Natural Language Engineering (2022), **28**, pp. 223–248
doi:10.1017/S1351324920000455

ARTICLE

Unsupervised Arabic dialect segmentation for machine translation

Wael Salloum* and Nizar Habash

AI Research Department, Mendel.ai, San Jose, CA, USA
*Corresponding author. E-mail: wael@ccls.columbia.edu

(Received 12 November 2018; revised 19 May 2020; accepted 22 May 2020; first published online 23 September 2020)

Abstract
Resource-limited and morphologically rich languages pose many challenges to natural language processing tasks. Their highly inflected surface forms inflate the vocabulary size and increase sparsity in an already scarce data situation. In this article, we present an unsupervised learning approach to vocabulary reduction through morphological segmentation. We demonstrate its value in the context of machine translation for dialectal Arabic (DA), the primarily spoken, orthographically unstandardized, morphologically rich and yet resource poor variants of Standard Arabic. Our approach exploits the existence of monolingual and parallel data. We show comparable performance to state-of-the-art supervised methods for DA segmentation.

Keywords: Machine translation; Morphology; Arabic dialects; Unsupervised learning

1. Introduction

Resource-limited, morphologically rich languages pose many challenges to natural language processing (NLP) tasks. The highly inflected surface forms of these languages inflate the vocabulary size and increase sparsity in an already scarce data situation. NLP in general, and machine translation (MT) in particular, can greatly benefit from unsupervised learning approaches to vocabulary reduction, such as unsupervised morphological segmentation. Dialectal Arabic (DA) is an iconic representative of such languages: it has limited parallel and task-specific labeled data, and its large vocabulary is the result of its rich inflectional morphology and unstandardized spontaneous orthography. While these dialects have been historically primarily spoken, they are quite frequently written in social media today. The scarcity of DA parallel and labeled text is more pronounced when considering the large number of dialects and subdialects, the varying levels of dialectness and code switching, the diversity of domains and genres, and the timespan of the collected text. Hence, the need for unsupervised learning solutions to vocabulary reduction that use a more sustainable and continuously fresh source of training data arises, namely monolingual data. In this work, we utilize huge collections of monolingual Arabic text along with limited DA–English parallel data to improve the quality of DA-to-English MT.

We propose an unsupervised learning approach to morphological segmentation consisting of three successive systems. The first system uses word embeddings learned from huge amounts of monolingual Arabic text to extract and extend a list of possible segmentation rules for each vocabulary word and scores these rules with an expectation maximization (EM) algorithm. The second system uses the learned segmentation rules in another EM algorithm to label select DA words in DA–English parallel text with the best segmentation choice based on the English alignments of the word segments. Finally, the third system implements a supervised segmenter by training an averaged structured perceptron (ASP) on the automatically labeled text. The three systems can be used independently for other purposes. We evaluate the performance of our segmenter intrinsically on a portion of the labeled text and extrinsically on MT quality.

2. DA challenges

Contemporary Arabic is a collection of varieties: Modern Standard Arabic (MSA), which has a standard orthography and is used in formal settings, and DAs, which are commonly used informally and with increasing presence on the web, but which do not have standard orthographies. There are several DA varieties which are identified primarily by geography, for example, Gulf Arabic, Levantine Arabic, Egyptian Arabic. (Habash 2010). DAs differ from MSA phonologically, morphologically, and to some lesser degree syntactically. The differences between MSA and DAs have often been compared to Latin and the Romance languages (Habash 2006).

On Arabic morphology and orthography. In the context of this article, we focus on two challenging areas for modeling Arabic: morphology and orthography. Morphologically, Arabic (MSA and DA) has a rich inflectional system, expressed both templatically and affixationally, and several classes of attachable clitics. For instance, the Levantine Arabic word وحيكتبوها *w+H+y-ktb-w+hA*[a] "and they will write it" has two proclitics (+ و *w+* "and" and + ح *H+* "will"), one prefix -ي *y-* "3rd person," one suffix و- *-w* "plural," and one pronominal enclitic ها+ *+hA* "it/her." Orthographically, Arabic is written with optional diacritics that specify short vowels, consonantal doubling, and the nunation morpheme (Habash 2010). The diacritics are almost never written outside of the domains of religious text, children's books, and non-native learners' textbooks. The almost universal absence of these diacritics together with the language's rich morphology lead to a high degree of ambiguity, for example, the Buckwalter Arabic Morphological Analyzer (BAMA) produces an average of 12 analyses per word (Buckwalter 2004). Moreover, some Arabic letters are often spelled inconsistently which leads to an increase in both sparsity (multiple forms of the same word) and ambiguity (same form corresponding to multiple words), for example, variants of Hamzated Alif, أ > or, إ <, are often written without their Hamza (ء '): ا *A*; and the Alif-Maqsura (or dotless Ya) ى Y and the regular dotted Ya ي *y* are often used interchangeably in word final position (El Kholy and Habash 2010). Furthermore, in the case of DA, there are no orthographic standards; as such, there is a huge problem with orthographic variants causing model sparsity, for example, Habash *et al.* (2018) report on a single Egyptian word encountered online in 27 spellings.

NLP challenges. Arabic's complex morphology and high degree of ambiguity are usually handled through preprocessing using tools for analysis, disambiguation, and tokenization (Habash and Rambow 2005; Diab *et al.* 2007; Pasha *et al.* 2014; Abdelali *et al.* 2016; Khalifa *et al.* 2016; Zalmout and Habash 2017a). The lack of standard orthographies for the dialects and their numerous varieties cause new challenges to NLP (Habash *et al.* 2012c; Eskander *et al.* 2013; Habash *et al.* 2018). DAs are rather impoverished in terms of available tools and resources compared to MSA, for example, there is very little parallel DA–English corpora and almost no MSA–DA parallel corpora. The number and sophistication of morphological analysis and disambiguation tools in DA is very limited in comparison to MSA (Duh and Kirchhoff 2005; Habash and Rambow 2006; Abo Bakr *et al.* 2008; Habash *et al.* 2012a; Khalifa *et al.* 2017) MSA tools cannot be effectively used to handle DA: Habash and Rambow (2006) report that less than two-thirds of Levantine verbs can be analyzed using an MSA morphological analyzer; and Habash *et al.* (2012a) report that 64% of Egyptian Arabic words are analyzable using an MSA analyzer.

In this article, we present an approach that minimizes the requirement of developing morphological preprocessing tools for DA through exploiting readily available monolingual corpora and

[a] Arabic transliteration is in the Buckwalter scheme (Habash *et al.* 2007).

(limited but relatively easy to obtain) parallel corpora. Our approach is easily reusable for other languages and dialectal varieties, and it can be applied to a variety of genres and domains that may be relatively poor even for standardized languages.

3. Related work

3.1 Dialectal Arabic natural language processing

Extending MSA resources. Much work has been done in the context of MSA NLP (Habash 2010). Specifically for Arabic-to-English statistical machine translation (SMT), the importance of tokenization using morphological analysis has been shown by many researchers (Habash and Sadat 2006). For the majority of Arabic dialects, dialect-specific NLP resources are nonexistent or in their early stages. Several researchers have explored the idea of exploiting existing MSA-rich resources to build tools for DA NLP, for example, Chiang *et al.* (2006) built syntactic parsers for DA trained on MSA treebanks. Such approaches typically expect the presence of tools/resources to relate DA words to their MSA variants or translations. Given that DA and MSA do not have much in terms of parallel corpora, rule-based methods to translate DA to MSA or other methods to collect word pair lists have been explored. For example, Abo Bakr *et al.* (2008) introduced a hybrid approach to transfer a sentence from Egyptian Arabic into MSA. This hybrid system consisted of a statistical system for tokenizing and tagging, and a rule-based system for constructing diacritized MSA sentences. Moreover, Al-Sabbagh and Girju (2010) described an approach of mining the web to build a DA-to-MSA lexicon. In the context of DA-to-English SMT, Riesa and Yarowsky (2006) presented a supervised algorithm for online morpheme segmentation on DA that cut the out-of-vocabulary words (OOVs) by half.

DA morphological analysis. By comparison to MSA, only a few efforts have targeted DA morphology (Kilany *et al.* 2002; Habash and Rambow 2006; Abo Bakr *et al.* 2008; Salloum and Habash 2011; Mohamed *et al.* 2012; Habash *et al.* 2012a; Hamdi *et al.* 2013; Khalifa *et al.* 2017; Samih *et al.* 2017a; Samih *et al.* 2017b; Zalmout and Habash 2019). Efforts for modeling DA morphology generally fall into two camps. First are solutions that focus on *extending MSA tools to cover DA phenomena*. For example, Abo Bakr *et al.* (2008) and Salloum and Habash (2011) extended the BAMA/SAMA databases (Buckwalter 2004; Graff *et al.* 2009) to accept DA prefixes and suffixes. These solutions are fast and cheap to implement but are limited in their modeling of DA linguistic phenomena. The second camp is interested in *modeling DA directly*. The earliest effort on Egyptian that we know of is the Egyptian Colloquial Arabic Lexicon (Kilany *et al.* 2002). This resource was the base for developing the CALIMA Egyptian morphological analyzer (Habash *et al.* 2012a). Another effort is the work by Habash and Rambow (2006) which focuses on modeling DAs together with MSA using a common multi-tier finite-state machine framework. Eskander *et al.* (2013) presented a method for automatically learning inflectional classes and associated lemmas from morphologically annotated corpora. Hamdi *et al.* (2013) takes advantage of the closeness of MSA and its dialects to build a translation system from Tunisian Arabic verbs to MSA verbs. Eskander *et al.* (2016) presented an approach to annotating words with a conventional orthography, a segmentation, a lemma, and a set of features. They use these annotations to predict unseen morphological forms, which are used, along with the annotated forms, to create a morphological analyzer for a new dialect. The second approach to modeling Arabic dialect morphology results in better quality morphological analyzers compared to the shallow techniques presented by the first camp. However, they are expensive and need a lot more resources and efforts. Furthermore, they are harder to extend to new dialects since they require annotated training data and/or handwritten rules for each new dialect.

Morphological tokenization for MT. Reducing the size of the vocabulary by tokenizing morphologically complex words proves to be very beneficial for any statistical NLP system in general, and MT in particular. Sadat and Habash (2006) explored a number of tokenization schemes for Arabic when translating to English (one scheme at a time). However, Zalmout and Habash (2017b) experimented with different tokenization schemes for different words in the same Arabic text. Their work showed that these different target languages (English, French, Spanish, Russian, and Chinese) require different Arabic (source) language tokenization schemes. It also showed that combining different tokenization options while training and decoding the SMT system improves the overall performance. The work we present in this article is similar to their work in that the segmentation of a word is influenced by the target language (in our case English) and this can change if the target language changes. We differ from that work in that we do not use predetermined tokenization schemes or combine them; instead, we learn to segment words, and that segmentation is dependent on the word itself and its context.

3.2 MT of dialects

Dialects present many challenges to MT due to their spontaneous, unstandardized nature, and the scarcity of their resources. In this section, we discuss different approaches to handle dialects in the context of MT.

MT for closely related languages. Using closely related languages has been shown to improve MT quality when resources are limited. Hajič *et al.* (2000) argued that for very close languages, for example, Czech and Slovak, it is possible to obtain a better translation quality using simple methods, such as morphological disambiguation, transfer-based MT, and word-for-word MT. Zhang (1998) introduced a Cantonese–Mandarin MT that uses transformational grammar rules. In the context of Arabic dialect translation, Sawaf (2010) built a hybrid MT system that uses both statistical and rule-based approaches for DA-to-English MT. In his approach, DA is normalized into MSA using a dialectal morphological analyzer. In this work, we present a rule-based DA–MSA system to improve DA-to-English MT. Our approach used a DA morphological analyzer (ADAM) and a list of handwritten morphosyntactic transfer rules. This use of "resource-rich" related languages is a specific variant of the more general approach of using pivot/bridge languages (Kumar *et al.* 2007; Utiyama and Isahara 2007). In the case of MSA and DA variants, it is plausible to consider the MSA variants of a DA phrase as monolingual paraphrases (Callison-Burch *et al.* 2006; Du *et al.* 2010). Also related is the work by Nakov and Ng (2011), who use morphological knowledge to generate paraphrases for a morphologically rich language, Malay, to extend the phrase table in a Malay-to-English SMT system.

DA-to-English MT. Two approaches have emerged to alleviate the problem of DA–English parallel data scarcity: using MSA as a bridge language (Sawaf 2010; Salloum and Habash 2011; Salloum and Habash 2013; Sajjad *et al.* 2013) and using crowdsourcing to acquire parallel data (Zbib *et al.* 2012). Sawaf (2010) and Salloum and Habash (2013) used hybrid solutions that combine rule-based algorithms and resources such as lexicons and morphological analyzers with statistical models to map DA to MSA before using MSA-to-English MT systems. Sawaf (2010) classified words into 15 dialects plus MSA as part of the mapping process to MSA. Sawaf (2010) and Salloum and Habash (2013) achieve around 1.6% and 1.4% BLEU point increases, respectively, over a baseline that does not map DA to MSA.

Pivoting approaches. Sawaf (2010) built a hybrid MT system that uses both statistical and rule-based approaches to translate both DA and MSA to English. In his approach, DA is normalized into MSA using a character-based normalizer, MSA and DA-specific morphological analyzers, and a class-based n-gram language model to classify words into 16 dialects (including MSA). These components produce a lattice annotated with probabilities and morphological features (part-of-speech, stem, gender, etc.), which is then n-best decoded with character-based and word-based, DA and MSA language models. The 1-best sentence is then translated to English with the hybrid MT system. He also showed an improvement of up to 1.6% BLEU by processing the SMT training data with his technique. Salloum and Habash (2011; 2012) and Sajjad *et al.* (2013) applied character-level transformation to reduce the gap between DA and MSA. This transformation was applied to Egyptian Arabic to produce EGY data that looks similar to MSA data. They reduced the number of OOV words and spelling variations and improved translation output.

Cheaply obtaining DA–English parallel data. Zbib *et al.* (2012) demonstrated an approach to cheaply obtain DA–English data via Amazon's Mechanical Turk (MTurk). They created a DA–English parallel corpus of 1.5M words and used it along with a 150M MSA–English parallel corpus to create the training corpora of their SMT systems. They found that the DA–English system outperforms the DA+MSA–English even though the ratio of DA data size to MSA data size is 1:100. They concluded that simple vocabulary coverage is not sufficient and the domain mismatch is a more important problem. Similar to Zbib *et al.* (2012), in this work, we add 3.5M words DA–English parallel data to their 1.5M words data and we build three SMT systems such as DA–English, MSA–English, and DA+MSA–English where the ratio of DA data size to MSA data size becomes 1:10.

3.3 *Supervised learning approaches to morphological segmentation*

Supervised learning techniques, like MADA, MADA-ARZ, and AMIRA (Habash and Rambow 2005; Diab *et al.* 2007; Habash *et al.* 2013; Pasha *et al.*, 2014), have performed well on the task of morphological tokenization for Arabic MT. They require handcrafted morphological analyzers, such as SAMA (Graff *et al.* 2009) or CALIMA (Habash *et al.* 2012b) in addition to treebanks to train tokenizers. Using the analyzer, the system turns an input sentence into a lattice of analyses that can be decoded using a context-sensitive model trained on the manually annotated treebank. This is expensive and time-consuming and thus hard to scale to different dialects. Alongside the work on DA morphological tokenization, there has been work on dialectal segmentation. Most recently, some neural models have been achieving good performance without the need to create linguistically motivated analyzers. For example, in Samih *et al.* (2017a, b), the authors propose a dialectal segmentation model using a deep learning-based, character-level sequence tagger. The model uses character embeddings fed into a bidirectional LSTM layer followed by a CRF layer that computes the probability distribution over the output tags. The model rivals MADAMIRA-EGY on Egyptian and outperforms an SVM ranker on Levantine, Gulf, and Maghribi dialects. Most recently, Zalmout and Habash (2019) presented a joint multi-feature cross-dialectal morphological disambiguation model for MSA and Egyptian Arabic using adversarial training for cross-dialectal morphological knowledge transfer. Their models achieve state-of-the-art results for both Arabic variants.

3.4 *Unsupervised learning approaches to morphological segmentation*

Given the wealth of unlabeled monolingual text freely available on the Internet, many unsupervised learning algorithms (Creutz and Lagus 2002; Stallard *et al.* 2012; Narasimhan *et al.* 2015) took advantage of it and achieved outstanding results, although not to a degree where they outperform supervised methods, at least on DA to the best of our knowledge. Traditional approaches to unsupervised morphological segmentation, such as MORFESSOR (Creutz and Lagus 2002; Creutz and Lagus 2007), use orthographic features of word segments (prefix, stem, and suffix). However, many researchers worked on integrating semantics in the learning of morphology (Schone and Jurafsky 2000; Narasimhan *et al.* 2015), especially with the advances in neural network-based distributional semantics (Narasimhan *et al.* 2015). Most recently, Erdmann *et al.* (2019) presented a linguistically motivated alternative to greedy or other unsupervised methods, requiring only minimal language-specific input with large unannotated corpora. In their evaluations, they consistently outperform competitive unsupervised baselines and approach the performance of state-of-the-art models such as MADAMIRA (Pasha *et al.* 2014) and Farasa (Abdelali *et al.* 2016).

In this work, we leverage the use of both approaches. We implement an unsupervised learning approach to automatically create training data which we use to train supervised algorithms for morphological segmentation. Our approach incorporates semantic information from two sources, Arabic (through monolingual data) and English (through parallel data), along with linguistic features of the source word and its target segments to learn a morphological segmenter.

4. Our approach to unsupervised segmentation

Inspired by the supervised tokenization approaches discussed above (Habash and Rambow 2005; Habash *et al.* 2013; Pasha *et al.* 2014), our approach to *unsupervised* morphological segmentation consists of three components with strong parallels to the supervised methodology:

1. *Monolingual identification of segmentation rules.* In this component, we build a system that learns Arabic segmentation rules from monolingual text using distributional semantics (Section 5). This component is analogous to a morphological analyzer in that it produces *out-of-context* segmentation options.
2. *Alignment guided segmentation choice.* In this component, we build a system that creates synthetic segmentations of Arabic words in a way that optimizes their alignment with English words in a parallel text (Section 6). This component effectively incorporates English semantics in choosing the best *in-context* segmentation of an Arabic word in a sentence. This is strongly analogous to manual human annotation. However, it is cheaper in that it does require special training to do annotation for the specific NLP task of segmentation. It is also more flexible and tunable, since different approaches to MT or different target languages may require optimizing for different segmentation choices.
3. *Supervised segmentation using synthetic data.* Starting from the automatically segmented (labeled) data created by the previous component, we train a tagger that learns to score all possible segmentations for a given word in a sentence (Section 7).

One challenge to this approach is that the automatic labeling of words will introduce errors that will affect the quality of the supervised segmenter. To reduce the number of errors in the automatically labeled data, we only label words when the system has a high confidence in its decision. This will result in many unlabeled words in a given sentence that raises another challenge to the supervised segmenter which we solve by modifying the training algorithm.

The underlying assumption of this approach is that if the unsupervised labeling process does not cover all words in the vocabulary, the supervised segmenter will learn to generalize to the missed words and OOVs. We evaluate this approach on two Arabic dialects: Egyptian and Levantine. The next three sections discuss the three systems used in this approach and present the experimental setup, examples, and discussion.

5. Monolingual identification of segmentation rules

Given a word out of context, we would like to generate a ranked list of possible segmentations of it. For example, the word ما تجوزت *mAtjwzt* "I-did-not-marry" should be ideally segmentable into ما تجوزت *mA tjwzt*, ما تجوز ت or *mA tjwz t* but not ماتج وزت *mAtj wzt* (a nonsensical segmentation). Our approach to learn segmentation rules from monolingual data consists of three steps: word clustering, rule learning, and rule scoring.

5.1 Word clustering based on word embeddings

We learn word vector representations (word embeddings) from huge amounts of monolingual Arabic untokenized text using Word2Vec (Mikolov *et al.*, 2013). For every Arabic word *x*, we then compute the closest N words using cosine distance (with cosine distance above threshold D_a). We consider these N words to be x's semantic cluster. Every cluster has a source word. It is important to mention that a word x might appear in y's cluster but not vice versa.

After a manual error analysis of the data, we picked a high cosine distance threshold $D_a = 0.35$, since we need only words that we have high confidence in their belonging to a cluster in order to produce high-quality segmentation rules. This high threshold results in many words having small or even empty clusters. In *small clusters*, the main word will not have enough segmentation rules, limiting the possibility of identifying the optimal segmentation in the next step. For example, the word ماتجوزت *mAtjwzt* "I-did-not-marry" might have ماتجوز *mAtjwz* "did-not-marry" but not اتجوز *Atjwz* "married," which is the desired stem. We attempt to solve this problem with a rule expansion algorithm discussed in Section 5.2. The *empty clusters* happen when the closest word to the cluster's main word is beyond the distance threshold. This means that we will not have any labeled examples for this word; hence, it will be an OOV word for the supervised segmenter. We address this issue later on by designing the segmenter so that it generalizes to unseen words.

Even with this high threshold, many words end up with *very large clusters* due to Word2Vec putting thousands of certain types of words very close in the vector space (e.g., proper names, words that appeared only few times in the training data). This adds noise to the rule scoring training data discussed later. We solve this problem by deciding on a maximum cluster size $N = 200$.

5.2 Rule learning

Rule extraction. We extract segmentation rules from the clusters learned previously. For every word x, for every word y in x's cluster where y is a substring of x, we generate a segmentation rule $x \rightarrow p + y - q$ where y is the stem, $p+$ is the prefix (the substring of x before y starts), and $-q$ is the suffix (the substring of x after y ends). A rule might have an empty prefix or suffix denoted as P+ and -Q, respectively. If y happens to appear at different indices inside x, we generate multiple rules (e.g., x is "hzhzt" and y is "hz" we produce "hzhzt $\rightarrow$ P+ hz -hzt" and "hzhzt $\rightarrow$ hz+ hz -t"). We define a function $dist(x \rightarrow y)$ as the cosine distance between words x and y if there is a rule from x to y, equal to 1 if $x = y$, and 0 otherwise:

$$dist(x \rightarrow y) = \begin{cases} cosineDistance(x, y), & \text{if } \exists\, x \rightarrow p\text{+ } y \text{ -}q \\ 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Rule expansion. Given the high cluster admission threshold, we consider expanding the rule set by adding further segmentations. We build an acyclic directed graph from all the extracted rules where words are nodes and rules are edges. Since a rule is always directed from a longer word to a shorter one, the graph will not have cycles and will not have very long paths. Figure 1 shows a partial segmentation graph built from some rules that lead to the word اتجوز *Atjwz* "I marry/he married." We then expand the rules by generating a rule $x \rightarrow y$ for every node x that has a path to node y in the graph. To do so, we recursively scan the graph starting from leaves (words that cannot be segmented) all the way back to the beginning of the graph and add a rule $x \rightarrow y$ to the graph if there are two rules $x \rightarrow w$ and $w \rightarrow y$ where w is a word. The recursive scan insures that we fully expand w and add its outgoing edges to the graph before we expand x. It also allows us to compute a *confidence score* $conf(x \rightarrow y)$ that starts with the cosine distance between x and y and will increase the more paths we find between them:

$$conf(x \rightarrow y) = dist(x \rightarrow y) + \sum_{w} conf(x \rightarrow w) \times conf(w \rightarrow y) \tag{2}$$

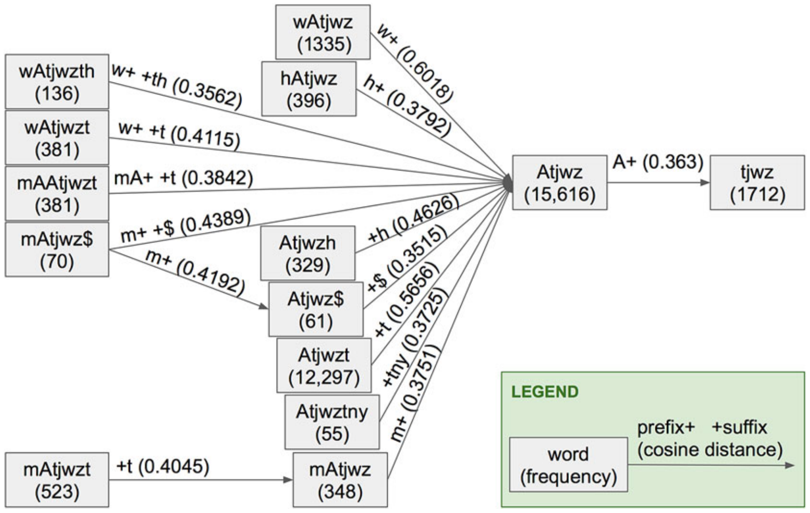

Figure 1. Example of a segmentation graph that leads to the word *Atjwz* "I marry/he married." The frequencies of the words are enclosed in parenthesis inside the nodes. The cosine distance and split affixes are on the arcs.

Projected frequency. The main reason for segmentation is to reduce the vocabulary size and thus increase word frequencies which improves the quality of any subsequent statistical system. However, chopping affixes off a word (as opposed to clitics) may affect the integrity of the word, for example, it may slightly change the meaning of the word or may cause it to align to more general words (e.g., segmenting "parking" to "park -ing" in English may negatively affect alignment depending on the foreign language). Additionally, every time we make a segmentation decision, we may introduce errors. Therefore, it is important to know at what point we do not need to segment a word anymore. To do so, we consider word frequencies as part of scoring the rules because frequent words are likely to be aligned and translated correctly to their inflected English translations without the need for segmentations. For example, in Figure 1, we might not want to segment "Atjwz" to "tjwz" considering its high frequency and the number and frequencies of words that lead to it. We define a *projected frequency score* of a word as:

$$pf(y) = \sum_{x \in V} conf(x \to y) \times log(freq(x)) \tag{3}$$

where x is any word in the vocabulary V and $log(freq(x))$ is the logarithm of x's frequency. We use logarithms to smooth the effect of frequent words on the final score to better represent y as a hub for many words. Note that $conf(y \to y) = 1$ and thus we include $log(freq(y))$ in the score.

Word-to-stem score. Given the projected frequency scores, we compute the ratio $pf(y)/pf(x)$ that represents the gain we get from segmenting x to y. For example, if x is frequent and has many words that lead to it, and y gets most of its projected frequency from x, then the ratio will be small. But if x is infrequent (i.e., not a hub) and y has a high pf score through other sources, then the ratio will be high. Now, we can compute the final *word-to-stem (a2s) score*, which we will be using next, as follows:

$$a2s(a \to s) = conf(a \to s) \times pf(s)/pf(a) \tag{4}$$

Fully reduced words. All the rules extracted so far will segment a word to a shorter word with at least one affix. The automatic annotations need to have some examples where words do not get segmented in order for the segmenter to learn such cases. Therefore, we need to identify a list of words that cannot be segmented and thus produce rules that transform a word to itself with no affixes. Such rules will be of the form $x \to$ P + x -Q. The list does not have to be complete; it

Algorithm 1 Affix–stem joint probability estimation.

```
1:  // Initialization:
2:  $v(p, s) \leftarrow \frac{\sum_{a \to psq} a2s(a \to s)}{\sum_{a \to p's'q} a2s(a \to s')}$ for all $p, s$
3:  $u(q, s) \leftarrow \frac{\sum_{a \to psq} a2s(a \to s)}{\sum_{a \to ps'q'} a2s(a \to s')}$ for all $q, s$
4:  // Estimation:
5:  for $round := 1 \to MAX$ do
6:      // Collect counts:
7:      for each rule $a \to psq$ do
8:          $c_v(p, s) = \sum_{p'} \sum_{s'} v(p, s') \times v(p', s')$
9:          $c_u(q, s) = \sum_{q'} \sum_{s'} u(q, s') \times u(q', s')$
10:         $\delta \leftarrow a2s(a \to s) \times c_v(p, s) \times c_u(q, s)$
11:         $count_v(p, s) \mathrel{+}= \delta$
12:         $count_u(q, s) \mathrel{+}= \delta$
13:         $total \mathrel{+}= \delta$
14:     // Estimate joint probabilities:
15:     $v(p, s) \leftarrow count_v(p, s)/total$ for all $p, s$
16:     $u(q, s) \leftarrow count_u(q, s)/total$ for all $q, s$
17: // Calculate rule scores:
18: $score(a \to psq) = a2s(a \to s) \times v(p, s) \times u(q, s)$ for all rules $a \to psq$
```

just needs to be of high confidence. To generate the list, we consider words that appear on the target side of rules but never on the source side. We then reduce the list to only frequent stems (with over 3000 occurrences) that have at least three words that can be segmented to them, which gives us enough confidence as we have seen them in various contexts and they have appeared in at least three clusters but yet do not have any substrings in their own clusters. These thresholds are determined empirically. We compute the word-to-stem score for these fully reduced rules using this equation:

$$a2s(a \to a) = \sum_{x} conf(x \to a) \tag{5}$$

5.3 Rule scoring

Since a rule $a \to psq$ produces three segments: a stem s, a prefix p, and a suffix q, we define its score as the product of the word-to-stem score $a2s(a \to s)$, the joint probability of the prefix and the stem $v(p, s)$, and the joint probability of the suffix and the stem $u(q, s)$:

$$score(a \to psq) = a2s(a \to s) \times v(p, s) \times u(q, s) \tag{6}$$

Affix correlation. To estimate the correlation between a prefix p and a prefix p', we iterate over all stems s' and compute

$$corr_{pref}(p', p) = \sum_{s'} v(p, s') \times v(p', s') \tag{7}$$

This score indicates the similarity in morphosyntactic behavior of these two prefixes. For example, the Egyptian prefix +ه *h*+ "will" and the MSA prefix +وس *ws*+ "and will" attach to present tense verbs; therefore, we would expect them to share many of the stems in the rules they appear in, which leads to a high correlation score.[b]

We similarly define suffix correlation as:

$$corr_{suff}(q', q) = \sum_{s'} u(q, s') \times u(q', s') \tag{8}$$

Affix–stem correlation. Using these affix correlation scores, we can estimate the correlation between a prefix p and a stem s by iterating over all prefixes p' that we have seen with s in a rule and summing their correlation scores with p:

$$c_v(p, s) = \sum_{p'} corr_{pref}(p', p) = \sum_{p'} \sum_{s'} v(p, s') \times v(p', s') \tag{9}$$

We similarly define suffix–stem correlation as:

$$c_u(q, s) = \sum_{q'} corr_{suff}(q', q) = \sum_{q'} \sum_{s'} u(q, s') \times u(q', s') \tag{10}$$

Affix–stem joint probabilities. Given affix–stem correlation scores, we can define the prefix–stem joint probability, $v(p, s)$, and the suffix–stem joint probability, $u(q, s)$, as follows:

$$v(p, s) = \frac{c_v(p, s)}{\sum_{p', s'} c_v(p', s')}, \qquad u(q, s) = \frac{c_u(q, s)}{\sum_{q', s'} c_u(q', s')} \tag{11}$$

To estimate the parameters in these *recursive* equations, we implement the EM algorithm shown in Algorithm 1 The initialization step uses only word-to-stem scores computed earlier, that is, it is equivalent to the first round in the following EM loop with the exception that $\delta \leftarrow a2s(a \rightarrow s)$. We found that running the EM algorithm for 50 rounds provides good results.

5.4 Experimental data

We use two sets of monolingual Arabic text: about 2B tokens from Arabic GigaWord Fourth Edition which is mainly MSA (Parker *et al.* 2009) and about 392M tokens of Egyptian text,[c] resulting in about 2.4B tokens of monolingual Arabic text used to train Word2Vec (Mikolov *et al.* 2013) to build word vectors. In this work, we did not have access to a sizable amount of Levantine text to add to the monolingual data. Access to Levantine text would help this task learn Levantine segmentation rules and thus hopefully improve the final system. We did not want to use the Levantine side of the parallel data to keep this system separate from the second system to avoid any resulting biases.

6. Alignment guided segmentation choice

In the previous section, we learned and scored segmentation rules for words out of context. In this section, we use these rules and their scores to learn *in-context segmentations* of words guided

[b]In practice, we iterate over the N stems with the highest $v(p, s')$ values for a prefix because some prefixes, like +و *w*+ "and," attach to tens of thousands of stems and that unnecessarily slows the algorithm. We found $N = 500$ to be fast and provide good results.

[c]The 392M Egyptian words were selected from the LDC catalog numbers: LDC2012E30, LDC2012E51, LDC2012E94, LDC2012E96, LDC2012E75, LDC2012E77, LDC2012E107, LDC2012E19, LDC2012E54, and LDC2012E17 (Al-Badrashiny *et al.* 2016).

by their English alignments in parallel text. The premise of this approach is that if we find enough Arabic words where we are confident in their segmentation choices in-context given the English translation, then we can use those segmentation choices as labeled data to train a supervised segmenter.

6.1 Approach

Unsupervised learning of word alignments from a parallel corpus is pretty much established. A ltool like Giza++ (Och and Ney 2003b) can be run on the Arabic–English parallel data to obtain many-to-many word alignments. That means, each Arabic word aligns to multiple English words and each English word aligns to multiple Arabic words. However, these algorithms look at the surface form without considering morphological inflections.

Our alignment algorithm concerns with aligning the internal structure of Arabic words (the rule segments) to their English translations. We start by running Giza++ on our Arabic–English parallel corpora to obtain initial, surface form alignments. We use Giza++ with default symmetrization (*grow-diag-final-and*) as part of the Moses toolkit pipeline for statistical MT (Koehn *et al.* 2007). Then, we consider one-to-many aligned pairs $\langle a_i, E_{a_i}\rangle$, where a_i is an Arabic word at position i and $E_{a_i} = (e_1, e_2, ..., e_{|E_{a_i}|})$ is the sequence of English words aligned to a_i ordered by their position in the English sentence. Since the Arabic side of the parallel data is unsegmented, the plethora of inflected words will dramatically extend the vocabulary size and the Zipfian tail of infrequent words, which will negatively affect parameter estimation in Giza++ resulting in many inaccurate alignments. To reduce the effect of this problem on our algorithm, we expand the definition of E_{a_i} to also include surrounding words of the words aligned to a_i by Giza++. The order of the English words is preserved. We model dropping words from E_{a_i} in our alignment model.

Given an aligned pair $\langle a_i, E_{a_i}\rangle$ where a_i has a set of segmentation rules $R = \{r : r = a_i \rightarrow g_1g_2g_3\}$, we estimate an *alignment probability* for every rule r based on aligning its segments to words in E_{a_i}. We, then, pick the rule with the highest probability, r^*, as the segmentation choice for a_i in that context. It is important to note that the context here is determined by the English translation instead of the surrounding Arabic words. Note that the rule itself has a score derived from the Arabic context of the word through word embedding. Therefore, if we incorporate the rule score in the alignment probability model, we can combine Arabic semantics and English semantics in determining the segmentation choice in context.

6.2 The alignment model

In order to compute an alignment probability for every pair (r, E_{a_i}), we need to estimate how r's segments translate to E_{a_i}'s tokens. To translate a source text sequence to a sequence in a target language, two questions must be asked:

1. What target words/phrases should we produce?
2. Where should we place them?

The IBM models as motivation.

We provide a quick introduction to the IBM models to give a general motivation to our proposed alignment model. Details that do not relate to our model are not discussed. For detailed discussion of the IBM models, refer to Brown *et al.* (1993). *IBM Model 1* answers *only* the first question by introducing a *lexical translation model*, $t(e_i|f_j)$, that estimates how well a source token e_i translates to a target token f_j. IBM Model 1 does not model word alignments explicitly which means once the target words are generated, they can be put in any order in the target sentence. To answer the second question, *IBM Model 2* adds an *absolute alignment model*, $a(i|j, m, n)$, that measures the probability of a target token at position j in a target sentence of length m to be aligned to a source word at position i in a source sentence of length n. This independent modeling of translation and alignment makes the problem easier to solve. *IBM Model 3* takes the first question a step further by modeling fertility which allows source words to produce multiple target words or even get dropped from translation and allows target words to be inserted without a source word generating them. Fertility gets handled by two separate models: (a) The *fertility model*, $y(n_{slots}|f)$, handles source word fertility by estimating the probability of a source word f to produce zero or more slots to be filled with target words. If $n_{slots} = 0$, source word f will be dropped from translation. If $n_{slots} > 0$, one or more target words will be generated. And (b) *NULL insertion* models the introduction of new target words without a source translation. While IBM Model 3 keeps the regular lexical translation model as is $t(e_i|f_j)$, it reverses the direction of Model 2's absolute alignment model to become $d(j|i, m, n)$, which they call an *absolute distortion model. IBM Model 4* further improves Model 3 by introducing a *relative distortion model* which allows target words to move around based on surrounding words instead of the length of source and target sentences. Finally, *IBM Model 5* fixes the deficiency in Model 3 and 4 that allows multiple target words to be placed in the same position.

Our alignment probability model.
The IBM models were originally proposed as MT systems, but now they are widely used as part of word alignments since more advanced MT approaches were introduced. While our alignment model is inspired by the IBM models, we have no intention to use it as an MT system; therefore, we are not bound to design an alignment model that generates fluent translations. In other words, the placement of English words in their *exact* positions (the second question) is not essential. Our model should measure how well a certain segmentation of an Arabic word a_i, produced by rule $r = a_i \rightarrow g_1g_2g_3$, aligns to English words in the target sequence E_{a_i}.

The English sequence could contain words that do not align to any segment of the source Arabic word. This is a result of erroneous alignments by Giza++ or due to our inclusion of surrounding words. To handle this, we model dropping words from E_{a_i} by introducing a *NULL token* on the Arabic side (with index 4) that misaligned English words can align to. This makes the Arabic sequence of length 4 indexed: 1 for prefix, 2 for stem, 3 for suffix, and 4 for the NULL token. We use the variable j to refer to this index. The English sequence can be of any length, denoted as m. As mentioned above, the original order of English words is preserved in the sequence E_{a_i} but re-indexed from 1 to m. We use the variable k to refer to this index.

Definition: Alignment vector. An *alignment vector* is a vector of m elements denoted as $L = (l_1, l_2, ..., l_m)$, where l_k is position in the Arabic segment that e_k aligns to. This allows multiple English words can align to the same token in the Arabic sequence, for example, "and" and "will" can align to + وح *wH*+ "and will." However, an English word, in our model, cannot align to multiple Arabic tokens, which forces the English word to pick its best aligned Arabic token. We define $L_{(r,E_{a_i})}$ as the set of all possible alignment vectors that align E_{a_i}'s words to r's segments and the NULL token.

Definitions: Center and direction. We define the *center* of the English sequence as the English word that best aligns to the Arabic stem. We denote its index as k_{stem}. Given k_{stem}, we define a *direction vector* $D = \{d_k : d_k = \mathbf{sgn}(k - k_{stem})\}$[d], where every English word e_k has a direction d_k relative to the center $e_{k_{stem}}$. This means that the center $e_{k_{stem}}$ has a direction of 0, words that appear *before* the center have a direction of −1, and words that appear *after* center have a direction of +1,

It is intuitive to assume that the direction of an English word relative to the center could have an impact on the decision of whether to align it to a prefix, a stem, a suffix, or even NULL to drop it from alignment as it might align to a previous or subsequent word in the Arabic sentence. To

[d]**sgn** is the sign function.

motivate this intuition, let us observe *closed-class* and *open-class* English words and their relations to the types of Arabic segments based on their directions from the center.

In our approach to segmentation, an Arabic affix is split from the stem as one unit without further splitting its internal components which could contain pronouns, prepositions, or particles such as conjunction, negation, and future particles. These affixes tend to align to closed-class English words. For example, the Arabic definite article, +ال *Al+* "the," appears only in prefixes (e.g., in +وال *wAl+* "and the"); similarly, the English word "the" appears only *before* the center when it aligns to a prefix. If "the" appears *after* the center, it probably should be aligned to a subsequent Arabic word in the source sentence. Moreover, the Arabic conjunction particle, + و *w+* "and," appears in prefixes (e.g., in +وح *wH+* "and will") or as a separate word و *w*; therefore, when "and" appears *before* or *at* the center it tends to align to a prefix or a stem, respectively. If "and" appears *after* the center, it probably should be dropped. Furthermore, the English word "to" could align to any token of the source sequence at any position in the target sequence; however, its direction relative to the center correlates with the position of the Arabic token it aligns to. Here are the four cases:

1. "to" could align to the prefix particle +ل *l+* "to" (as in this example attaching to a verb and a noun: "ليبعت لرفيقه " ***l**ybEt **l**rfyqh* "*to* send *to* his friend").[e] In such cases, the English word "to" has a direction of −1.
2. "to" could align to a stem such as إلى *<lY* "to," a separate preposition in Arabic. In these cases, "to" has a direction of 0.
3. "to" could align to a suffix containing the indirect object preposition ل- *-l* "to" (as in the suffix- ولك--*wlk* "they, to you" in "يبعتولك" *ybEtwlk* "they send to you"). In such cases, "to" has a direction of +1.
4. "to" could align to NULL if misaligned which could occur at any value for d_k.

Similar to closed-class words, open-class words tend to either align to the stem or to NULL. For example, there is no prefix or suffix that aligns to the word "send"; therefore, if it appears on either side of the center, it probably belongs to a surrounding word of the current Arabic word a_i. This motivates the design of a probability distribution that capitalize on this correlation.

Our model answers the two questions introduced earlier with two separate probability distributions:

1. *Lexical translation model:* $t(e_k|g_{l_k})$. This model is identical to IBM Model 1. It estimates the probability of translating an Arabic segment to an English word. For example, t("and"|"wH+") represents the probability of producing the word "and" from the prefix +وح *wH+* "and will."
2. *Lexical direction model:* $z(l_k|e_k, d_k)$. This model estimates the probability of aligning an English word e_k with direction d_k to position l_k in the Arabic sequence. For example, $z(1|\text{"and"}, -1)$ is the probability of the word "and" aligning to a prefix knowing that it appeared *before* the center.

In this model, the *exact* position of the generated English word is not important; instead, the direction relative to center is what matters.

To compute k_{stem}, we find the English word with the highest $t \times z$ score as in the following equation:

$$k_{stem} = \arg\max_{k} t(e_k|g_2) \times z(2|e_k, 0)$$

This might seem like a circular dependency: z depends on d_k which is computed from k_{stem} that depends on z. In other words, using the direction from the center while trying to find the center.

[e]In Arabic, *l+* can be a preposition that attaches to nouns, or a justification particle that attaches to verbs.

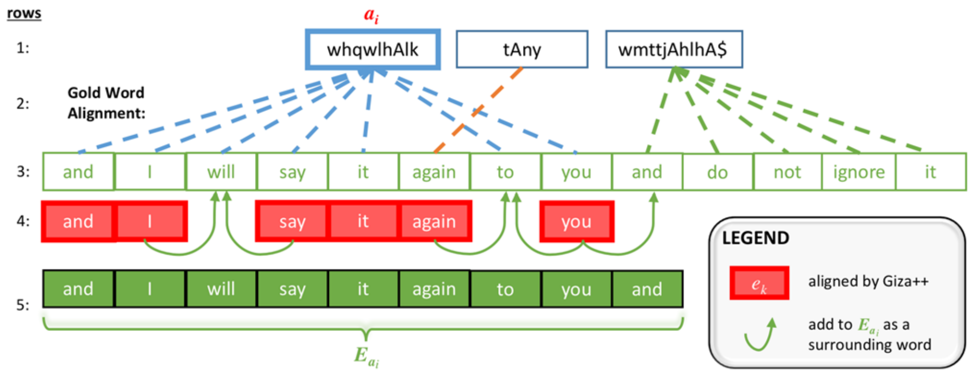

Figure 2. Example of sentence alignment that shows how we extract the English sequence E_{a_i} that aligns to a source word a_i. The figure is organized as rows indexed from 1 to 5 as shown on the left margin. Row 1 and 3 show the source Arabic sentence and its English translation. Row 2 shows the perfect word-level alignment between the two sentences. Row 4 shows the automatic process of extracting E_{a_i} by first adding words aligned by Giza++ (in red rectangles) and then adding surrounding words (identified by the green arrows). Row 5 shows the resulting E_{a_i}.

In fact, we do not need the direction from the center to compute k_{stem}. Instead, we set $d_k = 0$ in $z(2|e_k, 0)$, which, when multiplied with $t(e_k|g_2)$, basically asks the question: if word e_k were to be selected as the center, how well will it align to position 2 (the stem) in the source sequence? This breaks the circular dependency.

Example
Consider the Egyptian Arabic sentence وهقولهالك تاني ومتتجاهلهاش *whqwlhAlk tAny wmttjAhlhA$* translated to English as "And I will say it again to you and do not ignore it." Figure 2 presents the Arabic and English sentences in rows 1 and 3 (index is in the left margin), as well as their perfect word-level alignment (Row 2). For our example, we consider the first word, *whqwlhAlk*, as a_i and we construct E_{a_i} in Row 4 by first including words aligned by Giza++ (in red rectangles) and then adding surrounding words (identified by the green arrows). Row 5 shows the resulting E_{a_i}. Due to the infrequency of such highly inflected words, Giza++ tends to make errors aligning them. In this case, it erroneously aligns "again" to a_i and misses "will" and "to" which should have been aligned. Our inclusion of surrounding words results in adding the missed words but also includes the trailing "and" erroneously. This approach increases recall while compromising precision since it depends on the probabilistic model to maximize English alignment to a_i's internal structure while dropping the misaligned English words.

Figure 3 shows the alignment of the Arabic word وهقولهالك *whqwlhAlk* from Figure 2 with its aligned English sequence E_{a_i} = (and, I, will, say, it, again, to, you, and). This example shows how our model would score an *alignment vector* $L = (1, 2, 1, 2, 3, 4, 3, 3, 4)$ linking E_{a_i} tokens one-to-many to the four Arabic tokens. L, shown in Part (*b*) of the figure (index is in the left margin), is actually the gold alignment vector. Part (*a*) shows the *lexical translation model*, $t(e_k|g_{l_k})$, generating English words from Arabic tokens under alignment vector L. The English word "say" is picked as *the center* over "I" because $t(\text{say}|\text{qwl}) > t(I|\text{qwl})$. Part (*b*) shows how the *lexical direction model*, $z(l_k|e_k, d_k)$, predicts the position an English word e_k with direction d_k aligns to.

Decoding with the model: finding the best segmentation choice
The probability of an alignment vector L that aligns the words of an English sequence, E_{a_i}, to the four Arabic tokens produced by a rule r and the NULL token is denoted as $p_{align}(E_{a_i}, L|r)$ and is given by this equation:

$$p_{align}(E_{a_i}, L|r) = \prod_{k=1}^{m} t(e_k|g_{l_k}) \times z(l_k|e_k, d_k) \tag{12}$$

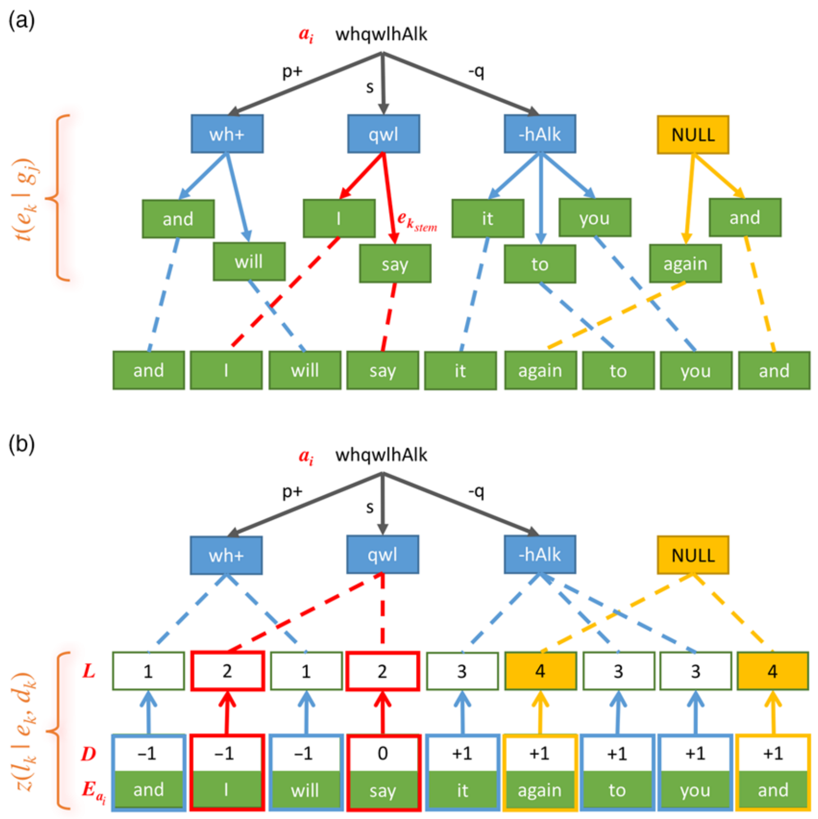

Figure 3. Example of alignment model parameters *t* and *z* for an Arabic word aligned to an English phrase.

To find the best segmentation choice for an Arabic word a_i with a set of rules R_{a_i}, we pick the rule, r^*, that has the highest score when aligned to the English sequence E_{a_i}. To evaluate how well a rule r aligns to E_{a_i}, we scan all possible alignment vectors generated from r and E_{a_i} to find the one with the highest probability $p_{align}(E_{a_i}, L|r)$. Therefore, the best segmentation choice for a_i is generated by the rule r^* that has the best alignment to E_{a_i}'s words among all other rules in R_{a_i}, as shown in the following equation:

$$r^* = \arg\max_{r \in R_{a_i}} \max_{L \in L_{(r, E_{a_i})}} p_{align}(E_{a_i}, L|r) \tag{13}$$

It is important to note that the rule score computed in Section 5, *score*(r), is not used directly in these equations; however, it is used in estimating the model's parameters t and z.

6.3 Parameter estimation with EM

In this subsection, we present our EM algorithm to estimate our model's parameters. First, we start with initializing our parameters and then we explain the EM algorithm used for parameter estimation.

Initialization. Our initialization step starts by running Giza++ (Och and Ney 2003b) on the untokenized Arabic–English parallel data to produce many-to-many word-level alignments. Using these alignments, we estimate two word translation probability distributions for each Arabic word a and English word e:

1. The *word translation probability*: $p_1(e|a) = count_{aligned}(a, e)/count(a)$.
2. The *reverse word translation probability*: $p_2(a|e) = count_{aligned}(a, e)/count(e)$.

where $count_{aligned}(a, e)$ is the number of times we saw Arabic word a aligned to English word e in Giza++ output, while $count(x)$ is the frequency of word x in the parallel corpora. The word translation probability distribution is used to initialize the $t(e_k|g_h)$ parameter of our model as shown in Algorithm 2 in Equation (14). Since the stems are actual Arabic words in our definition, we will have $p_1(e_k|g_2)$ probabilities for stems; however, this is not possible for affixes. Therefore, we compute the count $c(e_k, g_h)$ by summing over all $\langle a_i, e_k\rangle$ pairs where $p_1(e_k|a_i) > 0$ and a_i has one or more rules, $r = a_i \rightarrow G$, that generate the segment g_h, and computing the score $p_1(e_k|a_i) \times score(a_i \rightarrow G)$ that is then added to $p_1(e_k|g_h)$ if nonzero. The $z(l_k|e_k, d_k)$ parameters are uniformly distributed as shown in Algorithm 2, Equation (15).

Parameter estimation. Algorithm 2 presents an EM algorithm where every epoch iterates over every aligned Arabic–English pair $\langle a_i, E_{a_i}\rangle$ in the parallel text and computes counts from every possible segmentation of a_i that aligns to E_{a_i}'s tokens. In Line 5, we compute the confidence in this aligned pair, $conf$, that will be used in computing δ. In Line 8, we compute k_{stem} which gives us the English token, $e_{k_{stem}}$, with the highest alignment to the stem in the current segmentation of a_i. If the alignment score of this word, $t(e_{k_{stem}}|g_2) \times z(2|e_{k_{stem}})$ is lower than a threshold, we ignore the rule that produced this segmentation. The threshold can be manipulated to trade off quality and number of labeled segmentation choices. Once k_{stem} is found, the direction vector, D, can be computed (Line 11). Then, every possible combination of E_{a_i} tokens and a_i segments are considered to compute δ_{tz}, using the last iteration t and z parameters (Line 16), which is combined with the rule score and the aligned pair confidence score to compute δ. δ is then used to compute the counts. Finally, the model's probabilities are calculated (Lines 23–24) to be used in the next epoch.

Experiments. We use three parallel corpora obtained from several LDC corpora including GALE and BOLT data and preprocessed by separating punctuation marks and Alif/Yah normalization. The corpora are Egyptian–English (Egy–En) corpus of ~2.4M tokens, Levantine–English (Lev–En) corpus of ~1.5M tokens, and MSA–English (MSA–En) corpus of ~49.5M tokens. The combined corpus, which amounts to ~53.5M tokens, is word-aligned using GIZA++ (Och and Ney 2003a) and used as training data to this step.

We trained the EM algorithm for 50 rounds on this data and we labeled 11.9M words with acceptable confidence.[f]

7. Supervised segmentation using synthetic data

We train a supervised segmenter to learn how to segment Arabic words in a given sentence. For every word in the sentence, the segmenter considers a list of rules to transform that word and scores them using an ASP. The transformation rules are segmentation rules that produce a stem and affixes. This setup allows for more advanced transformation rules where the stem is not a partial string of the source word. Examples are spelling variation normalization, templatic morphology segmentation, and even infrequent-to-frequent word translation; although, new features should be introduced to capture those advanced transformations. The empty affixes "P+" and "-Q" are not generated in the segmented output. We train and evaluate our segmenter on train/test split sets of our automatically labeled data.

[f]The confidence threshold was determined manually after sorting the labeled examples by confidence.

Algorithm 2 Affix–stem joint probability estimation.

```
1:  // Parameter initialization:
        $t(e_k|g_h) = c(e_k, g_h) / \sum_{e'} c(e', g_h)$    (14)
    where $c(e_k, g_h) = p_1(e_k|g_h) + \sum_{\langle a_i, e_k \rangle} \sum_{\substack{a_i \to G \\ g_h \text{ in } G}} p_1(e_k|a_i) \times score(a_i \to G))$
        $z(l_k|e_k, d_k) = 1/4$ // uniform    (15)
2:  // Parameter estimation:
3:  for $round := 1 \to MAX$ do
4:      for each aligned pair $\langle a_i, E_{a_i} \rangle$ do
5:          $conf \leftarrow \sum_k p_1(e_k|a_i) \times p_2(a_i|e_k)$ // Alignment Confidence:
6:          // Collect counts:
7:          for each rule $r = a \to g_1g_2g_3$ do
8:              $k_{stem} \leftarrow \arg\max_k t(e_k|g_2) \times z(2|e_k, 0)$
9:              if $t(e_{k_{stem}}|g_2) \times z(2|e_{k_{stem}}, 0) >$ threshold then
10:                 // Get direction vector:
11:                 $D \leftarrow \{d_k = \text{sgn}(k - k_{stem})\}$
12:                 // For each English word:
13:                 for $k := 0 \to m$ do
14:                     // For each alignment position:
15:                     for $j := 0 \to 3$ do
16:                         $\delta_{tz} \leftarrow t(e_k|g_j)z(j|e_k, d_k) / \sum_h t(e_k|g_h)z(h|e_k, d_k)$
17:                         $\delta \leftarrow conf \times score(r) \times \delta_{tz}$
18:                         $count_t(e_k, g_j) \mathrel{+}= \delta$
19:                         $total_t(g_j) \mathrel{+}= \delta$
20:                         $count_z(j, e_k, d_k) \mathrel{+}= \delta$
21:                         $total_z(e_k, d_k) \mathrel{+}= \delta$
22:     // Estimate probabilities:
23:     $t(e_k|g_h) \leftarrow count_t(e_k, g_j) / total_t(g_j)$
24:     $z(j|e_k, d_k) \leftarrow count_z(j, e_k, d_k) / total_z(e_k, d_k)$
```

7.1 Challenges of automatically labeled data

Two challenges for training and decoding arise from the nature of our automatically labeled data: frequent OOVs and unlabeled words in a training sentence.

Frequent words as OOVs. The general case in manually annotated training data for NLP tasks is that the data are selected randomly from the text expected to be handled by the NLP tool (sometimes with a bias toward more frequent cases in the target area, such as domain, dialect, and genre). The frequent words in the vocabulary, as a result, will generally be labeled in the training data. This means that OOVs in a given test set are usually infrequent words and, thus, rare in those sets. In our setup, we label words with their segmentation choice only when we have high confidence in our decision. This leaves our partially labeled training data with many unlabeled frequent words. These words are naturally frequent in a given test set, which means they will be OOVs for a system trained on our partially labeled training data. This problem requires us, as we design the segmenter, to give special attention to its ability to learn how to generalize to unseen words, since those unseen words are now a frequent phenomena. To do so, we use the following strategies:

1. We introduce features that deliberately try *not* to memorize specific segmentations in order to allow them to generalize to OOVs.
2. We drop all segmentation rules learned in Section 5 since they do not extend to a large portion of the vocabulary. To generate a list of rules for a given word in a sentence, the decoder, now, considers all possible segmentations of that word that produce stems that have been seen in the parallel data. This will introduce new, unseen affixes, which is intended to generalize to unseen dialectal morphology.
3. Some of our best features use distance scores from the Arabic and English clusters of the word being processed (discussed later). Since these clusters were used in creating our labeled data, all labeled words have both clusters. This results in a generalizability issue as many other words may have one or no clusters. To solve this issue, we deliberately drop either or both clusters for some labeled words in a random manner in order to allow other general features to be trained for such cases.
4. To evaluate our segmenter's ability to generalize to OOVs, we randomly drop some test set words from the training data to generate OOVs that we can evaluate on. This allows us to pick a system with high generalizability.

Unlabeled words in a training sentence. Unlike manually labeled data where all words in a training sentence are labeled, our data may have many unlabeled words in a given training sentence. This means that features of a rule cannot depend on the segmentation decision made for the previous word since we cannot guarantee knowing that decision during training. Therefore, the decoder cannot use the Viterbi algorithm; instead, it picks the segmentation rule with the highest score for every word independently. We do, however, use features that look at the possible segmentation rules of surrounding words which are inspired by the gender/number agreement of Arabic.

7.2 Features

Global linear models allow us to use millions of features to score the different rules of a given word in context.

A feature is 1 if a certain condition is satisfied by the rule and 0 otherwise. For example, the feature below fires up when a rule is trying to segment the prefix وه *wh+* "and will" from a stem with four letters (which could be a present verb in Arabic):

$$\phi_{1000}(a \rightarrow psq) = \begin{cases} 1, & \text{if length of stem } s = 4 \\ & \text{and prefix } p = \text{"wh+"} \\ 0, & \textit{otherwise} \end{cases}$$

Salloum (2018) includes a detailed description of the features we used, such as features based on the main word and its segments, surrounding words, tokens' length and frequency, affixes, English and Arabic clusters, and affix correlation. We use the structured perceptron algorithm with averaging (for regularization) to learn the weights of those features.

7.3 Experiments and evaluation

Experimental setup. We implement our own ASP trainer and decoder.We split the automatically labeled examples from the previous step into train (∼9.9M labeled tokens) and dev and test sets (1M labeled tokens each).

Table 1. The segmentation decoder results in terms of accuracy (number of correct segmentations/total number of tokens) on both the dev and blind test sets. The first section shows the results on all tokens, while the following sections break the tokens down into categories

	dev				test			
	# correct	/	# tokens	= accuracy	# correct	/	# tokens	= accuracy
All tokens	710,527	/	721,771	98.44%	684,123	/	693,994	98.58%
Breakdown by INVs and OOVs:								
INVs	575,529	/	575,531	∼100.00%	556,767	/	556,767	100.00%
OOVs	134,998	/	146,240	92.31%	127,356	/	137,227	92.81%
OOV categories:								
No Arabic cluster	18,111	/	19,247	94.10%	14,304	/	16,441	87.00%
No English cluster	66,756	/	72,694	91.83%	16,435	/	17,947	91.58%
Neither cluster	7423	/	8283	89.61%	1650	/	1959	84.23%
Both clusters	58,690	/	62,582	93.78%	98,689	/	104,798	94.17%
No segmentation	5641	/	7284	77.44%	7379	/	9441	78.16%

Evaluation. We ran hundreds of experiments in a linguistically motivated greedy approach to engineer good feature types for our segmenter. The systems learned from the top-performing feature type combinations were then evaluated by the MT experiments to pick the best segmenter.

We empirically determined the number of epochs to be 14. In development and test, we automatically generate OOVs by randomly omitting Arabic and English clusters as discussed earlier. This makes the non-cluster features fully responsible for segmenting those words without the reliance on cluster features, which allows the segmenter to tune their weights and thus generalize to actual MT sets' OOVs.

Table 1 presents the performance of the best segmenter system in terms of accuracy on both dev and test sets (the last two columns across all sections). The sections of the table represent the breakdown of tokens into categories for in-depth evaluation. The accuracy scores for each of these categories were used to engineer our feature types to ensure that they generalize to frequent OOVs belonging to those categories. We also make sure, while automatically generating OOVs in dev/test sets, that we have enough tokens in each category to guarantee a representative evaluation.

Since the segmenter's job is to pick a segmentation rule out of a generated list of rules, we evaluate only on words with more than one rule (multi-choice) which constitute 721,771 tokens of the 1M-token dev set and 693,994 tokens of the 1M-token blind test set. The rest of the tokens have only one segmentation rule: no segmentation. The first section of Table shows the accuracy of the segmenter on multi-choice tokens. In the second section, we break down the evaluation to in-vocabulary words (INVs) and OOVs.

Since we might not have Arabic or English clusters for many words in test sets, we define four categories representing the absence of either cluster, both, or neither. These categories are mutually exclusive. We also evaluate on OOVs that should not be segmented, yet have multiple rules, to reduce our decoder's over-segmentation. The third section of Table presents evaluation on multi-choice OOV categories. The results on the dev set carry on to the blind test set.

8. Evaluation on MT

8.1 MT experimental setup

MT train/tune/test data. We combine the training, dev, and test sets we used to train and evaluate our segmenter into one parallel corpus and we use it to train our MT systems. The MT tune, dev, and test sets, however, are selected from several standard MT test sets. We use three Egyptian sets from LDC BOLT data with two references (EgyDevV2, EgyDevV3, and EgyTestV2), and one Levantine set from BBN (Zbib *et al.* 2012) with one reference which we split into LevDev and LevTest. We use EgyDevV3 to tune our SMT systems. We use the remaining sets for development and test on both Egyptian and Levantine. For dev, we use EgyDevV2 and LevDev and for test we use EgyTestV2 and LevTest. It is important to note that the segmenter has never seen these tune/dev/test sets. The segmenter was only trained on the MT training data.

MT tools and settings. We use the open-source Moses toolkit (Koehn *et al.* 2007) to build our Arabic–English phrase-based SMT systems.[g] Our systems use a standard phrase-based architecture. The language model for our systems is trained on English Gigaword (Graff and Cieri 2003). We use SRILM Toolkit (Stolcke 2002) to build a 5-gram language model with modified Kneser–Ney smoothing. Feature weights are tuned to maximize BLEU on the tuning set using minimum error rate training (Och 2003). Results are presented in terms of BLEU (Papineni *et al.* 2002) and METEOR (Banerjee and Lavie 2005). All evaluation results are case- *insensitive*. The English data are tokenized using simple punctuation-based rules. The Arabic text is also Alif/Ya normalized. For more details on processing Arabic, see (Habash 2010).

8.2 MT experiments

We use the same parallel data to train all of our MT systems and the same dev and test sets to evaluate. The only difference is the preprocessing of the Arabic side of training, dev, and test data. Table shows MT experiment results in terms of BLEU and METEOR on dev (first set of columns) and blind test (second set of columns).

Baseline systems. We build three baseline MT systems to compare our systems against. In the first baseline system, we Alif/Ya normalize the Arabic side but we leave it unsegmented. We call this baseline MT$_{\text{UNSEGMENTED}}$. The other two baseline systems are based on two previous research efforts representing two approaches to morphological segmentation. The first is a tool for language-independent, unsupervised learning of morphology: MORFESSOR (Creutz and Lagus 2002) to segment the Arabic side, and the second is a dialect-specific tool that requires handcrafted resources and is trained on hand-labeled data: MADAMIRA-EGY, the version of MADAMIRA (Pasha *et al.* 2014) that handles Egyptian as well as MSA. We use these two tools to preprocess Arabic and we name the resulting two MT systems after them: MT$_{\text{MORFESSOR}}$ and MT$_{\text{MADAMIRA-EGY}}$, respectively. All Arabic textual data (parallel and monolingual) were used to train MORFESSOR.

The first section of Table 2 presents results on these baselines. On the dev set, MT$_{\text{MORFESSOR}}$ performs significantly better than MT$_{\text{UNSEGMENTED}}$ on Egyptian (1.6% BLEU, 1.4% METEOR) and slightly better on Levantine (0.2% BLEU, 0.3% METEOR). This could be due to the limited Levantine text in MORFESSOR's segmentation training data compared to Egyptian and MSA.

[g]This work is concerned with unsupervised segmentation of Arabic dialects, a preprocessing step that could help improve the performance of any MT system, whether neural (NMT) or phrase-based SMT. We chose to evaluate on SMT over NMT because, as of the time of this work, SMT systems still outperform NMT systems when it comes to Arabic dialects. This is due to many reasons, one of which is the limited size of parallel data. In the recent work of (Oudah *et al.* 2019), the authors showed that with 1.2M sentences in Standard Arabic, SMT outperformed NMT and there was a consistent added value of segmentation.

Table 2. Evaluation in terms of BLEU and METEOR (MET) of our two MT systems, MT$_{\text{Context-Sensitive}}$ and MT$_{\text{Context-Insensitive}}$, on a dev test (first set of columns) and a blind test set (second set of columns), in comparison with three baselines, MT$_{\text{Unsegmented}}$, MT$_{\text{Morfessor}}$, and MT$_{\text{Madamira-Egy}}$. Results in bold show the highest performing system in the column.

	Dev				Test			
	Egy		Lev		Egy		Lev	
	BLEU	MET	BLEU	MET	BLEU	MET	BLEU	MET
MT$_{\text{Unsegmented}}$	19.2	27.0	13.5	21.3	21.8	28.1	13.3	21.7
MT$_{\text{Morfessor}}$	20.8	28.4	13.7	21.6	21.7	29.2	13.6	22.4
MT$_{\text{Madamira-Egy}}$	**21.5**	29.0	15.2	22.4	**23.0**	**30.0**	15.2	23.1
MT$_{\text{Context-Sensitive}}$	21.0	28.3	15.6	22.9	22.4	29.3	15.6	23.6
MT$_{\text{Context-Insensitive}}$	21.4	**29.2**	**16.2**	**23.5**	**23.0**	29.9	**16.3**	**24.0**
Over MT$_{\text{Unsegmented}}$	+2.2	+2.2	+2.7	+2.2	+1.2	+1.8	+3.0	+2.3
Over MT$_{\text{Morfessor}}$	+0.6	+0.8	+2.5	+1.9	+1.3	+0.7	+2.7	+1.6
Over MT$_{\text{Madamira-Egy}}$	*−0.1*	+0.2	+1.0	+1.1	0.0	*−0.1*	+1.1	+0.9

MT$_{\text{Madamira-Egy}}$ outperforms the other baselines on both dialects on both metrics. An interesting case is MT$_{\text{Madamira-Egy}}$ results on Levantine dev: it improves over MT$_{\text{Unsegmented}}$ by 2.3% BLEU, 2.0% METEOR, and over MT$_{\text{Morfessor}}$ by 1.5% BLEU, 0.8% METEOR. MT$_{\text{Madamira-Egy}}$'s good performance on Levantine can be explained by the fact that these two dialects share many of their dialectal affixes and clitics (e.g., +ح *H*+ "will," ب+ *b*+ "simple present," ـلك *-lk* "to you") as well as many lemmas. Moreover, most of the phonological differences between Levantine and Egyptian do not show up in the orthographic form since Arabic writers tend to drop short vowels and spell some sounds etymologically, thus normalizing them, for example, the words for "leather" (Egyptian /geld/, Levantine /ʒeled/, and MSA /ʤild/) are all written جلد *jld*. This leads to many Levantine words looking identical to their Egyptian equivalents although they are pronounced differently.

Our MT systems. We present two MT systems to evaluate two of our segmentation models. The first model is trained using the best-performing combination of context-sensitive and insensitive features, while the second model uses the best-performing combination of context-insensitive features only presented in Table 1. We call the resulting MT systems: MT$_{\text{Context-Sensitive}}$ and MT$_{\text{Context-Insensitive}}$, respectively. We present MT results on our systems in the second section of Table. MT$_{\text{Context-Insensitive}}$ outperforms MT$_{\text{Context-Sensitive}}$ across dialects and metrics. Investigating the output of both systems shows that the inconsistencies generated by context-based segmentation outweighs the benefits of disambiguation, especially that phrase-based SMT is robust toward infrequent systematic segmentation errors across training, tuning, and test sets.

The third section of Table 2 reports the differences between our best system's results and those of the three baselines. MT$_{\text{Context-Insensitive}}$ improves over both resource-free baselines (MT$_{\text{Unsegmented}}$ and MT$_{\text{Morfessor}}$) across dev sets and metrics ranging from 2.2% BLEU on Egyptian and 2.7% BLEU on Levantine over MT$_{\text{Unsegmented}}$ to 0.6% BLEU on Egyptian and 2.5% BLEU on Levantine over MT$_{\text{Morfessor}}$. These results demonstrate the usefulness of such approach where resources are unavailable.

Table 3. An Arabic sentence .بنشوفهن كتير بسوق الحمرا والصالحية "We see them a lot in the Hamra and Salehieh markets." translated by the three baselines and our best system

System	Processed Arabic	English Translation
MT_{UNSEGMENTED}	bn$wfhn ktyr bswq AlHmrA wAlSAlHyp	wAlSAlHyp red market ; we see them influenced a lot
MT_{MORFESSOR}	b+ n$wfhn ktyr b+ swq AlHmrA w+ Al+ SAlHyp	to see them a lot in the market the red salihiyah,
MT_{MADAMIRA-EGY}	bn$wfhn ktyr b+ swq AlHmrA wAlSAlHyp	we see a lot of hamra wAlSAlHyp market
MTCONTEXT -INSENSITIVE	bn+ $wf -hn ktyr b+ swq Al+ HmrA w+ AlSAlHyp	we see them a lot in souk al hamra and al-saleheyya

When compared to $\text{MT}_{\text{MADAMIRA-EGY}}$, however, performance on Egyptian differs from Levantine. The results on Egyptian are inconclusive: in terms of BLEU, $\text{MT}_{\text{MADAMIRA-EGY}}$ leads by 0.1%, while in terms of METEOR, $\text{MT}_{\text{CONTEXT-INSENSITIVE}}$ leads by 0.2%. These results mean that our best MT system is on par with $\text{MT}_{\text{MADAMIRA-EGY}}$ on Egyptian, which we consider a good result since MADAMIRA-EGY has been optimized for years with a wealth of dialect and task-specific resources. On Levantine, however, our system outperforms $\text{MT}_{\text{MADAMIRA-EGY}}$ by 1.0% BLEU, 1.1% METEOR. The results on blind test sets, presented in the second set of columns of Table 2 (Test), agree with the results on the dev sets and confirm their conclusions.

8.3 *Example and discussion*

Table 3 presents an example Levantine sentence translated by the three baselines and our best system. While each baseline has its own errors, our system produces a perfect translation although the reference does not match it word-for-word due to the several acceptable transliterations of the mentioned proper names found in our MT training data.[h] This results in penalties by BLEU and, in this example, METEOR; nevertheless, the translation is sound. The example contains words with different characteristics that are handled differently and sometimes similarly by the four systems:

1. The word بنشوفهن *bn$wfhn* "we see them" has rich Levantine morphology. Unlike $\text{MT}_{\text{MORFESSOR}}$ and $\text{MT}_{\text{MADAMIRA-EGY}}$, our system segments this word to three tokens that map directly to the three English words of the correct translation.
2. The word بسوق *bswq* "in market" has MSA morphology and is segmented correctly by all systems (except $\text{MT}_{\text{UNSEGMENTED}}$) which results in correct translations ($\text{MT}_{\text{CONTEXT-INSENSITIVE}}$ translates *swq* to "Souk," a correct transliteration found in our MT training data since it is frequently part of a proper noun).
3. The word الحمرا *AlHmrA* "Al Hamra" ("Al" is the definite article in Arabic) is a proper noun although the word literally means "red" which led to the mistakes by $\text{MT}_{\text{UNSEGMENTED}}$ and $\text{MT}_{\text{MORFESSOR}}$. Both $\text{MT}_{\text{CONTEXT-INSENSITIVE}}$ and $\text{MT}_{\text{MADAMIRA-EGY}}$ produce an acceptable transliteration.

[h]None of the systems discussed in this work has a transliteration component. All transliterations produced by these systems (e.g., the different transliterations of *AlSAlHyp*) are found in our MT training data.

4. The word والصالحية *wAlSAlHyp* “and Al Salehieh” is the name of the second market with the conjunction particle و *w* “and” attached to it. Both MT_{CONTEXT-INSENSITIVE} and MT_{MORFESSOR} succeed in segmenting this word to produce an acceptable translation and transliteration, although MT_{MORFESSOR} fails to produce “and.” This word shows an advantage that our segmenter and MORFESSOR have over MADAMIRA-EGY. Since they learn their morphemes and stems from data, they can better handle morphologically inflected proper nouns and dialectal/infrequent lemmas that do not appear in MADAMIRA-EGY’s internal morphological analyzer database.

9. Conclusion and future directions

In this work, we presented an approach to cheaply scale morphological segmentation to many dialects without the need for DA preprocessing tools. This approach attempts at learning an underlying Arabic preprocessing models for all Arabic varieties, including MSA. The approach expects a small amount of DA–English parallel data along with a sizable amount of MSA–English data.

Our approach learns out-of-context preprocessing rules for DA from unlabeled monolingual data. We use an unsupervised approach applied on large quantities of unlabeled Arabic text to extract a list of out-of-context preprocessing rules with weights estimated with EM. We use these rules in another unsupervised learning approach to automatically label words in the dialectal side of a DA–English parallel corpus. In a given DA sentence, a word is labeled in context with its best preprocessing rule which generates tokens that maximize alignment and translation to English words in the English translation of the corresponding sentence. This synthetic labeled corpus is used to train a supervised segmenter with features designed to capture general orthographic, morphological, and morphosyntactic behavior in Arabic words in order to generalize to unseen words.

We evaluated our approach on morphological segmentation and showed significant improvements on Egyptian and Levantine compared to other unsupervised segmentation systems. We also showed that our system is on par with the state-of-the-art morphological tokenizer for Egyptian Arabic built with supervised learning approaches that require manually labeled data, a large budget, and years to build. This shows that our approach can cheaply and quickly scale to more dialects while still performing on par with the best supervised learning algorithm. Furthermore, our evaluation on Levantine Arabic showed an improvement of 3% over an unsegmented baseline, 2.7% over the unsupervised segmentation system, and 1.1% over the supervised tokenization system, in terms of BLEU. This is especially important given that our system was not trained on monolingual Levantine text, which means that Levantine preprocessing rules were not learned; yet, our segmenter was able to generalize to Levantine.

In the future, we plan to evaluate our work on more dialects and subdialects where DA–English may or may not be available. We also plan to apply our approach to tasks other than morphological segmentation.

Acknowledgements. This research was supported in part by the Defense Advanced Research Projects Agency (DARPA) GALE program, contract HR0011-06-C-0022, and the DARPA BOLT program, contract No. HR0011-12-C-0014. Any opinions, findings, conclusions, or recommendations expressed in this work are those of the authors and do not necessarily reflect the view of DARPA.

References

Abdelali A., Darwish K., Durrani N. and Mubarak H. (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*. San Diego, California: Association for Computational Linguistics, pp. 11–16.

Abo Bakr H., Shaalan K. and Ziedan I. (2008). A hybrid approach for converting written Egyptian colloquial dialect into Diacritized Arabic. In *The 6th International Conference on Informatics and Systems, INFOS2008*. Cairo University.

Al-Badrashiny M., Pasha A., Diab M.T., Habash N., Rambow O., Salloum W. and Eskander R. (2016). SPLIT: Smart Preprocessing (Quasi) Language Independent Tool. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC-2016)*.

Al-Sabbagh R. and Girju R. (2010). Mining the web for the induction of a Dialectical Arabic Lexicon. In Calzolari N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M. and Tapias, D. (eds), *LREC*. European Language Resources Association.

Banerjee S. and Lavie A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72.

Brown P.F., Pietra S.A. Della P., Della V.J. and Mercer, R.L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, **19**, 263–312.

Buckwalter T. (2004). *Buckwalter Arabic Morphological Analyzer Version 2.0*. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Callison-Burch C., Koehn P. and Osborne M. (2006). Improved statistical machine translation using paraphrases. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 17–24.

Chiang D., Diab M., Habash N., Rambow O. and Shareef S. (2006). Parsing arabic dialects. In *Proceedings of the European Chapter of ACL (EACL)*.

Creutz M. and Lagus K. (2002). Unsupervised discovery of morphemes. In: *ACL 2002 Workshop on Morphological and Phonological Learning*. ACL.

Creutz M. and Lagus K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, **4**(1).

Diab M., Hacioglu K. and Jurafsky D. (2007). Automated methods for processing Arabic text: From tokenization to base phrase chunking. In van den Bosch A. and Soudi A.morphological analyzer for Egyptian Arabic (eds), *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Kluwer/Springer.

Du J., Jiang J. and Way A. (2010). Facilitating translation using source language paraphrase lattices. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2010, pp. 420–429.

Duh K. and Kirchhoff K. (2005). POS tagging of dialectal Arabic: A minimally supervised approach. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, Semitic 2005*, pp. 55–62.

El Kholy A. and Habash N. (2010). Techniques for Arabic morphological detokenization and orthographic denormalization. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*.

Erdmann A., Khalifa S., Oudah M., Habash N. and Bouamor H. (2019). A little linguistics goes a long way: Unsupervised segmentation with limited language specific guidance. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Florence, Italy: Association for Computational Linguistics, pp. 113–124.

Eskander R., Habash N. and Rambow O. (2013). Automatic extraction of morphological lexicons from morphologically annotated corpora. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 1032–1043.

Eskander R., Habash N., Rambow O. and Pasha A. (2016). Creating resources for dialectal Arabic from a single annotation: A case study on Egyptian and Levantine. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 3455–3465.

Graff D. and Cieri C. (2003). *English Gigaword, LDC Catalog No.: LDC2003T05*. Linguistic Data Consortium, University of Pennsylvania.

Graff D., Maamouri M., Bouziri B., Krouna S., Kulick S. and Buckwalter T. (2009). *Standard Arabic Morphological Analyzer (SAMA) Version 3.1*. Linguistic Data Consortium LDC2009E73.

Habash N. (2006). On Arabic and its dialects. *Multilingual Magazine*, **17**(81).

Habash N. (2010). *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Habash N. and Rambow O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pp. 573–580.

Habash N. and Rambow O. (2006). MAGEAD: A morphological analyzer and generator for the Arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 681–688.

Habash N. and Sadat F. (2006). Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 49–52.

Habash N., Soudi A. and Buckwalter T. (2007). On Arabic transliteration. In van den Bosch A. and Soudi A. (eds.), *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Habash N., Eskander R. and Hawwari A. (2012a). A morphological analyzer for Egyptian Arabic. In *NAACL-HLT 2012 Workshop on Computational Morphology and Phonology (SIGMORPHON2012)*, pp. 1–9.

Habash N., Eskander R. and Hawwari A. (2012b). A morphological analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pp. 1–9.

Habash N., Diab M. and Rabmow O. (2012c). Conventional orthography for dialectal Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Habash N., Roth R., Rambow O., Eskander R. and Tomeh N. (2013). Morphological analysis and disambiguation for dialectal Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Habash N., Eryani F., Khalifa S., Rambow O., Abdulrahim D., Erdmann A., Faraj R., Zaghouani W., Bouamor H., Zalmout N., Hassan S., Shargi F.A., Alkhereyf S., Abdulkareem B., Eskander R., Salameh M. and Saddiki H. (2018). Unified guidelines and resources for Arabic Dialect orthography. In: *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Hajič J., Hric J. and Kubon V. (2000). Machine translation of very close languages. *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000)*, pp. 7–12.

Hamdi A., Boujelbane R., Habash N., Nasr A., *et al.* (2013). The effects of factorizing root and pattern mapping in bidirectional Tunisian-Standard Arabic machine translation. *MT Summit 2013*.

Khalifa S., Zalmout N. and Habash N. (2016). YAMAMA: Yet another multi-dialect Arabic morphological analyzer. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 223–227.

Khalifa S., Hassan S. and Habash N. (2017). A morphological analyzer for Gulf Arabic verbs. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*.

Kilany H., Gadalla H., Arram H., Yacoub A., El-Habashi A. and McLemore C. (2002). *Egyptian Colloquial Arabic Lexicon*. LDC catalog number LDC99L22.

Koehn P., Hoang H., Birch A., Callison-Burch C., Federico M., Bertoldi N., Cowan B., Shen W., Moran C., Zens R., Dyer C., Bojar O., Constantin A. and Herbst E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pp. 177–180.

Kumar S., Och F.J. and Macherey W. (2007). Improving word alignment with bridge languages. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 42–50.

Mikolov T., Chen K. Corrado G. and Dean J. (2013). Efficient estimation of word representations in vector space. *CoRR*.

Mohamed E., Mohit B. and Oflazer K. (2012). Annotating and learning morphological segmentation of Egyptian colloquial Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Nakov P. and Ng H.T. (2011). Translating from morphologically complex languages: A paraphrase-based approach. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL 2011)*.

Narasimhan K., Barzilay R. and Jaakkola T. (2015). An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics (TACL)*, **3**, 157–167.

Och F.J. (2003). Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*, pp. 160–167.

Och F.J. and Ney H. (2003a). A systematic comparison of various statistical alignment models. *Computational Linguistics*, **29**(1), 19–51.

Och F.J. and Ney H. (2003b). A systematic comparison of various statistical alignment models. *Computational Linguistics*, **29**(1), 19–52.

Oudah M., Almahairi A. and Habash N. (2019). The impact of preprocessing on Arabic-English statistical and neural machine translation. *CoRR*, abs/1906.11751.

Papineni K., Roukos S., Ward T. and Zhu W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318.

Parker R., Graff D., Chen K., Kong J. and Maeda K. (2009). *Arabic Gigaword Fourth Edition*. LDC catalog number No. LDC2009T30, ISBN 1-58563-532-4.

Pasha A., Al-Badrashiny M., Diab M.T., El Kholy A., Eskander R., Habash N., Pooleery M., Rambow O. and Roth R. (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-2014)*.

Riesa J. and Yarowsky D. (2006). Minimally supervised morphological segmentation with applications to machine translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA 2006)*, pp. 185–192.

Sadat F. and Habash N. (2006). Combination of Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia: Association for Computational Linguistics, pp. 1–8.

Sajjad H., Darwish K. and Belinkov Y. (2013). Translating dialectal Arabic to English. In *The 51st Annual Meeting of the Association for Computational Linguistics - Short Papers (ACL Short Papers 2013), Sofia, Bulgaria*.

Salloum W. (2018). *Machine Translation of Arabic Dialects*. Ph.D. thesis, Columbia University in the City of New York.

Salloum W. and Habash N. (2011). Dialectal to standard Arabic paraphrasing to improve Arabic-English statistical machine translation. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, pp. 10–21.

Salloum W. and Habash N. (2012). Elissa: A dialectal to standard Arabic machine translation system. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012): Demonstration Papers*, pp. 385–392.

Salloum W. and Habash N. (2013). Dialectal Arabic to English machine translation: Pivoting through modern standard Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Samih Y., Eldesouki M., Attia M., Darwish K., Abdelali A., Mubarak H. and Kallmeyer L. (2017a). Learning from relatives: Unified dialectal Arabic segmentation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 432–441.

Samih Y., Attia M., Eldesouki M., Abdelali A., Mubarak H., Kallmeyer L. and Darwish K. (2017b). A neural architecture for dialectal Arabic segmentation. In *Proceedings of the Third Arabic Natural Language Processing Workshop*. Valencia, Spain: Association for Computational Linguistics, pp. 46–54.

Sawaf H. (2010). Arabic dialect handling in hybrid machine translation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

Schone P. and Jurafsky D. (2000). Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of CoNLL-2000 and LLL-2000*, pp. 67–72.

Stallard D., Devlin J., Kayser M., Lee Y.K. and Barzilay R. (2012). Unsupervised morphology rivals supervised morphology for Arabic MT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, pp. 322–327.

Stolcke A. (2002). SRILM an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing*.

Utiyama M. and Isahara H. (2007). A comparison of pivot methods for phrase-based statistical machine translation. In *HLT-NAACL*, pp. 484–491.

Zalmout N. and Habash N. (2017a). Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 704–713.

Zalmout N. and Habash N. (2017b). Optimizing tokenization choice for machine translation across multiple target languages. *The Prague Bulletin of Mathematical Linguistics*, **108**(1), 257–269.

Zalmout N. and Habash N. (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*. Florence, Italy: Association for Computational Linguistics, pp. 1775–1786.

Zbib R., Malchiodi E., Devlin J., Stallard D., Matsoukas S., Schwartz R., Makhoul J., Zaidan O.F. and Callison-Burch C. (2012). Machine translation of Arabic dialects. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 49–59.

Zhang X. (1998). Dialect MT: A case study between Cantonese and Mandarin. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, ACL 1998*, pp. 1460–1464.

Cite this article: Salloum W and Habash N (2022). Unsupervised Arabic dialect segmentation for machine translation. *Natural Language Engineering* **28**, 223–248. https://doi.org/10.1017/S1351324920000455