Utilizing Subword Entities in Character-Level Sequence-to-Sequence Lemmatization Models

Nasser Zalmout and **Nizar Habash**
Computational Approaches to Modeling Language (CAMeL) Lab
New York University Abu Dhabi
United Arab Emirates
{nasser.zalmout,nizar.habash}@nyu.edu

Abstract

In this paper we present a character-level sequence-to-sequence lemmatization model, utilizing several subword features in multiple configurations. In addition to generic n-gram embeddings (using FastText), we experiment with concatenative (stems) and templatic (roots and patterns) morphological subwords. We present several architectures that embed these features directly at the encoder side, or learn them jointly at the decoder side with a multitask learning architecture. The results indicate that using the generic n-gram embeddings (through FastText) outperform the other linguistically-driven subwords. We use Modern Standard Arabic and Egyptian Arabic as test cases, with up to 22% and 13% relative error reduction, respectively, from a strong baseline. An error analysis shows that our best system is even able to handle word/lemma pairs that are both unseen in the training data.

1 Introduction

Lemmatization is an important natural language processing task that maps inflected and possibly ambiguous word forms to abstract lemmas. Lemmatization is particularly important for modeling morphologically rich languages to address the sparsity of inflected forms. Data-driven lemmatization models have the challenge of generalizability beyond the training data, to handle unseen words, unseen lemmas and new contexts. Semitic languages, like Arabic and Hebrew, add other challenges: while words are generally written without disambiguating diacritical marks, unambiguous lemmas are expected to be fully diacritized; and there are many clitics that may be interpreted as part of the stem leading to alternative lemma readings in different contexts (see Table 1). Semitic languages are also known for their extensive use of templatic morphology which can greatly help in the task of lemmatization, except that words can have multiple pattern (template) readings also. As such, we expect that modeling subword features is a key to the lemmatization task. In this paper we experiment with several kinds of subword entities, without relying on expensive external resources. We experiment with generic n-gram embeddings, captured through FastText (Bojanowski et al., 2017), which generates word embeddings through n-gram sequence embedding. We also introduce and utilize *greedy* morphological subword features, which can be determined consistently at runtime *without* using any morphological dictionaries (Greedy columns in Table 1). We also use Morfessor (Creutz and Lagus, 2005) to obtain morpheme-like subwords. We compare several architectures that condition on these subwords at the encoder, or learn them jointly in a multitask learning architecture.

The results indicate that the generic FastText embeddings matched or outperformed the other linguistically-driven subwords. These subwords are nevertheless considered linguistically shallow, as opposed to models using deeper – and more expensive – linguistic resources, like morphological analyzers (Zalmout and Habash, 2019; Pasha et al., 2014). We use Modern Standard Arabic (MSA) and Egyptian Arabic (EGY) as test cases, where our models provide about 22% relative error reduction for MSA, and 13% for EGY, from a baseline using character-level sequence-to-sequence model with attention. Our models also outperform a strong baseline system that uses rich morphological dictionaries, and can scale to unseen words/lemmas.

				Gold			**Greedy**		
#	**Surface Form**	**Lemma**	**Gloss [Lemma]**	**Stem**	**Pattern**	**Root**	**Stem**	**Pattern**	**Root**
(1a)	wllbywt	bayt	and for the [house]s	bywt	wll+__w_+	byt	bywt	wll+_yw_+	bt
(1b)	Albywt	bayt	the [house]s	bywt	Al+__w_+	byt	bywt	Al+_yw_+	bt
(1c)	wllsywf	sayf	and for the [sword]s	sywf	wll+__w_+	syf	sywf	wll+_yw_+	sf
(2a)	AlktAb	kut∼Ab	the [writer]s	ktAb	Al+__A_+	ktb	ktAb	Al+__A_+	ktb
(2b)	AlktAb	kitAb	the [book]s	ktAb	Al+__A_+	ktb	ktAb	Al+__A_+	ktb
(3a)	kbAby	kabAb	my [kebab]	kbAb	__A_+y	kbb	bAb	k+_A_+y	bb
(3b)	kbAby	bAb	like my [door]	bAb	k+_A_+y	bwb	bAb	k+_A_+y	bb

Table 1: Examples highlighting the complexity of lemmatization in MSA. Examples 1a-c highlight cases where greedy features, albeit different from their gold values, can consistently help across words. Examples 2a-b are ambiguous cases but with identical gold and greedy features; while, in examples 3a-b, greedy features are inconsistent with gold features and the disambiguation task is more involved.

2 Related Work

Lemmatization has been shown to be very useful in several NLP tasks, including machine translation (Sennrich and Haddow, 2016; Fraser et al., 2012) and parsing (Dozat and Manning, 2018; Björkelund et al., 2010; Seddah et al., 2010), among others. Early contributions used finite state machines (Schmid et al., 2004; Minnen et al., 2001), which could not handle unseen words and had limited in-context modeling capacity. Other contributions approached lemmatization as a lemma-selection task (Roth et al., 2008; Ezeiza et al., 1998), where the goal is to select the correct lemma from a set of lemmas provided by a morphological analyzer. These systems handled in-context analysis better, but had a limited capability of modeling unseen words. To handle unseen words, these models rely entirely on the morphological analyzer, and would usually involve other tasks that might help the disambiguation pipeline (like POS tagging). Other data-driven models approach lemmatization as a classification task (Müller et al., 2015; Chrupala et al., 2008), through obtaining the set of edit operations that transform words into lemmas, and then learning to select the optimal lemma from a set of candidates. These candidates are generated by applying all possible edit-trees learned from the training data. There were also several lemmatization contributions utilizing sequence-to-sequence models (Bergmanis and Goldwater, 2018; Pütz et al., 2018), which consider lemmatization as a generation task. Several other contributions use additional morphosyntactic features as part of the modeling architecture (Kanerva et al., 2019; Kondratyuk et al., 2018). As far as we are aware, no other contributions explicitly utilize other word-level features, like embeddings, stems, or orthographic patterns, in lemmatization.

3 Approach

Greedy vs Gold Stems The stem is the form of the word after all inflectional affixes are removed from the surface form. The surface form can be represented as *(prefix+stem+suffix)*, so stemming removes the affixes at both ends of the word to get its base form. In contrast to lemmatization, which generates proper citation forms. **Gold stems** (and gold patterns and roots) are pre-annotated and can be used while training only, ideally as auxiliary tasks. **Greedy stems** are heuristic, and can be used both at training and runtime. Greedy stems are obtained through greedily matching the longest possible affixation sequence from both sides of the surface form, assuming the availability of a set of the potential affixation sequences in the language. The affixes set is inexpensive to obtain through minimal linguistic annotation, or can be obtained directly from the training data if available. The greedily matched sequences are removed from the sides of the word, and the rest is returned as the greedy stem. Table 1 shows a few examples. In our case, the training dataset contains the gold stems, which we used to extract the affixation sequences. In our training data, greedy stems matched gold stems about 65% of the time. Despite being crude, this approach should work as long as this method results in consistent stemming behavior. Since our data does not have gold templatic roots, we approximate the roots and patterns through a simple orthographic variation (Eskander et al., 2013). **Orthographic roots** (henceforth *roots*) are obtained through further abstracting the stem (gold or greedy), by removing the vowel letters. **Orthographic patterns** (henceforth *patterns*) are obtained by removing root letters from the stem of the surface word, and keeping affixes.

Morfessor Base Words Morfessor (Creutz and Lagus, 2005) is an unsupervised morphological segmentation tool, based on a probabilistic generative model, designed to handle languages with mainly concatenative morphologies. Morfessor generates morpheme-like segmentations, which often resemble linguistic morpheme segments. The Morfessor segmentation does not specify the base word explicitly, we therefore consider the least frequent segment in the detected word segments (relative to all the segments in the model) as the base word. This is under the assumption that base words are less frequent than affixes. In MSA, the base words from Morfessor matched the gold stems about %68 of the time.

Architecture We use a similar architecture to Lematus (Bergmanis and Goldwater, 2018) as a baseline. The model is based on a context-aware character-level sequence-to-sequence architecture, where context is modeled through a sliding window around the target word. Lematus uses a fixed 20-character window for each side. However, our experiments show that respecting the word-boundaries, by looking at the characters of the n words before and after the target word, provides better results. We use two LSTM layers for both the encoder and decoder (bidirectional for the encoder). We use Luong attention (Luong et al., 2015) over the encoder outputs h_i, and use the last encoder output as the initial state for the decoder.

Conditioning on the Subwords at the Encoder We condition on the different subwords through concatenating their embedding vector with the input character embeddings. Each character embedding $\mathbf{c}_i$ is replaced by the concatenation $[\mathbf{c}_i; \mathbf{w}_j]$ before being fed to the encoder, where $\mathbf{w}_j$ is the subword embedding for the word in which character i appears in. These features are incorporated at training time and runtime, and do not rely on expensive gold data. This is why we use the greedy stems, patterns, and roots here, instead of the gold subwords. The embedding vectors for the different subwords can be learnt as part of the end-to-end system, or pretrained using a large external corpus. For pretraining, we apply the Morfessor and greedy stemming approaches (repeated for the roots and patterns) to a large external corpus, then learn the embeddings using Word2Vec (Mikolov et al., 2013). We use the same corpus to learn the FastText embeddings.

Multitask Learning at the Decoder The gold stems, patterns, and roots can also be learnt jointly with the lemmatization task, as auxiliary tasks. In this model the encoder is shared between lemmatization and the other tasks, with the same input and parameters of the Bi-LSTM network. The different tasks (gold subwords) would have separate decoders specific to each task. The gold subword tasks are also modeled on the character-level, like the lemmatization task. The loss is the average of the individual decoder losses, which are based on minimizing cross entropy H for the decoder of each feature f, where F is the set of word-level features we model (gold stems, roots, or patterns), in addition to lemmatization:

$$H(\hat{y}, y) = \frac{1}{|F|} \sum_{f \in F} H_f(\hat{y}, y)$$

4 Experiments and Results

We work with two Arabic variants, MSA and EGY, as test cases. Arabic is a morphologically-rich language, with many affixes that result in a large number of inflected forms for each word.

Datasets We use the words and lemmas from the Penn Arabic Treebank (PATB parts 1,2, and 3) (Maamouri et al., 2004), of about 503K training tokens (of which 63K are for development), for MSA, and the ARZ corpus (Maamouri et al., 2012), parts 1 through 5, of about 133K training tokens for EGY (of which 21K are for development). We follow the data splits recommended by Diab et al. (2013) for both. We use the LDC's Gigaword corpus (Parker et al., 2011) to train the pretrained MSA embeddings, and the BOLT Arabic Forum Discussions corpus (Tracey et al., 2018) for the EGY embeddings.

Baselines We use several baselines to highlight different aspects of the contributions. The first baseline is based on a simple maximum likelihood estimation (MLE) approach. We use the training data to identify the most common lemma for each surface form without considering the context, and use this word-lemma mapping for inference. At inference time we return the mapped lemma if the word is seen in the training

Model	MSA	EGY
Maximum Likelihood Estimation (MLE)	90.6	77.5
LSTM LM (Zalmout and Habash, 2017)	92.5	82.8
Seq2seq+attention	93.5	76.6
Limited context seq2seq+attention* (baseline)	94.1	81.8
+ Morfessor base words	95.0	82.0
+ Greedy roots	95.1	82.2
+ Greedy stems	95.1	81.9
+ Greedy patterns	94.5	80.8
+ Pretrained n-grams (FastText)	**95.4**	**83.3**
+ Pretrained Morfessor base words	95.2	82.4
+ Pretrained greedy roots	95.3	82.6
+ Pretrained greedy stems	**95.4**	82.8
+ Pretrained greedy patterns	94.9	81.8
+ Multitask learning with predicted roots	95.0	81.9
+ Multitask learning with predicted stems	94.5	81.5
+ Multitask learning with predicted patterns	94.0	81.2
Gold Stems (oracle)	95.6	83.6

Table 2: Results for the various baselines and models (accuracy). *The limited context seq2seq with attention model is the main baseline in this work.

Dialect	Model	Seen		Unseen	
		Words	Lemmas	Words	Lemmas
MSA	% of train data	92.0%	93.8%	8.0%	6.2%
	MLE	96.9	92.4	19.5	65.0
	Baseline	96.5	95.9	67.4	67.3
	Best model	**97.3**	**96.9**	**73.1**	**71.9**
EGY	% of train data	84.6%	94.1%	15.4%	5.9%
	MLE	**90.4**	81.2	8.2	22.7
	Baseline	88.4	86.4	48.9	17.2
	Best model	89.4	**87.5**	**52.7**	**23.9**

Table 3: Error analysis results.

data, and back-off to the word itself if the word is unseen. This approach has several limitations, but could nevertheless serve as an assessment of the difficulty of the lemmatization task.

We also use Lematus (Bergmanis and Goldwater, 2018) as a baseline for the sequence-to-sequence approach. Lematus uses a sliding window of a fixed number of characters around the target word, and passes the context and target word into a character-based sequence-to-sequence model. However, instead of the character-based context, we use a word-level context of two words before and after the target word, which we found to provide better results. We use this as the main baseline in this work, where the various models are compared against it. We also report the results of a selection-based lemmatization model (Zalmout and Habash, 2017), using an LSTM-based language model with access to rich external morphological analyzers (which makes it not directly comparable to our model).

Experimental Setup We use layers of size 400 for both the encoder and decoder, dropout value of 0.4, fixed sampling probability of 0.4 (Bengio et al., 2015), word embedding size of 250, and character embedding size of 100. We train the models for 50 epochs, and use the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0.0005. We use beam decoding with beam size of 5. We only use the subwords that are associated with at least two words, with a special token for the others, which eliminates many noisy subwords. We use a context window of three words around the target word.

Results and Discussion The results in Table 2 show that the generic FastText embeddings outperform all of the other subword features, in both MSA and EGY, with a significant improvement over the baseline. The morphological subwords (stems, roots, and patterns), in addition to the Morfessor base words, whether with learnt or pretrained embeddings, all show improvements over the baseline, but still lag behind or match the FastText results. The multitask learning architecture, modeling the gold subwords as auxiliary tasks, also shows limited improvement over the baseline, or does slightly worse for EGY. This suggests that using the simple and generic FastText embeddings provides the same or better results than more sophisticated subword identification approaches. We also tried combinations of the subwords, like FastText embeddings with other morphological subwords. The combinations did not result in any significant accuracy improvement. The results further show that our models outperform the language-modeling-based approaches, which use expensive external morphological analyzers.

Oracular Experiments The datasets that we work with, for both MSA and EGY, include gold stem annotations for all words. To be able to fully assess the maximum potential accuracy gain of utilizing morphological subwords at the lemmatization task, we use the gold stems for an oracular experiment. We condition on the gold stems just like we did for the greedy subwords and Morfessor base words. The resulting accuracy represents the upper limit when it comes to utilizing stems through our architecture. Table 2 shows the result for the oracular experiment, compared with the all the other models and baselines. The gold stems provide only a slight improvement over the FastText embeddings, for both MSA and EGY. This indicates that using inexpensive unsupervised embeddings provides a competitive result compared to much more expensive supervised annotations, that are difficult to obtain at runtime.

Error Analysis Table 3 shows the baselines and best system results when handling seen and unseen words/lemmas in the training data. The results indicate significant accuracy gains for our models compared to the baselines in both. Moreover, the model seems to be able to handle the cases where both the word and lemma are not seen in the training data. The word *wAlmwrdyn* “and the suppliers”, for example, is not in the training data, nor its corresponding lemma, *muwar~id* “supplier”. The model was nevertheless still able to lemmatize it correctly in context.

We can also observe in Table 3 that the MLE baseline is very competitive in the seen words and lemmas. We therefore experiment with using the MLE model for the seen words in the training data, and backing off to our sequence-to-sequence model for the unseen words. The results of this setup are presented in Table 4. This approach results in large gains for EGY, but not for MSA. This is probably due to the much larger training dataset available for MSA, where the model can generate more accurate lemmas overall for both seen and unseen words and lemmas.

Model	MSA	EGY
Best model (FastText embeddings)	**95.4**	83.3
MLE + best model	95.0	**84.1**

Table 4: Results for using MLE for the seen words, and the best model as backoff for the unseen words.

We also analyzed the errors in a sample of 1,000 tokens from the development set, lemmatized using both the models with pretrained stems and FastText subwords. We found that many of the errors in the model using FastText embeddings tend to be correct lemmas, but not suitable for the word in its context. Whereas the errors in the model with pretrained stems tend to be invalid lemmas altogether.

5 Conclusions

In this paper we presented a character-level sequence-to-sequence lemmatization model, with access to additional subword features. Our experiments show that simply conditioning on the pretrained FastText word embeddings at the encoder, in addition to the character embeddings, improves accuracy significantly. FastTex embeddings also provide the same or better results compared to sophisticated morphological subword identification approaches. Our models outperform a strong baseline, which has access to expensive external resources, and can handle ambiguous and unseen words.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1171–1179. MIT Press.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with Lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1391–1400.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. *LREC 2008*, pages 2362–2367.

Mathias Creutz and Krista Lagus, 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*, chapter A. Publications in computer and information science. Report A. Helsinki University of Technology, Finland.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Timothy Dozat and Christopher D Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 484–490.

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic Extraction of Morphological Lexicons from Morphologically Annotated Corpora. In *Proceedings of tenth Conference on Empirical Methods in Natural Language Processing*.

Nerea Ezeiza, Iñaki Alegria, José María Arriola, Rubén Urizar, and Itziar Aduriz. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 380–384. Association for Computational Linguistics.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in SMT. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674. Association for Computational Linguistics.

Jenna Kanerva, Filip Ginter, and Tapio Salakoski. 2019. Universal lemmatizer: A sequence to sequence model for lemmatizing universal dependencies treebanks. *arXiv preprint arXiv:1902.00972*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Daniel Kondratyuk, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. Lemmatag: Jointly tagging and lemmatizing for morphologically rich languages with BRNNs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928.

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

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Mohamed Maamouri, Sondos Krouna, Dalila Tabessi, Nadia Hamrouni, and Nizar Habash. 2012. Egyptian Arabic Morphological Annotation Guidelines.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with Lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2011. Arabic Gigaword Fifth Edition. LDC catalog number No. LDC2011T11, ISBN 1-58563-595-2.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *In Proceedings of LREC*, Reykjavik, Iceland.

Tobias Pütz, Daniël De Kok, Sebastian Pütz, and Erhard Hinrichs. 2018. Seq2seq or perceptrons for robust lemmatization. an empirical examination. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018), December 13–14, 2018, Oslo University, Norway*, pages 193–207. Linköping University Electronic Press.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *ACL 2008: The Conference of the Association for Computational Linguistics; Companion Volume, Short Papers*, Columbus, Ohio, June. Association for Computational Linguistics.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *LREC*, pages 1–263. Lisbon.

Djamé Seddah, Grzegorz Chrupała, Özlem Çetinoğlu, Josef Van Genabith, and Marie Candito. 2010. Lemmatization and lexicalized statistical parsing of morphologically rich languages: the case of French. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 85–93. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, volume 1, pages 83–91.

Jennifer Tracey, Haejoong Lee, Stephanie Strassel, and Safa Ismael. 2018. BOLT Arabic Discussion Forum Source Data. LDC catalog number LDC2018T10.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713, Copenhagen, Denmark, September. Association for Computational Linguistics.

Nasser Zalmout and Nizar Habash. 2019. Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1775–1786.