Preprocessing* is applied first, after which the sentence is segmented with the SRE BPE model just described. See Figure 2 for an example of SRE Sub-word Segmentation.

SRE Sub-word "De-segmentation". To reverse the sub-word segmentation on model hypotheses, each output sequence is first detokenized with the SRE BPE model. Afterwards, the segment is split into words. For each word in the sequence, each radical of the root wrapped in angle brack-

[1] https://camel-tools.readthedocs.io/en/latest/api/morphology/analyzer.html

[2] https://camel-tools.readthedocs.io/en/latest/

[3] https://camel-tools.readthedocs.io/en/latest/api/tokenizers/word.html

[4] https://github.com/google/sentencepiece

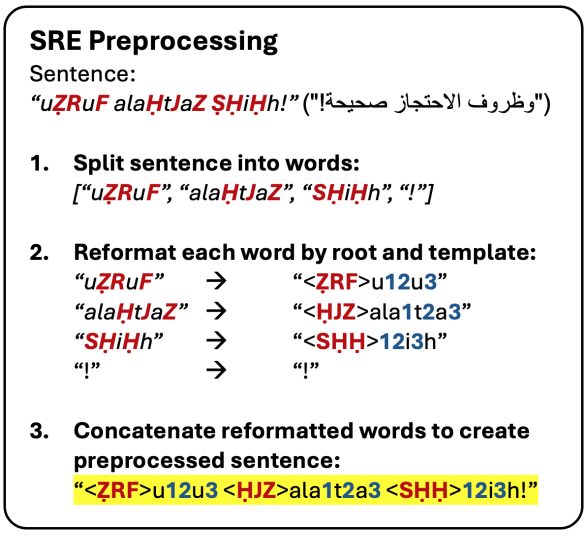

Figure 1: *SRE Preprocessing* example. Radicals are bold and red. Template placeholders are bold and blue. The final *SRE Preprocessed* sentence is highlighted in yellow.

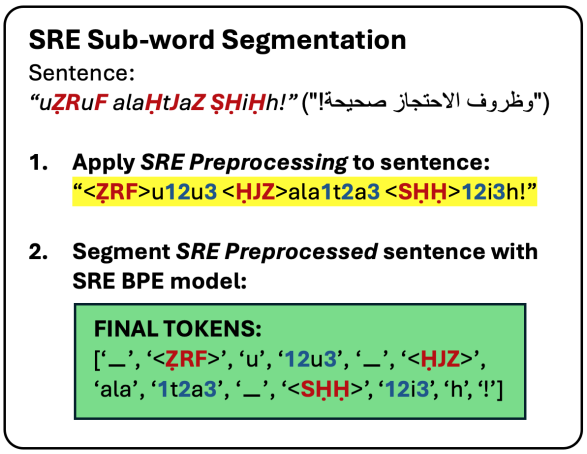

Figure 2: *SRE Sub-word Segmentation* example. Radicals are bold and red. Template placeholders are bold and blue. The *SRE Preprocessed* sentence is highlighted in yellow. Final tokens are in the green box.

ets (if one exists) is inserted into its corresponding placeholder in the template to create the reconstructed word. The reconstructed words are then concatenated to create the final output. Figure 3 provides an example of this "de-segmentation" process.

While SRE, due to the complexity of the morphological analyzer and SRE Preprocessing, is computationally slower than BPE, it more accurately represents the non-concatenative components of Semitic words in ways impossible to other tokenizers.

We created two SRE sub-word segmentation models, *SRE-8k* and *SRE-20k*. Both had 3,956 root tokens and 305 template stem tokens, which were retrieved from *RootCache*. The SentencePiece-based SRE BPE models inside *SRE-8k* and *SRE-20k* were both trained on 500,480 Arabic sentences, with vocabulary sizes set to 8,000 and 20,000, respectively, which included unknown, beginning-of-sequence, and end-of-sequence tokens by default. We then added a pad token, making the final vocabulary sizes 8,001 and 20,001.

3.2 BPE

We use the following implementation for the BPE models described later in this section as well as the SRE BPE models wrapped inside all versions of SRE (see Sections 3.1, 3.3, 3.4, and Appendices F and G).

We use the SentencePiece implementation of

SENTENCE: *"w'**RSL**t **RS**a'**L**"* (وأرسلت رسائل)
GLOSS: *"And she sent messages"*

Method	Preprocessing
SRE	*"<**RSL**>w'123t <**RSL**>12a'3"*
Fake-SRE	*"<**'LT**>w1rs23 <**SA'**>r123l"*

Table 2: *SRE Preprocessing* compared to *Fake-SRE Preprocessing*. In the sentence at the top, the true roots are represented with bold capital letters. SRE extracts the true roots; however, Fake-SRE does not, and therefore, the different sets of letters it selects as "roots" are shown in bold capital letters in the second row. The apostrophe (') is used as transliteration for letters أ and ئ.

BPE with 1.0 character coverage. As mentioned in Section 3.1, the SentencePiece module will always extract vocabulary items added to *user_defined_symbols* as one piece. We added the character '▁', which SentencePiece uses to represent whitespace, to *user_defined_symbols*, therefore compelling segmentation on whitespace in all BPE and SRE tokenizers in this work.

Further details of SRE BPE models are described as needed in their respective sections.

As for BPE models, we created the following: two English with vocab sizes of 8,001 and 20,001, *BPE-en-8k* and *BPE-en-20k*; and two Arabic of the same sizes, *BPE-ar-8k* and *BPE-ar-20k*. These four models were each trained on 500,480 sentences that had ***not*** undergone *SRE Preprocessing*.

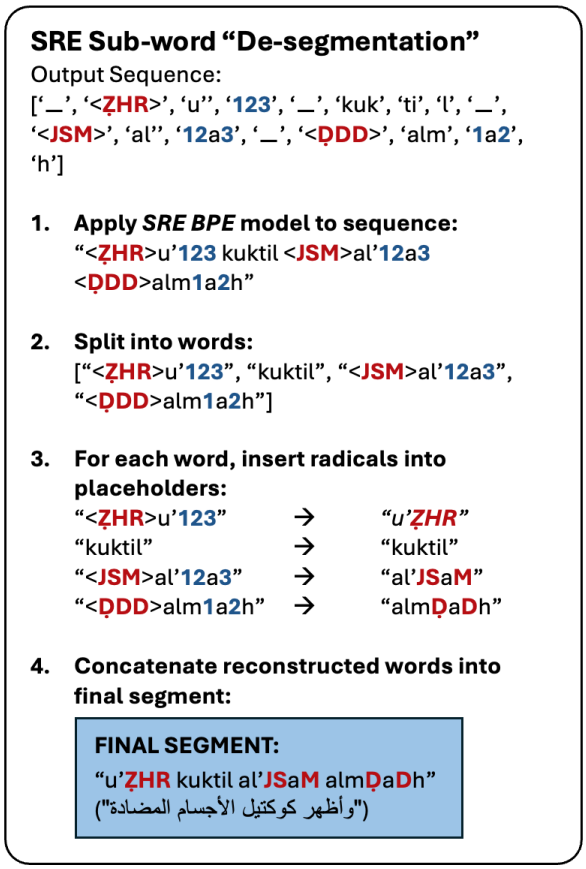

Figure 3: *SRE Sub-word "De-segmentation"* example. Radicals are bold and red. Template placeholders are bold and blue. Final postprocessed segment is in the blue box.

3.3 Fake-SRE

To confirm that the NMT models make meaningful generalizations of root and template stem tokens, we designed two variations of SRE to serve as quasi-ablations, the first being Fake-SRE. In Fake-SRE, sets of non-continuous characters in each word are selected to be the "root" and the "template stem", even though they generally are not the real linguistic root and template stem. The intuition behind this is that if non-linguistic root and template stem tokens are presented to the model, then the model will be compelled to rely on non-linguistic patterns and memorization to learn word forms. If a model performs better with tokenization based on the real linguistic root and template tokens than with tokenization based on the false ones, then it suggests it is indeed leveraging the non-concatenative linguistic patterns rather than simply memorizing word forms.

To accomplish this, we created *FakeRootCache*, which associates each word in the data with a non-lingustic "root" and "template stem". We describe its creation in Appendix D. The SRE method from Section 3.1 is then applied, but using instead the false root and template parses in *FakeRootCache*. We show an example of how *SRE* and *Fake-SRE Preprocessing* compare in Table 2, demonstrating that SRE can represent the semantic relationship between the words *w'**RSL**t* (*and she sent*) and ***RSa'L*** (*messages*) with the root token <RSL>, whereas Fake-SRE cannot, since it selects different letters to serve as roots.

We created the tokenizer *Fake-SRE-20k*, which contained 14,282 root tokens and 2,413 template stem tokens. Because so many tokens were needed for roots and template stems, we created it with total vocabulary size of 20,001. The results of using Fake-SRE compared to SRE are discussed in Section 4.3 below.

3.4 Stem-SRE

The second quasi-ablation is conducted with Stem-SRE, where rather than performing segmentation on roots and template stems, segmentation is performed on whole stems, which again are the continuous subsequences extending from the first radical to the last radical. In short, instead of representing each stem as two tokens, a root and a template stem, each stem is represented by a single token. The BPE algorithm then determines prefixes and suffixes. The reasoning behind this quasi-ablation is if NMT performs better with SRE than with Stem-SRE, it suggests that NMT models are indeed able to leverage the knowledge encoded in the non-concatenative morphemes (*i.e.*, the root and template stem). We describe the details of Stem-SRE in Appendix E.

We created one of these tokenizers, called *Stem-SRE-20k*. This model contains 10,984 stem tokens, and for the sake of comparability with *Fake-SRE-20k*, has a total vocabulary size of 20,001. The results of using Stem-SRE compared to SRE are discussed in Section 4.3 below.

3.5 Additional Sub-word Segmentation Methods

Appendix F addresses SRE-MF, where SRE is applied to only the least frequent word forms. Appendix G addresses In-Situ-SRE, where we experimented with an alternative token order.

4 Experiments and Results

All NMT models in this work use the architecture of *BartForConditionalGeneration* (Lewis et al.,

2020)[5], available from the *transformers*[6] Python library. We set the number of encoder and decoder layers each to 6, and the number of encoder and decoder attention heads each to 8. The max length for generation was set to 1,024. All other architectural configurations were kept at their default values. All models were trained to convergence, early stopping with a patience of 10.

We use four divisions of our training data in our experiments, each containing 10M sentence pairs with no overlap, referred to as the Trial 1, Trial 2, Trial 3, and Trial 4 versions of the training set. We validate on 997 sentences, and evaluate general translation quality on a test set of 1,009 sentences with BLEU (Papineni et al., 2002) and chrF (Popović, 2015), calculated with SacreBLEU (Post, 2018)[7]. The creation of our datasets and sources are described in detail in Appendix A.

4.1 General Translation Quality

To assess whether tokenization with SRE yields improvements in overall translation quality, two English-to-Arabic NMT models were trained, *en2ar-SRE* and *en2ar-BPE*, which differ in the tokenization methods used on the source and target data. These were trained with a batch size of 512, validating on intervals of 625 batches, and applying a linear warm-up for 10,240 steps with a max learning rate of 2e-5. The model initialization and data loader were seeded with 0, as is the case in all experiments in this work.

en2ar-SRE was trained tokenizing the English source sentences with *BPE-en-8k* and the Arabic target sentences with *SRE-8k*.

en2ar-BPE was trained tokenizing the English source sentences with *BPE-en-8k* and the Arabic target sentences with *BPE-ar-8k*.

BLEU and chrF scores over 4 trials are reported in Table 3. Each trial used a separate version of the training set, though using the same validation and test set. Across all trials, *en2ar-SRE* has greater scores than *en2ar-BPE*, with an average lead of 1.15 BLEU. Paired approximate randomization (Riezler and Maxwell, 2005) was calculated with SacreBLEU, revealing that the *en2ar-SRE* BLEU scores were significantly different in three of the four trials. These results suggest a small improvement in translation quality as a result of using SRE tokenization.

To corroborate this finding, we conducted a human evaluation of these models. Three native Arabic speakers, referred to as Evaluators 1, 2, and 3, examined the same set of 100 random source sentences of the test set and the translations from Trial 1 of *en2ar-SRE* and *en2ar-BPE*. For each sentence, they had access to both the source sentence and reference translation, and were presented the *en2ar-SRE* and *en2ar-BPE* hypotheses in a random order. They then scored the better hypothesis with a score of 1, and the worse with a score of 0. If they thought the two hypotheses were equal in quality, they could give 0s to both or 1s to both. The sums of the scores (in essence, the number of translations out of 100 sentences with a score of 1) for each system from each evaluator are reported in Table 4, along with the number of times each system generated a translation with a score that was better and the same as the other system.

Evaluators 1 and 2, who both teach Arabic as a second language, prefer *en2ar-SRE* with "Better" margins of 9 and 21, respectively. Evaluator 2 is also more discriminating, giving tying scores far less often than Evaluator 1 and rating 41 *en2ar-SRE* translations as better, whereas Evaluator 2 only rates 16 as better. However, they ultimately agree in their preference for translations generated by a system trained with SRE tokenization. On the other hand, Evaluator 3, who is a graduate student in linguistics, shows a slight preference for *en2ar-BPE*, though with less significant margin of 3. Given the years of experience of Evaluators 1 and 2 as Arabic language educators, more confidence should be placed in their scores as they are likely more alert to subtle differences between translations. It is therefore reasonable to conclude that tokenizing with SRE leads to a small increase in translation quality.

We conducted a single trial of similar experiments in low-resource scenarios, described in Appendix C, where translation models trained with SRE do not hold a lead according to automated metrics over those trained with BPE. It may be that a significantly greater number of roots and template stems are needed to provide benefit to translation quality.

SRE represents a sentence with more sub-words than BPE, which only represents infrequent words as a series of sub-words. We considered whether

[5] https://huggingface.co/docs/transformers/en/model_doc/bart - Again, we use ONLY the architecture and NOT the pretrained weights.

[6] https://huggingface.co/docs/transformers/en/index

[7] https://github.com/mjpost/sacrebleu

Model	Trial 1		Trial 2		Trial 3		Trial 4		Avg.	
	BLEU	chrF	BLEU	chrF	BLEU	chrF	BLEU	chrF	BLEU	chrF
en2ar-BPE	26.93	58.86	27.23	58.80	25.76	58.58	25.64	57.14	26.39	58.35
en2ar-SRE	**27.92***	**58.96**	**28.02**	**59.39***	**27.50***	**58.72**	**26.72***	**58.03***	**27.54**	**58.78**

Table 3: BLEU and chrF scores for *en2ar-BPE* and *en2ar-SRE* over 4 trials. * indicates that the *en2ar-SRE* score is statistically significantly different than the baseline *en2ar-BPE* score with a p-value < 0.05.

Model	Evaluator 1			Evaluator 2			Evaluator 3			Avg.		
	Sum	Bet	Tie	Sum	Bet	Tie	Sum	Bet	Tie	Sum	Bet	Tie
en2ar-BPE	23	7	77	40	20	39	**84**	**10**	83	49	12.33	66.33
en2ar-SRE	**32**	**16**	77	**61**	**41**	39	81	7	83	**58**	**21.33**	66.33

Table 4: Human rank scores for the Trial 1 translations of 100 sentences. **Sum** represents the number of the system's translations scored with 1. **Bet** (Better) represents the number of times the system's translations scored 1 when the other system's scored 0. **Tie** represents the number of times the system's translations score (0 or 1) was the same as the other system's.

this complicates that translation task for NMT models and conducted an experiment using a variation of SRE (SRE-MF, described in Appendix F) that keeps the most frequent words as single tokens, rather than as series of sub-words. We found this made insignificant impact on BLEU. While not segmenting frequent word forms into sub-words arguably simplifies the task, allowing the model to generalize about their meanings with 1 embedding rather than 2 or more, the segmentation of these frequent word forms provides more instances of roots and template stems which may allow the model to make better representations of less frequent word forms where transparency into the morphological components may be helpful. Possible benefits of segmenting versus not segmenting frequent word forms may be competing with each other, and hence, similar scores result from the tradeoff, though this would need to be investigated further.

4.2 Dubious Word Stems

While a given root may be inserted into many templates, not all roots can be inserted into all templates and form valid words. We wanted to ensure that NMT models trained with SRE were not generating dubious word stems by generating an invalid combination of a root and template stem. We therefore ran the four trials of the *en2ar-SRE* and *en2ar-BPE* translation models from Section 4.1 on the test set as well as an *extra* test set of 9,669 sentences (described in Appendix A.2), since more generated sentences will better tell us how robust an NMT model is against generating dubious word stems. For each generated sentence, the sentence was split into words using the CAMeL Tools word tokenizer. We then checked if each word existed in an *Arabic dictionary* (described in Appendix A.3), distinguishing between "Arabic words", which contain at least one Arabic character, "and non-Arabic words". This distinction is important because many out-of-dictionary words are are written in Latin letters, like some proper nouns. Table 10 of Appendix H.1 reports the raw number of Arabic out-of-dictionary words and non-Arabic out-of-dictionary words generated. We observed no patterns between the number of out-of-dictionary words generated by *en2ar-SRE* and *en2ar-BPE*.

We examined a portion of the out-of-dictionary words generated for the test set by *en2ar-SRE* and noticed that many of them were transliterations of proper nouns, whether done well or not, and likely did not contain a Semitic root. We ran *SRE Preprocessing* on all of the Arabic out-of-dictionary words generated for the test set by Trial 1 of *en2ar-SRE*, and noticed that out of 83, only 4 have a Semitic root. We manually reviewed these 4 with Evaluator 1 and discovered that all are actually valid word forms that happen to not be in the *Arabic dictionary*. We repeated this process for the hypotheses on the *extra* test set. Of 264 Arabic out-of-dictionary words, 36 have a Semitic root. Of the 36, 30 are valid words, 4 have valid stems with invalid affixes, and 2 have dubious stems.

We conducted this evaluation with Evaluator 1 again on the *en2ar-SRE* Trial 2 hypotheses of the *extra* test set, in which, of 30 Arabic out-of-dictionary words with Semitic roots, 3 are invalid

Model	BLEU	chrF
en2ar-SRE-20k	27.62	59.12
en2ar-BPE-20k	**28.03**	**59.42**
en2ar-Fake-SRE-20k	24.12*	56.69*
en2ar-Stem-SRE-20k	25.67*	58.06*

Table 5: BLEU and chrF scores for *en2ar-SRE-20k*, *en2ar-BPE-20k*, *en2ar-Fake-SRE-20k*, and *en2ar-Stem-SRE-20k*. * indicates that the scores are statistically significantly different than those of *en2ar-SRE-20k*.

words due to invalid affixes, and ***none*** are invalid due to dubious stems. All of the counts can be seen in Table 11 of Appendix H.2.

We conclude that NMT models trained with SRE rarely generate invalid root + template stem combinations.

4.3 Fake-SRE and Stem-SRE

In this section, we present two quasi-ablations to answer the following questions: (1) Can we confirm that NMT models generalize about root and template stem meanings, or do they just memorize word pieces? (2) Is there benefit for an NMT model to see both root and template stem, or would segmentation based on stems (without decomposing them into roots and template stems) perform just as well or better? To find out, we compare SRE, BPE, Fake-SRE, and Stem-SRE. Because the Fake-SRE and Stem-SRE tokenizers were created with vocabularies of 20,001, we used versions of the SRE and BPE tokenizers of the same size for the sake of comparability.

When training all the following NMT models, English source sentences were tokenized with *BPE-en-20k*, while Arabic target sentences were tokenized as follows: The *SRE-20k* tokenizer was used for *en2ar-SRE-20k*, *BPE-ar-20k* was used for *en2ar-BPE-20k*, *Fake-SRE-20k* was used for *en2ar-Fake-SRE-20k*, and *Stem-SRE-20k* was used for *en2ar-Stem-SRE-20k*.

These models were trained on the Trial 1 training set with the same hyperaparameters and configurations as *en2ar-SRE* and *en2ar-BPE*, besides tokenizers and vocabulary size. Table 5 reports the BLEU and chrF scores.

We observe that we cannot perform random root and template stem tokenization and get the same performance, demonstrated by *en2ar-Fake-SRE-20k*, which was trained on tokens based on non-linguistic root and template stems, and which scores more than 3 BLEU less than the model trained with SRE tokenization, *en2ar-SRE-20k*. Tokenization based on linguistic stems using *en2ar-Stem-SRE-20k* also yields worse translations than tokenization based on stems decomposed into roots and templates using *en2ar-SRE-20k*. This suggests there is benefit for NMT models to embed the root and template stems separately and generalize about the meanings and functions of each.

We note as well that in this scenario with larger vocabularies, that the gap between BPE and SRE performances observed in Section 4.1 is closed. The apparent performance gain for increasing the vocabulary size for BPE-based NMT models does not seem to apply to SRE-based NMT models. We suspect that this might be because the number of root and template stem tokens, which are components of most words, in the SRE models is fixed, regardless of the total vocabulary size. The additional tokens in a SRE model with a larger vocabulary may be affecting mainly the handful of words that do not have Semitic roots. Future work would need to determine if this is indeed a flaw of SRE and if it can be remedied, perhaps by adjusting the number of root and template stem tokens.

5 Conclusions

BLEU and chrF scores of translation models trained with SRE tokenization on average have a lead of 1.15 BLEU over those trained with BPE, indicating that SRE tokenization yields better translations, a claim supported by the human evaluators, who tend to prefer the outputs of the model trained with SRE tokenization. This gap in performance, however, is closed when vocabulary sizes are increased.

Of the Arabic out-of-dictionary words with Semitic roots generated by SRE translation models, manual review revealed that most were actually valid word forms. In 9,669 sentences generated by a model trained with SRE, only 2 words were composed of a dubious root + template stem combination in Trial 1, and 0 in Trial 2. This indicates that the SRE method only rarely generates dubious word stems.

Additionally, tokenization based on false roots and template stems performs worse than models trained with SRE, suggesting there is value in using morphologically-based tokenization schemes over more random templatic schemes. Tokenization based on whole stems also does not perform as well as tokenization schemes that decompose the

stem into a root and template stem, indicating that NMT models are indeed able to learn and leverage knowledge from Semitic templatic morphology.

6 Future Work

Future work can corroborate these findings with other Semitic languages, perhaps employing unsupervised approaches in the root and template extraction. Its impact on translation into English or between Semitic languages, as well as on other downstream NLP tasks, are other avenues to explore.

Additional directions may also explore the impact of changing SRE vocabulary sizes on translation performance, experimenting both with the number of root and template stem tokens as well as the number of tokens determined by the BPE algorithm.

Future work should also investigate whether there are indeed competing benefits to segmenting versus not segmenting frequent word forms, and if so, how to optimize the tradeoff.

More comparisons of SRE to BPE would also be valuable. This includes evaluations on speed which will provide important baselines for developing SRE optimizations. This also includes more detailed qualitative comparisons of the word forms generated by SRE and BPE-based NMT models and their affects on human comprehension and translation adequacy.

Finally, we know SRE-based NMT models rarely generate dubious word stems, but whether they are able to hypothesize valid word stems, *i.e.* valid root + template stem combinations, that were not seen in the training data is to be determined.

Limitations

Templates in Arabic include diacritics written below and above letters, most of which indicate short vowels. In the greater part of most documents, these diacritics are omitted. Without diacritics, many surface forms can represent multiple utterances, though readers of Arabic are almost always able to disambiguate contextually. When clarity may be needed, writers may include diacritics, but the usage is inconsistent. Naturally, this means that a single surface template in writing may refer to many underlying templates used in speech, meaning that the ideal NMT model would associate each surface template with all functions of the underlying templates that it represents. For simplicity, and to maximize generalization of surface templates, we opted in this work to remove all written diacritics. However, an NMT model that uses SRE tokenization in a production environment will need to anticipate inputs that include diacritics, so SRE should be developed to handle them.

To support the claim about the impact of SRE on translation quality, we conducted a human evaluation. Though all evaluators were native speakers of Arabic and knew some English, they were mainly volunteers with some variance in their backgrounds. It is hard to say the impact that has on their evaluations, but we reasonably posit that the two evaluators who teach Arabic as a second language are better evaluators than the one who is a graduate student. Additionally, because none of the evaluators are experts in translation specifically, we opted for a simple ranking evaluation as opposed to an in-depth MQM [8] evaluation which would provide a more detailed and qualitative examination of the translations.

In this work, we evaluated many variations of SRE. Because of time and resource constraints, we opted to only train models that translate into Arabic, but the impact of SRE on translation from Arabic should be evaluated in the future as well.

Acknowledgments

We thank Ammon Shurtz for the helpful feedback he offered on various occasions, as well as members of the BYU MATRIX Lab, who developed the parallel data cleaning pipeline (Appendix B.1). We also express appreciation to Taoufik Ouzine for his consultations as well as the other native Arabic-speaking evaluators for their vital contribution.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 11–16.

Mohamed Alkaoud and Mairaj Syed. 2020. On the importance of tokenization in arabic embedding models. In *Proceedings of the fifth Arabic natural language processing workshop*, pages 119–129.

Abdulrahman Almuhareb, Waleed Alsanie, and Abdulmohsen Al-Thubaity. 2019. Arabic word segmentation with long short-term memory neural networks and word embedding. *IEEE Access*, 7:12879–12887.

[8] https://themqm.org/

Manel Aloui, Hasna Chouikhi, Ghaith Chaabane, Haithem Kchaou, and Chehir Dhaouadi. 2024. 101 billion arabic words dataset. *Preprint*, arXiv:2405.01590.

Zaid Alyafeai, Maged S Al-shaibani, Mustafa Ghaleb, and Irfan Ahmad. 2023. Evaluating various tokenizers for arabic text classification. *Neural Processing Letters*, 55(3):2911–2933.

Chantal Amrhein and Rico Sennrich. 2021. How suitable are subword segmentation strategies for translating non-concatenative morphology? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 689–705, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen, and Jaime Carbonell. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3285–3295, Brussels, Belgium. Association for Computational Linguistics.

Anne N. De Roeck and Waleed Al-Fares. 2000. A morphologically sensitive clustering algorithm for identifying Arabic roots. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 199–206, Hong Kong. Association for Computational Linguistics.

Ahmed El-Kishky, Xingyu Fu, Aseel Addawood, Nahil Sobh, Clare Voss, and Jiawei Han. 2019. Constrained sequence-to-sequence Semitic root extraction for enriching word embeddings. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 88–96, Florence, Italy. Association for Computational Linguistics.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Francisco Guzmán, Houda Bouamor, Ramy Baly, and Nizar Habash. 2016. Machine translation evaluation for Arabic using morphologically-enriched embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1398–1408, Osaka, Japan. The COLING 2016 Organizing Committee.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. Two new evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1094–1101, Reykjavik, Iceland. European Language Resources Association (ELRA).

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the*

Tenth Workshop on Statistical Machine Translation, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan. Association for Computational Linguistics.

Tarek Sakakini, Suma Bhat, and Pramod Viswanath. 2017. Fixing the infix: Unsupervised discovery of root-and-pattern morphology. *arXiv preprint arXiv:1702.02211*.

Rana Aref Salama, Abdou Youssef, and Aly Fahmy. 2018. Morphological word embedding for arabic. *Procedia computer science*, 142:83–93.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. 2020. Ccmatrix: Mining billions of high-quality parallel sentences on the web. *Preprint*, arXiv:1911.04944.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Pamela Shapiro and Kevin Duh. 2018. Morphological word embeddings for Arabic neural machine translation in low-resource settings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 1–11, New Orleans. Association for Computational Linguistics.

Kazem Taghva, Rania Elkhoury, and Jeffrey Coombs. 2005. Arabic stemming without a root dictionary. In *International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, volume 1, pages 152–157. IEEE.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, and 20 others. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

A Data

A.1 Standard Data

This section addresses the training, validation, and test sets. Each of the 4 versions of the training set consists of 10 million English-Arabic parallel sentences retrieved from the CCMatrix parallel corpus (Schwenk et al., 2020; Fan et al., 2021) available on Opus (Tiedemann, 2012)[9]. The English-Modern Standard Arabic portions of the FLORES-200 (Team et al., 2022; Goyal et al., 2022; Guzmán et al., 2019) dev and devtest sets were used respectively for validation and test sets. An extensive parallel data cleaning pipeline was applied to the CCMatrix training data. Additionally, all Arabic diacritics were removed from the training, validation, and test sets. While diacritics in Arabic, including short vowels, are components of a word's template, they are usually omitted in writing since most of the information they convey is gleaned from context. Since their usage is inconsistent, for simplicity, we decided to remove all of them for these experiments. Details of data cleaning, diacritic removal, and additional preprocessing are described in detail in Appendix B.

A.2 *Extra* Test Set

We felt that the 1,009 sentence pairs from the test set were too few to get a good picture of how often NMT models generate dubious word stems (see Section 4.2). We therefore retrieved 9,669 CCMatrix sentence pairs not included in any of the 4 versions of the training data to serve as additional testing data for this purpose. These data were cleaned in the same manner as the CCMatrix training data and are referred to as the *extra* test set. The *extra* test set was never used to evaluate general translation quality with BLEU and chrF metrics.

A.3 Arabic Dictionary

To determine whether translation models trained with SRE tokenization generate dubious word stems, a dictionary of Arabic words is needed. This dictionary was created by downloading a portion of the 101 Billion Arabic Words Dataset (Aloui

[9] https://opus.nlpl.eu/

et al., 2024)[10], and splitting the text on white space and then removing punctuation[11] from each word. Words that contained a numeral or a Latin letter were not included. The final unique set of these words serve as the *Arabic dictionary*, which contains ~5 million unique word forms.

B Data Cleaning and Preprocessing

B.1 Data cleaning

To clean the CCMatrix training data, a parallel data cleaning pipeline was applied. This pipeline follows the guidelines of the GILT Leaders Forum's Best Practices in Translation Memory Management[12], and performs the following steps:

1. Remove pairs containing empty source or target segments.
2. Remove pairs when the source segment exactly or nearly matches the target segment.
3. Remove duplicate source-target pairs.
4. Remove pairs with segments containing mostly non-alphabetic characters.
5. Remove pairs with segments containing abnormally long sequences of characters without spaces, including segments that are only URLs.
6. Remove pairs containing segments with unbalanced brackets.
7. Remove pairs containing fewer than 3 words in the English source segment.
8. Remove pairs with segments containing a higher number of characters than 5 standard deviations above the mean for that language (sentences that are too long).
9. Remove pairs in which the ratio of the lengths of the source and target segments exceeds a certain cutoff.
10. Normalize escaped Unicode characters.
11. Validate and normalize character encodings for each language.
12. Normalize whitespace
13. Shorten sequences of excessively repeated punctuation.
14. Normalize quotation marks.
15. Normalize HTML entities.
16. Remove all markup tags.

[10] https://huggingface.co/datasets/ClusterlabAi/101_billion_arabic_words_dataset

[11] This was done by replacing punctuation characters with whitespaces and then normalizing all series of whitespace to a single space and then removing trailing and leading whitespace.

[12] https://github.com/GILT-Forum/TM-Mgmt-Best-Practices/blob/master/best-practices.md

Hyperparam.	50K	100K	300K	500K
Val. interval	97	195	585	625
Warm-up	97	195	585	976

Table 6: Hyperparameters (number of training steps between validations and the number of warm-up steps) that are different than those described in Section 4.1. The columns correspond to translation models trained with SRE and BPE tokenization on a training set of the indicated size.

B.2 Diacritic Removal and Other Preprocessing

Arabic diacritics in the Arabic portions of the FLORES-200 and cleaned CCMatrix data were removed using the CAMeL Tools toolkit.

A few sentence pairs were removed from the FLORES-200 data and the Trial 1 version of the CCMatrix training data because they were invalid with an implementation we created of the Semitic root-based sub-word segmentation scheme proposed by El-Kishky et al. (2019), which we had originally planned to explore further, but eventually opted not to for the sake of constraining this work. Instances of these pairs were few and removal of them does not impact the conclusions of this paper.

C Low-Resource Experiments

We conducted the experiments similar to those described in Section 4.1 using *BPE-en-8k*, *BPE-ar-8k*, and *SRE-8k* tokenizers, but in low-resource scenarios. NMT models were trained on subsets of the Trial 1 version of the training set sized at 500K, 300K, 100K, and 50K. The resulting low-resource translation models (of each training set size) are described as follows:

-en2ar-SRE models were trained tokenizing the English source sentences with *BPE-en-8k* and the Arabic target sentences with *SRE-8k*.

-en2ar-BPE models were trained tokenizing the English source sentences with *BPE-en-8k* and the Arabic target sentences with *BPE-ar-8k*.

We used the same tokenizers as those mentioned in Section 4.1. We also trained these models with same hyperparameters except for the ones mentioned in Table 6. We refer to these models as *50k-en2ar-SRE*, *50k-en2ar-BPE*, *100k-en2ar-SRE*, etc., and report BLEU and chrF scores for one trial in Table 7. In low-resource scenarios, SRE does not hold a lead over BPE, although the differences may not be significant.

Model	BLEU	chrF
50k-en2ar-SRE	2.33	**28.35**
50k-en2ar-BPE	**2.42**	26.97
100k-en2ar-SRE	5.56	**34.51**
100k-en2ar-BPE	**5.80**	34.37
300k-en2ar-SRE	11.77	41.57
300k-en2ar-BPE	**13.09**	**42.41**
500k-en2ar-SRE	**14.72**	44.94
500k-en2ar-BPE	14.65	**45.29**

Table 7: BLEU and chrF scores for low-resource translation models trained with SRE and BPE.

Model	BLEU	chrF
en2ar-SRE	**27.92**	58.96
en2ar-BPE	26.93*	58.86
en2ar-SRE-MF-3.4k	27.33	**59.61***
en2ar-SRE-MF-2.4k	27.84	59.07

Table 8: BLEU and chrF scores for *en2ar-SRE*, *en2ar-BPE*, *en2ar-SRE-MF-3.4k*, and *en2ar-SRE-MF-2.4k*. Note that the scores for *en2ar-BPE* and *en2ar-SRE* are from Trial 1 and also appear in Table 3. * indicates that the scores are statistically significantly different than those of *en2ar-SRE*.

D *FakeRootCache*

To create *FakeRootCache*, 10,000 sentences were retrieved from the training set. For each unique word in the 10,000 sentences, every series of 3 letters that could serve as a possible "root" was retrieved. For example, the possible "roots" for the word *mktub* (مكتوب) are M-K-T, M-K-U, M-K-B, M-T-U, M-T-B, M-U-B, K-T-U, K-T-B, K-U-B, T-U-B. If the word had only a length of 2, every possible 1-letter "root" was retrieved instead. No "roots" were retrieved from words of length 1. Everything not in a given "root" served as the corresponding "template". For example, if extracting the "root" M-T-B from *mktub*, the corresponding "template" would be ***1k2u3***. A list of valid fake "roots", which were the 28,000 most common possible fake "roots" based on raw frequency in the 10,000 sentences, was then created, as well as a list of valid fake "template stems", which were the 2,500 most frequent possible fake "template stems".

Afterwards, for each word in *RootCache*, all possible parses using the valid fake "roots" and valid fake "template stems" were determined and one was selected at random. If no parse was possible, then the word was treated as if it had no root and template. Choosing parses from the lists of valid fake "roots" and "template stems" was important to restrict the size of the final vocabulary, which otherwise easily explodes. For each word, the selected parse, including the fake "root" with its "template" and "template stem", was cached in *FakeRootCache*.

E Stem-SRE

We describe the training of the Stem-SRE tokenizer, followed by the Stem-SRE sub-word segmentation and "de-segmentation" processes.

Training Stem-SRE. To train this model, a dataset of Arabic sentences is first preprocessed so that for each word that contains a Semitic root (detected with CAMeL Tools), the stem is simply wrapped in angle brackets. For example, the word *alm**KTuB**h* (المكتوبة) is preprocessed as *'alm<KTuB>h'*. A BPE model called Stem-SRE BPE is then trained in the manner described in Section 3.2 on a set of preprocessed data. Before training, stem tokens for all stems in *RootCache*, also wrapped in angle brackets (*e.g.*, '<KTuB>'), are added to the *user_defined_symbols*.

Stem-SRE Segmentation. To segment a sequence with Stem-SRE, the sequence is first preprocessed: words with Semitic roots are detected with CAMeL Tools, and then, for each word with a root, the stem (or subsequence ranging from the first radical to the last radical) is wrapped in angle brackets. The Stem-SRE BPE model then tokenizes the preprocessed sequence.

Stem-SRE Sub-word "De-segmentation". To reverse the segmentation on the model outputs, each sequence is first detokenized with the Stem-SRE BPE model, and then all angle brackets in the sequence are simply removed. This yields the final sentence. The results of using Stem-SRE compared to SRE are discussed in Section 4.3 above.

F SRE-MF

We describe SRE-MF, where MF refers to the "most frequent" words. SRE-MF works much like SRE except that it does not segment the most frequently occurring words into sub-words. The SRE method generally represents a sentence with far more sub-words than BPE does. On one trial of predictions on the test set, the SRE method represented each output sentence with 81.8 tokens on average, whereas the BPE method did with 53.8 tokens. This is due to BPE only representing infrequent word forms as a series of multiple sub-words.

The SRE method, on the other hand, always, where possible, splits a word into at least a root token and a template stem token, and then affix tokens as needed. We considered the possibility that this may complicate the translation task for NMT models. For this reason, we developed SRE-MF where the most frequent word forms are not split into sub-words.

To create an SRE-MF tokenizer, the *n* most frequent word forms (without punctuation) are selected from the tokenizer training data. For these words, *SRE Preprocessing* is not performed, and they are kept as is in the tokenizer training data. These words are then added to the *user_defined_symbols* along with the root and template stem tokens from *RootCache*. The total number of tokens needed to represent special tokens, whitespace, roots, and template stems is 4,266, which leaves 3,735 for everything else. A portion *n* of these leftover tokens are needed to represent the most frequent whole words. There are about 6,000 whole words in the tokenizer training data that occur in the *BPE-ar-8k* tokenizer's vocabulary of 8,001, which suggests, as far as it is possible, that it is worth trying to nearly, though not entirely, max out the *SRE-8k* vocabulary with whole words, leaving a relatively small portion to represent affixes and everything else as determined by the BPE algorithm. We therefore decided to experiment with two values of *n* that accomplish this, selecting 3,418 and 2,433 of the most frequent word forms to add, respectively, to the vocabularies of two SRE-MF subword segmentation models: *SRE-MF-3.4k-8k* and *SRE-MF-2.4k-8k*. Both models had a vocab size of 8,001 and contained 3,956 root tokens and 305 template stem tokens. We used these tokenizers to train the following NMT models:

en2ar-SRE-MF-3.4k was trained tokenizing English source sentences with *BPE-en-8k* and the Arabic target sentences with *SRE-MF-3.4k-8k*, which does not segment ~3.4K of the most frequent words into sub-word tokens.

en2ar-SRE-MF-2.4k was trained tokenizing English source sentences with *BPE-en-8k* and the Arabic target sentences with *SRE-MF-2.4k-8k*, which does not segment ~2.4K of the most frequent words into sub-word tokens.

The scores for these models, trained on the Trial 1 training set, are reported in Table 8 along with that of *en2ar-BPE* and *en2ar-SRE* for comparison, where it is observed that BLEU scores for *en2ar-SRE-MF-3.4k* and *en2ar-MF-2.4k* are narrowly under that of *en2ar-SRE*. This suggests that adding frequent whole words to an SRE vocabulary likely does not have a significant effect on translation quality, though this may need further investigation given that *en2ar-SRE-MF-3.4k* yields a significant increase in chrF.

Model	BLEU	chrF
en2ar-SRE	**27.92**	58.96
en2ar-BPE	26.93*	58.86
en2ar-In-Situ-SRE	27.66	**59.24**

Table 9: BLEU and chrF scores for *en2ar-SRE*, *en2ar-BPE*, and *en2ar-In-Situ-SRE* translation models. Note that the results for *en2ar-BPE* and *en2ar-SRE* are from Trial 1 and also appear in Table 3. * indicates that the scores are statistically significantly different than those of *en2ar-SRE*.

G In-Situ-SRE

In SRE, words are represented first with a root token, followed by 0 or more prefix tokens, followed by the template stem token, followed 0 or more suffix tokens. Given that roots are tied to the stem, rather than affixes, it is arguable that the best order linguistically should be first prefix tokens, followed by the root token, followed by the template stem token, followed by the suffix tokens. We created a modified SRE scheme based on this token order, and called it In-Situ-SRE, as the root remains *in situ*, *i.e.*, in the location of the stem. For example, the word *alm**KT**u**B**h* would be *SRE Preprocessed* as *'alm<KTB>12u3h'* and split into tokens `'alm'`, `'<KTB>'`, `'12u3'`, and `'h'`. We created an In-Situ-SRE tokenizer, which contained 3,956 root tokens and 305 template stem tokens, and a total vocab size of 8,001 called *In-Situ-SRE-8k*.

We trained a single NMT model called *en2ar-In-Situ-SRE* on the Trial 1 training set, tokenizing English source sentences with *BPE-en-8k* and the Arabic target sentences with *In-Situ-SRE-8k*. Table 9 reports its scores together with that of *en2ar-SRE* and *en2ar-BPE*. We observe there is negligible difference in performance based on BLEU and chrF scores between the SRE and In-Situ-SRE methods, suggesting this alternative token order may have no meaningful impact on translation quality.

Arabic Out-of-Dictionary Words

Set	Trial 1		Trial 2		Trial 3		Trial 4		Avg.	
	en2ar-SRE	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***
test	83	93	96	90	106	110	99	103	96	99
ext.	264	287	421	350	771	1,881	492	439	487	739

Non-Arabic Out-of-Dictionary Words

Set	Trial 1		Trial 2		Trial 3		Trial 4		Avg.	
	en2ar-SRE	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***	***en2ar-SRE***	***en2ar-BPE***
test	376	326	162	227	133	160	88	134	190	212
ext.	3,048	2,816	2,148	2,281	2,042	2,911	3,172	2,105	2,603	2,528

Table 10: Arabic and non-Arabic out-of-dictionary words generated by *en2ar-SRE* and *en2ar-BPE* over four trials, when run on the test (1,009 sentences) and *extra* (*ext.*) test (9,669 sentences) sets. Averages have been rounded to the nearest whole number.

	Trial 1					Trial 2				
Set	**Total**	**w/Sem**	**Val**	**ValStm**	**InvStm**	**Total**	**w/Sem**	**Val**	**ValStm**	**InvStm**
test	83	4	4	0	0	96	4	4	0	0
ext.	264	36	30	4	2	421	30	27	3	0

Table 11: **Total** is the number of Arabic out-of-dictionary words and **w/Sem** is the number of those Arabic out-of-dictionary words with a Semitic root. Of those Arabic out-of-dictionary words with Semitic roots, **Val** is the number that are valid words, **ValStm** is the number that have valid stems but invalid affixes, and **InvStm** is the number that have invalid stems. Counts are provided for Trial 1 and 2 predictions of *en2ar-SRE* for the test set and *extra* (*ext.*) test set.

H Out-of-Dictionary Words

H.1 Out-of-Dictionary Words

Table 10 shows the number of Arabic out-of-dictionary words and non-Arabic out-of-dictionary words for all four trials of *en2ar-SRE* and *en2ar-BPE* as described in Section 4.2. We observed no patterns between the number of Arabic or non-Arabic out-of-dictionary words generated by *en2ar-SRE* and *en2ar-BPE*. We do count fewer Arabic out-of-dictionary words generated by *en2ar-SRE* than those generated by *en2ar-BPE* in Trial 3, but we also suspect a lot of long nonsense hallucinations are occurring in Trial 3, explaining perhaps why so many out-of-dictionary words occured.

The ratio of the average number of output tokens to input tokens per sentence for *en2ar-SRE* and *en2ar-BPE* for the test set ranges from 1.47 to 1.51 and 0.98 to 1.01, respectively, across the four trials. Such consistency, noted in the narrow ranges, was not observed for the results of the *extra* test set. For *en2ar-SRE*, the ratios for Trials 1 and 2 were 1.63 and 1.64, but were 1.93 and 1.95 for Trials 3 and 4. This means that Trials 3 and 4 were on average generating much longer sentences, suggesting possibly they were hallucinating a lot.

It may be that Trial 3 *en2ar-SRE*'s hallucinations included more out-of-dictionary words than that of Trial 4, hence the high number of out-of-dictionary words in Trial 3. We noticed that Trial 3 *en2ar-SRE*'s final postprocessed outputs for the *extra* test set included more sentences containing template placeholders (which in practice were unique characters that did not exist in the original data, rather than numbers as was used in the demonstrations in this paper) than that of any of the other trials. This ideally should not happen after postprocessing of the outputs, but does occur, for instance, when a hallucination contains template stem tokens but without root tokens to fill the placeholders. Naturally, this results in out-of-dictionary words, and may explain in part why Trial 3 has more out-of-dictionary words than Trial 4, despite having similar ratios. Additionally, we found that Trial 3's outputs on the *extra* test set also contained more sentences with pound ("#") symbols than the other trials. These occur naturally in data, often in social media hashtags, but they also result sometimes in the middle of words in the final postprocessed output when certain root tokens are paired with incompatible templates. The word-splitting function we used will split words on punctuation characters, including "#", so this may also contribute to the high number of out-of-dictionary words counted in Trial 3. (We note that no instances of "#" or placeholders occur in the final postprocessed output of the standard test set for any of the trials of *en2ar-SRE* and *en2ar-BPE*.)

For *en2ar-BPE* on the *extra* test set, the ratios for Trials 1, 2, and 4 were 1.13, 1.27, and 1.26, whereas the ratio for Trial 3 was 1.93, indicating the latter was generating much longer sentences than the previous three, perhaps because it had this tendency to hallucinate. This could explain the high number of out-of-dictionary word forms generated by Trial 3 of *en2ar-BPE*. As to why some trials may be hallucinating more than others, an analysis of the training data may be needed.

This all to say, because we have reason to believe Trial 3 contains a lot of hallucination, we refrain from drawing conclusions on whether one system tends to generate more or fewer out-of-dictionary words than another.

H.2 Out-of-Dictionary Words With Semitic Roots

Table 11 shows the results of the manual review of Arabic out-of-dictionary words with Semitic roots and the judgements we made together with Evaluator 1. We reviewed the Arabic out-of-dictionary words from the *en2ar-SRE* Trial 1 and Trial 2 hypotheses of the test and *extra* test sets. Of those Arabic out-of-dictionary words with Semitic roots, we counted the number that are actually valid words. Of those that are invalid words, we counted the number that have valid stems with invalid affixes and the number that have invalid stems.