Constrained Sequence-to-sequence Semitic Root Extraction for Enriching Word Embeddings

Ahmed El-Kishky[†][*], Xingyu Fu[†][*], Aseel Addawood[†],
Nahil Sobh[†], Clare Voss[‡], and Jiawei Han[†]
[†]Department of Computer Science, The University of Illinois at Urbana Champaign
[‡]Computational & Information Sciences Directorate, Army Research Laboratory
[†]Urbana, IL, USA, [‡]Adelphi, MD, USA
{elkishk2,xingyuf2,aaddaw2,sobh,hanj}@illinois.edu
clare.r.voss.civ@mail.mil

Abstract

In this paper, we tackle the problem of "root extraction" from words in the Semitic language family. A challenge in applying natural language processing techniques to these languages is the data sparsity problem that arises from their rich internal morphology, where the substructure is inherently non-concatenative and morphemes are interdigitated in word formation. While previous automated methods have relied on human-curated rules or multi-class classification, they have not fully leveraged the various combinations of regular, sequential concatenative morphology within the words and the internal interleaving within templatic stems of roots and patterns. To address this, we propose a constrained sequence-to-sequence root extraction method. Experimental results show our constrained model outperforms a variety of methods at root extraction. Furthermore, by enriching word embeddings with resulting decompositions, we show improved results on word analogy, word similarity, and language modeling tasks.

1 Introduction

The Semitic languages are a language family commonly spoken throughout North Africa, the Horn of Africa, the Arabian peninsula, and the regions between. With approximately 500 million speakers, the proliferation of large online text collections of such news articles, social media, digitized literature, and web blogs has created a wealth of data offering challenges and opportunities for semantic understanding of Semitic texts. In these languages, a majority of words are derived from a small number of mostly triliteral consonantal roots, with some quadriliteral roots and a trace number of biliteral and quintliteral roots. It is estimated that two of the most prominent Semitic languages, Arabic and Hebrew, possess approximately 10,000 and 3,000 roots, respectively (Darwish, 2002; Daya et al., 2008). As such, root identification of a given Semitic word is often an important task in morphological analysis and the first step to morphological decomposition. Morphological analysis of Semitic languages poses a unique challenge to traditional NLP techniques due to the non-contiguous morphology inherent in these languages. This morphology is best described as the application of a *pattern* resulting in the interdigitation of morphemes within a single root to form derivative words (Habash, 2010). This fusional morphology allows for many surface form words derived from the same single root, but with different, yet abstractly-related semantic meanings depending on constituent morphemes. Because many surface words can be formed through this root and pattern word formation process, and the root's characters may not necessarily be contiguously situated within each resultant surface word, morpheme boundaries are often difficult to identify.

Unlike other fusional languages, the Semitic languages are unique in that the word formation process follows a highly-structured process of adding vowels and consonants to roots. This word formation process consists of a fixed number of slots for different morphemes, which are fixed in their position and order relative to each other. As such, these languages contain significant sequential (albeit not necessarily contiguous) substructure. In this work, we propose to leverage this sequential substructure to improve the root extraction process and morphological decomposition.

Morphological analysis is essential in working with Semitic languages as well as other highly-inflectional languages due to data sparsity. For instance, previous research has shown that many text corpora demonstrate long-tail distributions in

[*]Equal contribution

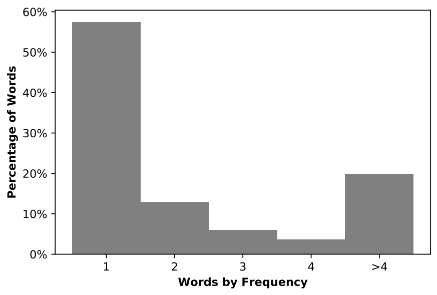


Figure 1: Word distribution in Arabic Wikipedia corpus.

relation to word frequency. This long-tail often results in corpora with many infrequent words, with 40% – 60% of words appearing just once (Kornai, 2007). We can verify this for Arabic in Figure 1, where, on a Wikipedia monolingual Arabic corpus (described in Section 5.1), approximately 80% of words occur fewer than five times and 60% occur once. To process such long-tailed corpora, it is necessary to exploit finer-granularity, highly-shared substructures between words that can be used to infer semantic meaning. In Table 1, we look at a selection of Arabic words sharing the common root –ب ت ك– (*transliteration K-T-B*), which means to *"write"*. These words are formed by appending different prefixes, suffixes, and other templatic interleavings of morphemes within the root. Despite the many surface words, the derivations share a semantic relationship based on the root, as well as other concatenative and interdigitated templatic morphemes. Additionally, as seen in the example, the root word's characters are not necessarily contiguous within the word; this is due to the non-concatenative templatic process whereby morphemes are inserted between characters of the root as part of the word formation process. Finally, not all characters in the root are necessarily found in the final surface-form of the word as some root characters can be dropped. Traditional concatenative morphological analyzers struggle to identify and extract roots precisely because root word characters are not necessarily contiguous or even present in the surface word.

To address these challenges, we present a supervised root extraction algorithm that, given a word, directly extracts the root with high accuracy. Given this root and the original word, we demonstrate how the templatic pattern-based word formation process that transforms the root to the original word can be used for further morphological decomposition. Our root extraction method differentiates itself from other methods in three

Word	*Translit.*	*Meaning*	*Pref.*	*Suff.*	*R-1*	*R-2*
كتبت	KTBT	she wrote	N/A	T	N/A	N/A
كاتب	KĀTB	writer	N/A	N/A	Ā	N/A
كتاب	KTĀB	book	N/A	N/A	N/A	Ā
الكتاب	ALKTĀB	the book	AL	N/A	N/A	Ā
مكتب	MKTB	desk	M	N/A	N/A	N/A
مكتبة	MKTBA	library	M	A	N/A	N/A

Table 1: Common Roots

ways: (1) It is fully data-driven, without any reliance on human-curated patterns; (2) it directly extracts word roots without stripping dictionary affixes, which can lead to incorrect roots when false affixes are stripped; and (3) by applying a novel sequence-to-sequence (seq2seq) model with a constrained decoding mechanism that leverages shared sequential semantics in the label (root) and input (word) space, it outperforms standard multi-class classification algorithms and achieves better generalization performance.

We demonstrate that our method outperforms unsupervised rule-based root extraction methods (Taghva et al., 2005; Khoja and Garside, 1999; Zerrouki, 2010) and our seq2seq classifier outperforms general multiclass classifiers (Kim, 2014; Chung et al., 2014). As a testament to the utility of root extraction, we demonstrate how one can leverage the root information alongside a simple slot-based morphological decomposition to improve upon word embedding representations as evaluated through word similarity, word analogy, and language modeling tasks.

2 Related Work

With the growth of the internet and the digitization of Arabic and other Semitic corpora, prior work has extensively studied root extractors with the goal of improving document retrieval (Larkey et al., 2002; Aljlayl and Frieder, 2002).

Early approaches to the problem of Arabic root extraction were predominantly unsupervised methods. Some researchers developed stemmers that remove some prefixes and suffixes while ignoring the templatic, interleaved morphemes within stems. A few of these methods relied on pattern matching and prefix/suffix pruning in order to extract roots (Taghva et al., 2005; Khoja and Garside, 1999). These methods may fail to identify the roots in many nouns and, like all prefix and suffix stripping algorithms, fail to correctly extract non-contiguous roots. Similar methods operate by removing not only prefixes and suffixes, but also "extra letters" until the triconsonantal roots remain (Momani and Faraj, 2007).

This method, however, may incorrectly remove many letters that are part of the root. Another of these models achieves high accuracy by incorporating sentence-level context and inferred syntactic categories into a parametric Bayesian model (Lee et al., 2011). Our model forgoes these context features as it attempts to identify the root solely on the word itself. Additionally, this method cannot model non-contiguous roots, of which Semitic languages have many. Other unsupervised methods utilize dictionaries to select the characters from within words (Darwish, 2002; Boudlal et al., 2011; Alhanini and Ab Aziz, 2011). Another line of research leverages the templatic nature for human-constructed rule-based constraints (Elghamry, 2005; Rodrigues and Cavar, 2007; Choueka, 1990). Finally, methods have been proposed that utilize both a root dictionary and rule-based templatic constraints (Yaseen and Hmeidi, 2014).

Supervised methods have been developed for identifying Hebrew roots by combining various multiclass classification models with Hebrew-specific linguistic constraints (Daya et al., 2004). This same technique was extended to extract both Arabic and Hebrew roots (Daya et al., 2008). While these supervised methods effectively address the non-contiguous nature of Semitic roots, they fail to leverage the sequential structure of the root label space. We show that such methods that forgo the sequential structure in the label space underperform on words with rare roots. Additionally, these methods are only applied to triconsonantal leaving out many biconsonantal and quadriliteral roots.

Sequence-to-sequence models have been utilized for learning to map sequences to other sequences and predominantly applied to machine translation (Sutskever et al., 2014), with later variations of these models enhanced with attention mechanisms (Luong et al., 2015). While LSTM variants have been dominant, previous work has shown that GRU-based models perform comparably to LSTM-based models with superior train time (Chung et al., 2014). More recent work has investigated character-level language models in order to handle the many out-of-vocabulary (OOV) words in morphologically rich languages (Gerz et al., 2018). Such methods have shown large improvements in language modeling across many morphologically rich languages. While such methods share the same character-level input space as does our own method, they ignore the sequential nature in the target class. Closely related to our model, constrained sequence-to-sequence models have been used for sentence simplification forcing the model to select simple words (Zhang et al., 2017). Similar approaches have been used for constrained image captioning (Anderson et al., 2017). Our model differs in that it constrains not only on specific vocabulary, but on specific sequences.

3 Root Extraction Framework

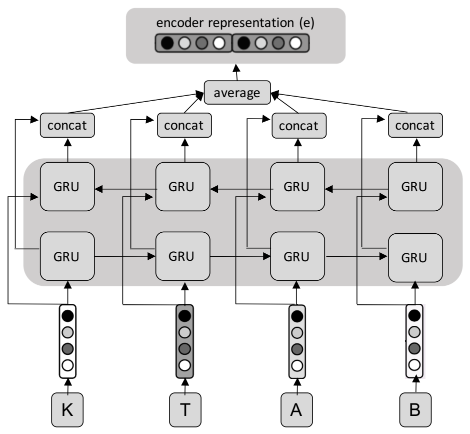
We introduce a framework for extracting the root from templatic words within the Semitic family. The proposed framework leverages the shared sequential semantics in both the word and root space to more accurately extract root morphemes.

3.1 Preliminaries

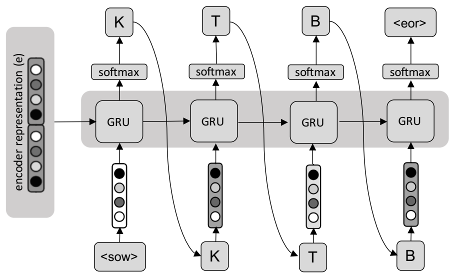
The input is a set of word-root pairs W, R, consisting of $|W|$ words and $|R|$ roots where $|W| = |R|$ and $W = w_1, \ldots, w_{|W|}$ and $R = r_1, \ldots, r_{|R|}$. In addition, the j^{th} word w_j is a sequence of $|w_j|$ characters: $c_{w_j,i}, i = 1, \ldots, |w_j|$. For convenience we index all the unique characters that compose the input vocabulary with C characters and $c_{w,i} = x$, where $x \in \{1, \ldots, C\}$ means that the i^{th} character in w^{th} word is the x^{th} character in the character vocabulary. Similarly the k^{th} root, r_k corresponding to the j^{th} word w_j is a sequence of $|r_k|$ characters: $c_{r_k,i}, i = 1, \ldots, |r_k|$. Given the input, the goal is to learn a function, $\mathcal{F} : W \rightarrow R$ that maps an input word onto its correct Semitic root.

3.2 Constrained Seq2Seq Root Extraction

Our main innovation and contribution is a unique way of extracting roots by utilizing seq2seq models for multiclass classification. While many methods traditionally approach root extraction through unsupervised application of templates or traditional supervised multiclass classification algorithms, we posit that the shared semantics between words and roots merits a different approach. As such, we apply a hybrid approach between multiclass classification and seq2seq models for root extraction. By constraining the outputs of the seq2seq models to the dictionary table of roots, the algorithm becomes a sequential multiclass classification model that implicitly leverages shared sequential substructure in both the input space and in the label space.



(a) Encoder Network



(b) Decoder Network

Figure 2: Sequence-to-sequence root extraction.

3.2.1 Encoder Network

As seen in Figure 2a, we begin with an encoder network that takes a word as input. Each of the input word's characters (from a total of C possible characters) is associated with a vector $c \in \mathbb{R}^d$. Using word, KTĀB from Table 1, the input becomes vector $[c_0, c_1, c_2, c_3] \in \mathbb{R}^{d \times 4}$. We then run this sequence of embedding vectors through both directions of a bi-directional GRU (BiGRU) and concatenate the resulting hidden vectors from each pass. Finally, we average the concatenated hidden vectors of the BiGRU across all time-steps. This serves as the encoder representation of the input word, which we denote as e. The encoding is then fed into a decoder network that attempts to generate the most likely root for the word.

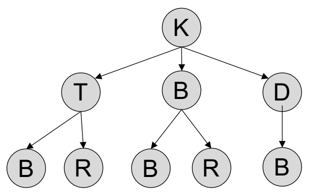
3.2.2 Decoder Network

In Figure 2b, the decoder takes the encoder representation e that captures the input word and predicts a root word. This is done by feeding e and a special "start-of-word" character $\langle \text{sow} \rangle$ as the input. A GRU computes the next hidden state $h_0 \in \mathbb{R}^h$. A scoring function is then applied, resulting in an output the size of the character vocabulary, C. This function: $\mathbf{g} : \mathbb{R}^h \rightarrow \mathbb{R}^C$, is then softmaxed to obtain a valid probability distribution over characters for each hidden state. The decoding stops when the predicted root is terminated with a special "end-of-root" token $\langle \text{eor} \rangle$.

3.2.3 Constrained Beam Search

Traditional decoders select the best character at each step to feed into the next time step of the RNN. However, this decoding maps the input sequence into an infinite space of possible output sequences and, as such, may result in an *invalid root* that is not part of the dictionary set of roots. As such, we propose an alternative output that restricts the decoder, forcing the decoded sequence to map onto a root within the valid roots set.

We realize this constraint by modifying the decoding scheme itself. During decoding, a greedy approach is often used where the single best character output is selected and propagated to later time steps. This greedy approach may not only lead to suboptimal output sequences, but also result in invalid sequences (not corresponding to any class). This can be circumvented using a beam search decoding scheme. When decoding to obtain the predicted roots, instead of utilizing the character with the highest probability at each step, the top k characters are considered at each step. As such, at each new time-step, for each of the k hypotheses, there are C possible choices. The top k are then once again selected and this process is applied to each time step. Once all candidate roots reach their special $\langle \text{eor} \rangle$ token, the most probable root is selected. To tailor beam search to



Candidate Roots
~~K - T - A~~
K - T - B
~~K - T - A - B~~
K - T - R
~~K - T - B - B~~

(a) Constraint Trie (b) Candidate root pool.

Figure 3: Constrained beam search.

root extraction from a dictionary of roots, we seek to modify beam search by enforcing the linguistic sequential constraints present in the label root set. This leverages our classification tasks's relatively small and enumerable root label set, contrasted with an unbounded sequence as found in machine translation models. Simultaneously, by using a decoder, the model exploits the task's sequential structure by generating the target label character-by-character. We utilize the target roots as guidance for the decoding process in order to implement this sequential prediction. We demonstrate on a toy example in Figure 3a, where by storing all the possible target roots in a trie data structure (a.k.a a prefix tree), invalid roots can be pruned during the decoding process. For example, as seen in Figure 3b, during a typical beam-search process, the top k candidate characters are selected. By cross-referencing the current prefix of the root with the trie storing all valid roots, many invalid roots can be pruned. As such, we can enforce that the top-k selections all correspond to valid prefixes present in the target roots. This strictly improves overall extraction accuracy over traditional beam search.

4 Templatic Word Embeddings

As the Semitic languages are templatic, there exist fixed slots that can contain morphemes. Given the correct root for a word identified as described in Section 3, we introduce a simple slot-based template. We indicate how to identify these slots within a word utilizing the Semitic root. Finally, we demonstrate how the morphemes within these slots, along with the root, can be utilized to enrich distributed word representations.

4.1 Morphological Decomposition

We posit that each word possesses a fixed number of slots allocated to certain morphemes, whereby the slots are fixed in their position and order relative to each other. As demonstrated in Table 1, in addition to the root word, we propose a simplified template that consists of four slots – two concatenative (prefixes and suffixes) and two non-concatenative (morphemes interdigitated within the stem). While we demonstrate the simplicity of identifying these within Arabic, this same template-based structure can, without loss of generality, be trivially created for other members of the Semitic family.

Example 1 (Stem, Prefix, and Suffix Identification) *For the root K-T-B, we can identify the consecutive characters that encompass the full root.*

AL + [KTĀB] + EEN

ال + [كتاب] + ين

The characters grouped together by [] form the stem, the smallest consecutive set of characters containing the full root. Any characters not falling within the stem are, respectively, the prefixes and suffixes.

As seen in Example 1, given the root, the stem can be identified as the shortest contiguous substring containing the root in correct order. Once the stem is identified, the two concatenative slots containing prefix and suffix are trivially identified by selecting the remaining affixes after removing the stem. The non-concatenative slots can be found interdigitated within the word stem whose boundary is demarcated by the root. Given the stem (as shown in square brackets in Example 1) and the root, these interdigitated slots can be identified as follows:

Example 2 (Interdigitated Slots) *Given a stem containing the core root K-T-B, the candidate slots are as follows.*

In stem, KĀTB, Ā occurs in the first slot.

In stem, KTĀB, Ā occurs in the second slot.

If a contiguous morpheme occurs after the first character in the root by before middle characters, it is a slot-1 addition. If after the middle character(s) of the root, it is slot-2.

Example 2 shows the identification of interdigitated slots within the stem. Once again, it is evident that correct extraction of the root is essential to correct identification of the slot positions within the word. In the next subsection we demonstrate how these extractions can be systematically leveraged to enrich distributed word representations in these templatic languages.

4.2 Morpheme-Enriched Embeddings

To demonstrate the utility of templatic subword extractions, we demonstrate how enriching word embeddings with these morphemes can improve word representations by providing parameter-sharing between words sharing common substructure. With this motivation, we propose TemplaticVec, an intuitive extension to FastText (Piotr Bojanowski and Mikolov, 2017), that utilizes the templatic decomposition of semantically-meaningful roots, affixes, and interdigitated morphemes for representation enrichment. By using these structures as embedding base units by and combining them to construct a word's distributed vector representation, the resultant word embeddings are robust to infrequent word-induced data-sparsity and can be constructed on many out-of-vocabulary (OOV) words. We begin with a brief review of FastText, and then demonstrate how one can naturally integrate roots as well as concatenative and templatic morphemes in place of FastText's standard naive subwords. FastText utilizes the skip-gram objective with negative sampling yielding the following objective (for simplicity, $\ell(x) = \log(1 + \exp(-x))$):

$$\sum_{x=1}^{|W|} \Big[\sum_{c \in \mathcal{C}_x} \ell(s(w_x, w_c)) + \sum_{t \in \mathcal{N}_{x,c}} \ell(-s(w_x, t)) \Big]$$

In the above equation, w_x is the x^{th} word in the corpus, $\mathcal{C}_x$ denotes the set of context words within a predefined window of word w_x, and $\mathcal{N}_{x,c}$ denotes the set of negative examples sampled from outside the context window.

The scoring function is then adapted to incorporate subword information as follows:

$$s(w_x, w_c) = \sum_{m \in w_x} \mathbf{z}_m^\intercal \mathbf{v}_c$$

In the above equation, each $\mathbf{z}_m$ denotes a subword embedding vector, so that the scoring function equates to the inner product of the summation each over subword embedding vector with the context word vector. While FastText incorporates all contiguous substrings of lengths three to seven as morphemes in the scoring function, because Semitic roots are not necessarily contiguous, two words sharing the same root may not share the same subwords using FastText. Because this important semantic morpheme is not shared among words, we posit that FastText's indiscriminate enumeration of contiguous subwords does not capture the essential semantic substructure. We claim that directly incorporating the root embedding and each slot's morpheme embeddings that have been extracted for each word and summing over these embeddings results in higher quality distributed representations. As such, similar to the approach in (El-Kishky et al., 2018), we modify the scoring function to incorporate the extracted root and slot-based templatic information:

$$s(w_x, w_c) = (\mathbf{z}_r + \mathbf{z}_p + \mathbf{z}_s + \mathbf{z}_{r1} + \mathbf{z}_{r2})^\intercal \mathbf{v}_c$$

This modification yields a scoring function that is the inner product of the summation over the root word embedding ($\mathbf{z}_r$), prefix embedding ($\mathbf{z}_p$), suffix embedding ($\mathbf{z}_s$), as well as the two possible in-root interdigitated morphemes ($\mathbf{z}_{r1}$ and $\mathbf{z}_{r2}$).

5 Experiments

We introduce the datasets and methods for comparison used. We then describe evaluations for root extraction and embedding quality.

5.1 Datasets and comparison methods

We use the following datasets and ground-truth labels for evaluation purposes:

- **Arabic Word & Root Pairs**: 140K words along associated with 11K roots from dictionary (al Zabidi and Murthada, 1886).
- **Hebrew Word & Root Pairs**. 11.5K words associated with approximately 500 roots from Wiktionary[1] and human curation.
- **Arabic Wikipedia Corpora**. Wikipedia corpus with 274K articles and 62.5M tokens and 1.26M unique words.

[1]wiktionary.org

For baseline methods to compare against our proposed constrained seq2seq (Constrain-S2S), we evaluate against three standard multiclass classification models: (1) a standard convolutional neural network, CNN-Class, (Kim, 2014), a GRU model, GRU-Class, and a bi-directional GRU model, BiGRU-Class. In addition, we compare against two unconstrained seq2seq models, encoder-decoder models using GRUs, GRU-S2S and bi-directional GRUs, BiGRU-S2S. Finally, for Arabic, we evaluate against three unsupervised Arabic root-extraction algorithms from the literature: Tashaphyne, ISRI, and Khoja. To evaluate on the quality of the resultant morphological decomposition, we compare against three variants of embeddings: (1) SkipGram (2) FastText (3) RootVec (Embedding enriched with solely the root) .

5.2 Root Extraction Accuracy

To evaluate the effectiveness of our proposed seq2seq extraction of roots, we perform five-fold cross-validation evaluation of our method compared to a variety of supervised and rule-based root-extraction methods. During each cross-validation, each supervised method is trained on four-fifth of the dictionary mappings of *word to root* pairs, and evaluated on a held-out 20%.

5.2.1 General Root Extraction

We first compare the performance of each supervised extraction method on extracting roots irrespective of root frequency. In Table 2, we report the performance of each extractor at successfully identifying the ground-truth root in each held-out word in a five-fold cross-validation evaluation. It is apparent that the unsupervised methods under-perform at extracting the ground-truth root as compared to the supervised methods. This is likely due to errors from human-curated patterns which possess many exceptions as well as many Semitic roots being non-contiguously situated with the word due to interdigitated mor-

Method	**Arabic**		**Hebrew**	
	ACC.	**SE**	**ACC.**	**SE**
CNN-Class	.6753	±.0009	.9622	±.0019
GRU-Class	.7539	±.0023	.9591	±.0033
BiGRU-Class	.7548	±.0015	.9629	±.0009
GRU-S2S	.7596	±.0017	.9692	±.0013
BiGRU-S2S	.7854	±.0010	.9788	±.0016
Constrain-S2S	**.8324**	**±.0011**	**.9879**	**±.0008**
Tashaphyne	.2778	0	-	-
ISRI	.4508	0	-	-
Khoja	.4434	0	-	-

Table 2: Root Extraction Accuracy.

phemes. Additionally, both the CNN-based and four RNN-based multiclass classification methods severely under-perform compared to our proposed constrained seq2seq model. This verifies our intuition that leveraging the shared semantic space between the words and the target roots is essential in extraction.

5.2.2 Rare Root Extraction

We claimed earlier that by decomposing root classification into seq2seq classification, sequential patterns within the roots can be leveraged for root extraction. This can be useful for identifying the correct root, even when the root is infrequent or even absent from the training data. To support this claim, we report the performance of each supervised extractor at successfully identifying the ground-truth of infrequent roots (appear three or fewer times in training) and a zero-shot case where the root is not present in the training data. As our Hebrew dataset consists of frequent roots, and performance is near perfect, we report results for the Arabic dataset.

Method	**Infreq.**		**Zero-Shot**	
	ACC.	**SE**	**ACC.**	**SE**
CNN-Class	.4823	±.0096	-	-
GRU-Class	.5697	±.0103	-	-
BiGRU-Class	.5706	±.0091	-	-
GRU-S2S	.6074	±.0166	.5389	±.0188
BiGRU-S2S	.6231	±.0191	.5532	±.0141
Constrain-S2S	**.6929**	**±.0164**	**.6292**	**±.0160**

Table 3: Arabic Rare Root Extraction Accuracy

As seen in Table 3, the seq2seq methods greatly outperform all multiclass methods with Constrain-S2S outperforming all methods on the infrequent roots. This effect is amplified in the zero-shot case, with only the seq2seq models handling unseen roots. This demonstrates the utility in jointly learning the sequential structure in semantically-shared label (root) and word space.

5.3 Word Analogy Evaluation

Given our comprehensive dataset of Arabic roots and human-curated evaluation set of Arabic word embeddings, we show the effectiveness of enriching Arabic word embeddings with their morphological decompositions via a word analogy task. The goal of said task is to identify the best value for D in analogies of the form "A is to B as C is to D". After training each embedding model on the Arabic Wikipedia dataset, we use an analogy dataset (Elrazzaz et al., 2017) curated for methodological evaluation of Arabic word embeddings. We further differentiate the analogies into two categories: (1) morphemic analogies (e.g. plurals, tense or gender) where a derivational or inflectional morpheme is inserted, removed, or replaced while the root remains unchanged, and (2) semantic analogies where the root itself changes between the analogous pairs (e.g. bird is to fly as fish is to swim).

Embedding Model	**Semantic**	**Morphemic**
SkipGram	**19.1**	11.4
FastText	13.8	16.8
ISRI-RootVec	15.4	11.2
BiGRU-RootVec	14.2	11.9
S2S-RootVec	18.0	11.9
CS2S-RootVec	18.9	12.2
ISRI-TemplaticVec	15.3	14.5
Class-TemplaticVec	16.3	16.9
S2S-TemplaticVec	17.6	20.2
CS2S-TemplaticVec	18.8	**22.9**

Table 4: Word Analogies

As seen in Table 4, embeddings that utilize morphemes or subword-level features perform significantly better at morphemic analogies than do SkipGram word embeddings. This does not extend to semantic analogies where all methods appear to degrade with the use of morpheme and subword-level enrichment. This is not surprising since, under the vector algebra that is used to compute the word analogies, the summation of the morphemes used to enrich the embeddings captures morphemic relationships but not necessarily semantic ones. This can be seen in the performance gap between the morpheme-enriched embeddings and SkipGram. Unlike the other methods, Templatic embeddings based on constrained roots maintains comparable performance to SkipGram on the semantic analogies while demonstrating superior performance on the morphemic analogies.

5.4 Word Similarity

The next embedding evaluation we consider is a word similarity task. The ground truth data consists of pairs of words and a human-annotated similarity score averaged across all human evaluations from a translation of the WS-353 dataset (Freitas et al., 2016). The scores are computed via the cosine similarity between the vector representation of each word in a pair. Their results are quantified through Spearman and Pearson rank correlation coefficients.

As seen in Table 5, enriching the embedding vectors with the template-based extracted

Embedding Model	Pearson	Spearman
SkipGram	0.496	0.520
FastText	0.459	0.468
ISRI-RootVec	0.491	0.518
BiGRU-RootVec	0.492	0.510
S2S-RootVec	0.508	0.516
CS2S-RootVec	0.507	0.514
ISRI-TemplaticVec	0.482	0.501
Class-TemplaticVec	0.474	0.491
S2S-TemplaticVec	**0.514**	0.529
CS2S-TemplaticVec	0.512	**0.533**

Table 5: Word Similarity

Embedding Model	Perplexity	
	LM-One	LM-Two
SkipGram	1757	1075
FastText	1720	1069
ISRI-RootVec	1729	1072
BiGRU-RootVec	1731	1071
S2S-RootVec	1728	1071
CS2S-RootVec	1726	1071
ISRI-TemplaticVec	1728	1071
Class-TemplaticVec	1724	1070
S2S-TemplaticVec	1718	**1065**
CS2S-TemplaticVec	**1716**	**1065**

Table 6: Language Modeling

morphemes substantially improves embeddings in capturing word similarity. This is in contrast with lower correlation coefficients from FastText embedding vectors, likely due to the indiscriminate generation of subwords that may degrade the overall embedding. On this task, template-based decomposition using unconstrained and constrained root extraction appears to perform similarly, yet both greatly outperform the other baselines.

5.5 Language Modeling Perplexity

Finally, we evaluate the effect of utilizing the extracted root and templatic decomposition on a downstream language modeling task. On each language model, the model quality is evaluated by computing the perplexity on a held-out portion of the corpus. The model used for language modeling is an LSTM with three hidden layers, 600 hidden units per layer, regularized with 0.2 probability drop-out, unrolled for 35 steps with a batch of 20. Parameters are learned using Adagrad with a gradient clipping of 1. We evaluate on two subsets of the Wikipedia dataset: (1) LM-1, a small subset (2) LM-2, a larger subset. LM-1 consists of 3.3M tokens and a vocabulary of 260K words while LM-2 consists of 7.6M tokens and a vocabulary of 400K unique words. Each language model instance is trained for 5 epochs on the training data. Evaluation of perplexity was computed for each model on the independent test set consisting of 900K tokens where 62K tokens were OOV in LM-1 and 27K in LM-2. Evaluation is performed after selecting the best performing iteration of the model on a validation set. While the morpheme-enriched method can generate embedding vectors for many OOV tokens, for SkipGram and instances when they cannot, an unknown token with fixed embedding is used.

The results are summarized in Table 6. Although perplexity is high, this is common for morphologically-rich languages such as Arabic as shown in (Gerz et al., 2018). It appears our constrained model's extracted roots yield a benefit over other baseline roots, yet utilizing the full decomposition outperforms all other methods, yielding lower held-out perplexity. The results also verify the intuition that morphemic decomposition is necessary to handle data-sparsity and OOV words when little training data is present, whereby perplexity is greatly reduced through the use of morpheme-based embeddings.

6 Conclusions

We proposed leveraging the shared semantic space between Semitic words and their roots for more effective root extraction. This was accomplished through a novel constrained sequence-to-sequence classifier. Experiments show a performance boost over unsupervised and supervised extraction models. We introduce a simple template-based morphological decomposition, and by enriching word embeddings with this decomposition, we show improved results on word analogy, word similarity, and language modeling tasks.

References

Yasir Alhanini and Mohd Juzaiddin Ab Aziz. 2011. The enhancement of arabic stemming by using light stemming and dictionary-based stemming. *JSEA*.

Mohammed Aljlayl and Ophir Frieder. 2002. On arabic search: improving the retrieval effectiveness via a light stemming approach. In *CIKM*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *EMNLP*.

Abderrahim Boudlal, Mohamed Ould Abdallahi Ould Bebah, Abdelhak Lakhouaja, Azzeddine Mazroui, and Abdelouafi Meziane. 2011. A markovian approach for arabic root extraction. *Int. Arab J. Inf. Technol.*

Yaacov Choueka. 1990. Mlim-a system for full, exact, on-line grammatical analysis of modern hebrew. In *ICCE*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Kareem Darwish. 2002. Building a shallow arabic morphological analyzer in one day. In *ACL-02 workshop on Computational approaches to semitic languages*.

Ezra Daya, Dan Roth, and Shuly Wintner. 2004. Learning hebrew roots: Machine learning with linguistic constraints. In *EMNLP*.

Ezra Daya, Dan Roth, and Shuly Wintner. 2008. Identifying semitic roots: Machine learning with linguistic constraints. *Computational Linguistics*.

Ahmed El-Kishky, Frank Xu, Aston Zhang, Stephen Macke, and Jiawei Han. 2018. Entropy-based subword mining with an application to word embeddings. In *SCLeM 2018*.

Khaled Elghamry. 2005. A constraint-based algorithm for the identification of arabic roots. In *Proceedings of the Midwest Computational Linguistics Colloquium. Indiana University. Bloomington, IN*.

Mohammed Elrazzaz, Shady Elbassuoni, Khaled Shaban, and Chadi Helwe. 2017. Methodical evaluation of arabic word embeddings. In *ACL*.

André Freitas, Siamak Barzegar, Juliano Efson Sales, Siegfried Handschuh, and Brian Davis. 2016. *Semantic Relatedness for All (Languages): A Comparative Analysis of Multilingual Semantic Relatedness Using Machine Translation*.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *TACL*.

Nizar Y Habash. 2010. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*.

Shereen Khoja and Roger Garside. 1999. Stemming arabic text. *Lancaster, UK, Computing Department, Lancaster University*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Andras Kornai. 2007. *Mathematical linguistics*. Springer Science & Business Media.

Leah S Larkey, Lisa Ballesteros, and Margaret E Connell. 2002. Improving stemming for arabic information retrieval: light stemming and co-occurrence analysis. In *ACM SIGIR*.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *CoNLL*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mohanned Momani and Jamil Faraj. 2007. A novel algorithm to extract tri-literal arabic roots. In *Computer Systems and Applications, 2007. AICCSA'07*.

Armand Joulin Piotr Bojanowski, Edouard Grave and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*.

Paul Rodrigues and Damir Cavar. 2007. Learning arabic morphology using statistical constraint-satisfaction models. *Amsterdam studies in the theory and history of linguistic science. Series 4*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Kazem Taghva, Rania Elkhoury, and Jeffrey Coombs. 2005. Arabic stemming without a root dictionary. In *Information Technology: Coding and Computing*.

Qussai Yaseen and Ismail Hmeidi. 2014. Extracting the roots of arabic words without removing affixes. *Journal of Information Science*.

Muhammad Murtada al Zabidi and Sayyid Muhammad Murthada. 1886. Taj al-'arus min jawahir al-qamus. *Dar al-Hidayah*.

Taha Zerrouki. 2010. Tashaphyne, arabic light stemmer/segment.

Yaoyuan Zhang, Zhenxu Ye, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. A constrained sequence-to-sequence neural model for sentence simplification. *arXiv preprint arXiv:1704.02312*.