A Probabilistic Morphological Analyzer for Syriac

Peter McClanahan, George Busby, Robbie Haertel, Kristian Heal †,
Deryle Lonsdale‡, Kevin Seppi, Eric Ringger
Department of Computer Science, ‡Department of Linguistics,
†Center for the Preservation of Ancient Religious Texts (CPART)
Brigham Young University
Provo, Utah 84604 USA
http://nlp.cs.byu.edu/

Abstract

We define a probabilistic morphological analyzer using a data-driven approach for Syriac in order to facilitate the creation of an annotated corpus. Syriac is an under-resourced Semitic language for which there are no available language tools such as morphological analyzers. We introduce novel probabilistic models for segmentation, dictionary linkage, and morphological tagging and connect them in a pipeline to create a probabilistic morphological analyzer requiring only labeled data. We explore the performance of models with varying amounts of training data and find that with about 34,500 labeled tokens, we can outperform a reasonable baseline trained on over 99,000 tokens and achieve an accuracy of just over 80%. When trained on all available training data, our joint model achieves 86.47% accuracy, a 29.7% reduction in error rate over the baseline.

1 Introduction

Our objective is to facilitate the annotation of a large corpus of classical Syriac (referred to simply as "Syriac" throughout the remainder of this work). Syriac is an under-resourced Western Semitic language of the Christian Near East and a dialect of Aramaic. It is currently employed almost entirely as a liturgical language but was a true spoken language up until the eighth century, during which time many prolific authors wrote in Syriac. Even today there are texts still being composed in or translated into Syriac. By automatically annotating these texts with linguistically useful information, we will facilitate systematic study by scholars of Syriac, the Near East, and Eastern Christianity. Furthermore, languages that are linguistically similar to Syriac (e.g., Arabic and Hebrew) may benefit from the methodology presented here.

Our desired annotations include morphological segmentation, links to dictionary entries, and morphological attributes. Typically, annotations of this kind are made with the assistance of language tools, such as morphological analyzers, segmenters, or part-of-speech (POS) taggers. Such tools do not exist for Syriac, but some labeled data does exist: Kiraz (1994) compiled an annotated version of the Peshitta New Testament (1920) and a concordance thereof. We aim to replicate this kind of annotation on a much larger scale with more modern tools, building up from the labeled New Testament data, our only resource. Motivated by this state of affairs, our learning and annotation framework requires only labeled data.

We approach the problem of Syriac morphological annotation by creating five probabilistic sub-models that can be trained in a supervised fashion and combined in a joint model of morphological annotation. We introduce novel algorithms for segmentation, dictionary linkage, and morphological tagging. We then combine these sub-models into a joint n-best pipeline. This joint model outperforms a strong, though naïve, baseline for all amounts of training data over about 9,900 word tokens.

1.1 Syriac Background

Since Syriac is an abjad, its writing system does not require vowels. As a dialect of Aramaic, it is an inflected language with a templatic (non-concatenative) morphology, based on a system of triliteral consonantal roots, with prefixes, suffixes, infixes, and enclitic particles. Syriac is written from

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 810–820,
MIT, Massachusetts, USA, 9-11 October 2010.

right to left. For the purposes of this work, all Syriac is transliterated according to the Kiraz (1994) transliteration[1] and is written left-to-right whenever transliterated; the Syriac appearing in the Serto script in this paper is shown right-to-left.

Since there is no standardized nomenclature for the parts of a Syriac word, we define the following terms to facilitate the definitions of segmentation, dictionary linkage, and morphological tagging:

- word token - contiguous characters delimited by whitespace and/or punctuation
- stem - an inflected form of the baseform and the main part of the word to which prefixes and suffixes can be attached; the affixes do not inflect the stem but include prepositions, object suffixes, and enclitic pronouns
- baseform - the dictionary citation form; also known as a lexeme or lemma
- root - the form from which the baseform is derived

To clarify, we will use an example word token ܠܡܠܟܟܘܢ, *LMLCCON*, which means "to your (masculine plural) king". For this word, the stem is ܡܠܟ, *MLC*; the baseform is ܡܠܟܐ, *MLCA* "king"; and the root is ܡܠܟ, *MLC*. To clarify, note that the word token (including the stem) can be spoken and written with vowels as diacritics; however, since the vowels are not written in common practice and since most text does not include them, this work omits any indication of vowels. Furthermore, the stem is an inflected baseform and does not necessarily form a word on its own. Also, the (unvocalized) stem and root are not necessarily identical. In Syriac, the same root ܡܠܟ, *MLC* is the foundation for other words such as promise, counsel, deliberate, reign, queen, kingdom, and realm.

1.2 Sub-tasks

Segmentation, or tokenization as it is sometimes called (e.g., Habash and Rambow, 2007), is the process of dividing a word token into its prefix(es) (if any), a stem, and a suffix (if any). For Syriac, each word token consists of exactly one stem, from zero to three prefixes, and zero or one suffix. Each prefix is exactly one character in length. Segmentation does not include the process of parsing the stem for its inflectional morphology; that step is handled separately in subsequent processes described below. While segmenting a Syriac word, we can handle all prefixes as a single unit. It is trivial to segment a prefix cluster into its individual prefixes (one character per prefix). Suffixes may be multiple characters in length and encode the morphological attributes of the suffix itself (not of the stem); the suffix usually encodes the object of the stem and has its own grammatical attributes, which we list later. As an example of token segmentation, for the word token ܠܡܠܟܟܘܢ, *LMLCCON*, the prefix is ܠ, *L* "to", the stem is ܡܠܟ, *MLC* "king", and the suffix is ܟܘܢ, *CON* "(masculine plural) your".

Dictionary linkage is the process of linking a stem to its associated baseform and root. In most Syriac dictionaries, all headwords are either baseforms or roots, and for a given word these are the only relevant entries in the dictionary. Each Syriac stem is derived from a baseform, and each baseform is derived from a root. There is ambiguity in this correspondence which can be caused by, among other things, homographic stems generated from different roots or even from homographic roots. As such, linkage may be thought of as two separate processes: (1) baseform linkage, where the stem is mapped to its most likely baseform; and (2) root linkage, where the baseform is mapped to its most likely root. For our example ܠܡܠܟܟܘܢ, *LMLCCON*, baseform linkage would map stem ܡܠܟ, *MLC* to baseform ܡܠܟܐ, *MLCA*, and root linkage would map baseform ܡܠܟܐ, *MLCA* to root ܡܠܟ, *MLC*.

Morphological tagging is the process of labeling each word token with its morphological attributes. Morphological tagging may be thought of as two separate tagging tasks: (1) tagging the stem and (2) tagging the suffix. For Syriac, scholars have defined for this task a set of morphological attributes consisting of twelve attributes for the stem and four attributes for the suffix. The attributes for the stem are as follows: grammatical category, verb conjugation, aspect, state, number, person, gender, pronoun type, demonstrative category, noun type, numeral type, and participle type. The morphological

[1] According to this transliteration all capital letters including *A* (ܐ, olaph) and *O* (ܘ, waw) are consonants. Additionally, the semi-colon (*;*), representing (ܝ, yod), is also a consonant.

Attribute	Value
Grammatical Category	noun
Verb Conjugation	N/A
Aspect	N/A
State	emphatic
Number	singular
Person	N/A
Gender	masculine
Pronoun Type	N/A
Demonstrative Category	N/A
Noun Type	common
Numeral Type	N/A
Participle Type	N/A

Table 1: The values for the morphological attributes of the stem ܡܠܟ, *MLC*, "king".

Attribute	Value
Gender	masculine
Person	second
Number	plural
Contraction	normal suffix

Table 2: The values for the morphological attributes of the suffix ܟܘܢ, *CON*, "(masculine plural) your".

attributes for the suffix are gender, person, number, and contraction. The suffix contraction attribute encodes whether the suffix is normal or contracted, a phonological process involving the attachment of an enclitic pronoun to a participle. These morphological attributes were heavily influenced by those used by Kiraz (1994), but were streamlined in order to focus directly on grammatical function. During morphological tagging, each stem is labeled for each of the stem attributes, and each suffix is labeled for each of the suffix attributes. For a given grammatical category (or POS), only a subset of the morphological attributes is applicable. For those morphological attributes (both of the stem and of the suffix) that do not apply, the correct label is "N/A" (not applicable). Tables 1 and 2 show the correct stem and suffix tags for the word ܡܠܟܟܘܢ, *LMLCCON*.

The remainder of the paper will proceed as follows: Section 3 outlines our approach. In Section 4, we describe our experimental setup; we present results in Section 5. Section 6 contrasts previous work with our approach. Finally, in Section 7 we briefly conclude and offer directions for future work.

2 The Syromorph Approach

Since lack language tools, we focus on automatically annotating Syriac text in a data-driven fashion based on the labeled data we have available. Since segmentation, linkage, and morphological tagging are not mutually independent tasks, we desire models for the sub-tasks to influence each other. To accommodate these requirements, we use a joint pipeline model (Finkel et al., 2006). In this section, we will first discuss this joint pipeline model, which we call **syromorph**. We then examine each of the individual sub-models.

2.1 Joint Pipeline Model

Our approach is to create a joint pipeline model consisting of a segmenter, a baseform linker, a root linker, a suffix tagger, and a stem tagger. Figure 1 shows the dependencies among the sub-models in the pipeline for a single word. Each sub-model (oval) has access to the data and predictions (rectangles) indicated by the arrows. For example, for a given word, the stem tagger has access to the previously predicted stem, baseform, root, and suffix tag. The baseform linker has access to the segmentation, most importantly the stem.

The training of **syromorph** is straightforward. Each of the individual sub-models is trained separately on the true labeled data. Features are extracted from the local context in the sentence. The local context consists first of predictions for the entire sentence from earlier sub-tasks (those sub-tasks upon which the sub-task in question depends). We created the dependencies shown in Figure 1 taking into account the difficulty of the tasks and natural dependencies in the language. In addition to the predictions for the entire sentence from previous sub-tasks, the local context also includes the previous o tags of the current sub-task, as the standard order o Markov model does. For example, when the stem tagger is being trained on a particular sentence, the local context consists of the words in the sentence, the predicted segmentation, baseform, root, and suffix tags for each word in the sentence, and additionally the labels for the previous o stems. To further elaborate

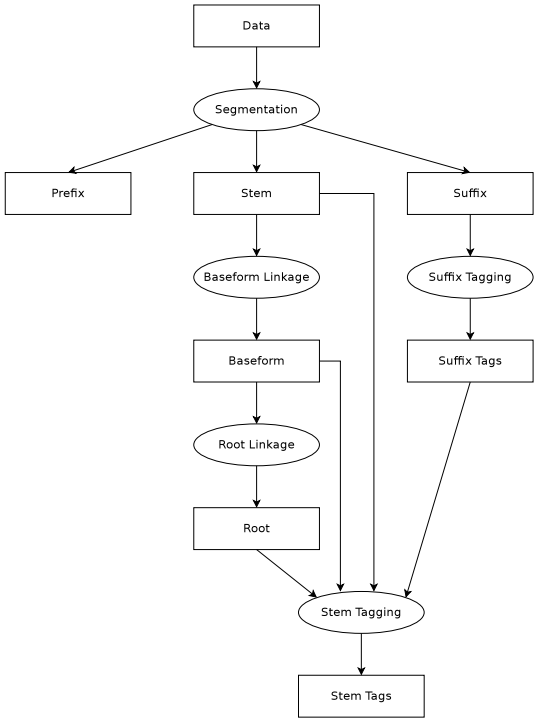

Figure 1: The **syromorph** model. Each rectangle is an input or output and each oval is a process employing a sub-model.

on the example, since features are extracted from the local context, for stem tagging we extract features such as current stem, previous stem, current baseform, previous baseform, current root, previous root, current suffix tags, and previous suffix tags. (Here, "previous" refers to labels on the immediately preceding word token.)

2.2 Segmentation

The **syromorph** segmentation model is a hybrid word- and consonant-level model, based on the model of Haertel et al. (2010) for data-driven diacritization. Each of our probabilistic sequence models is a maximum entropy Markov model (MEMM). Haertel et al. (2010) showed that the distribution over labels is different for known and words and rare words. In this work, we only consider words not seen in training (i.e., "unknown") to be rare. Following Haertel et al.'s (2010) model, a separate model is trained for each word type seen in training with the intent of choosing the best segmentation given that word. This approach is closely related to the idea of ambiguity classes mentioned in Hajič and Hladká (1998).

To handle unknown words, we back off to a consonant-level model. Our consonant-level segmentation model uses the notion of BI (Beginning and Inside) tags, which have proven successful in named-entity recognition. Since there are three labels in which we are interested (prefix, stem, and suffix), we apply the beginning and inside notion to each of them to create six tags: BEGINNING-PREFIX, INSIDE-PREFIX, BEGINNING-STEM, INSIDE-STEM, BEGINNING-SUFFIX, and INSIDE-SUFFIX. We train an MEMM to predict one of these six tags for each consonant. Furthermore, we constrain the decoder to allow only legal possible transitions given the current prediction, so that prefixes must come before stems and stems before suffixes. In order to capture the unknown word distributions, we train the consonant-level model on words occurring only once during training.

We call this word- and consonant-level segmentation model **hybrid**. As far as we are aware, this is a novel approach to segmentation.

2.3 Dictionary Linkage

For dictionary linkage, we divide the problem into two separate tasks: baseform linkage and root linkage. For both of these tasks, we use a hybrid model similar to that used for segmentation, consisting of a collection of separate MEMMs for each word type (either a stem or baseform, depending on the linker) and a model for unknown (or rare) words. For the unknown words, we compare two distinct approaches.

The first approach for unknown words is based on the work of Chrupała (2006), including the Morfette system. Instead of predicting a baseform given a stem, we predict what Chrupała calls a lemma-class. A lemma-class is the transformation specified by the minimum edit distance between the baseform (which he calls a lemma) and the stem. The transformation is a series of tuples, where each tuple includes (1) whether it was an insertion or deletion, (2) the letter inserted or deleted, and (3) the position of the insertion or deletion in the string (positions begin at zero). All operations are assumed to occur sequentially, as in Morfette. For example, the transformation of *XE;N* to *XEA* would proceed as follows: delete *;* from position 2, insert *A* into position 2, delete *N* from position 3.

In **hybrid-morfette** baseform linkage (respectively, root linkage), we predict a lemma-class (i.e., transformation) for each baseform (respectively, root). The predicted transformation is then applied to the stem (respectively, baseform) in order to construct the actual target baseform (respectively, root). The advantage to this method is that common transformations are grouped into a single class, thereby allowing the model to generalize and adequately predict baseforms (and roots) that have not been seen during training, but whose transformations have been seen. This model is trained on all words in order to capture as many transformations as possible.

The second approach for unknown words, called **hybrid-maxent**, uses an MEMM trained on all words seen in training. Given a stem (respectively, baseform), this approach predicts only baseforms (respectively, roots) that were observed in training data. Thus, this method has a distinct disadvantage when it comes to predicting new forms. This approach corresponds directly to the approach to handling unknown -words by Toutanova and Manning (2000) for POS tagging.

With regard to baseform and root linkage, we do not use the dictionary to constrain possible baseforms or roots, since we make no initial assumptions about the completeness of a dictionary.

2.4 Morphological Tagging

For morphological tagging, we break the task into two separate tasks: tagging the suffix and tagging the stem. Since there are a number of values that need to be predicted, we define two ways to approach the problem. We call the first approach the monolithic approach, in which the label is the concatenation of all the morphological attribute values. Table 3 illustrates the tagging of an example sentence: the stem tag and suffix tag columns contain the monolithic tags for stem tagging and suffix tagging. We use an MEMM to predict a monolithic tag for each stem or suffix and call this model **maxent-mono**. No co-occurrence restrictions among related or complementary morphological tags are directly enforced. Co-occurrence patterns are observed in the data, learned, and encoded in the models of the tagging process. It is worth noting further that constraints provided by the baseforms – predicted by dictionary linkage – on the morphological attributes are likewise not directly enforced. Enforcement of such constraints would require an infusion of expert knowledge into the system.

The second approach is to assume that morphological attributes are independent of each other. We call this the independent approach. Here, each tag is predicted by a tagger for a single morphological attribute. For example, the gender model is ignorant of the other 11 sub-tags during stem tagging. Using its local context (which does not include other stem sub-tags), the model predicts the best gender for a given word. The top prediction of each of these taggers (12, for stem tagging) is then combined naïvely with no notion of what combinations may be valid or invalid. We use MEMMs for each of the single-attribute taggers. This model is called **maxent-ind**.

2.5 Decoding

Our per-task decoders are beam decoders, with beam-size b. In particular, we limit the number of per-stage back-pointers to b due to the large size of the tagset for some of our sub-models. Although Viterbi decoding produces the most probable label sequence given a sequence of unlabeled words, it is potentially intractible on our hybrid models due to the unbounded dependence on previous consonant-level decisions. Our beam decoders produce a good approximation when tuned properly.

Decoding in **syromorph** consists of extending the per-task decoders to allow transitions from each sub-model to the next sub-model in the pipe. For example, in our pipeline, the first sub-model is segmentation. We predict the top n segmentations for the sentence (i.e., sequences of segmentations), where n is the number of transitions to maintain between each sub-task. Then, we run the remaining sub-tasks with each of the n sequences as a possible context. After each sub-task is completed, we narrow the number of possible contexts back to n.

We swept b and n for various values, and found $b = 5$ and $n = 5$ to be good values that balanced between accuracy and time; larger values saw only minute gains in accuracy.

Word	Transliteration	Pre.	Stem	Suffix	Baseform	Root	Suff. Tags	Stem Tags
ܐܒܕܗ	OEBDT	O	EBDT		EBD	EBD	0000	011012200000
ܗܘ	ANON		ANON		HO	HO	0000	300023222000
ܠܐܠܗܝ	LALHN	L	ALH	N	ALHA	ALH	1011	200310200200
ܡܠܟܘܬܐ	MLCOTA		MLCOTA		MLCOTA	MLC	0000	200310300200
ܘܟܗܢܐ	OCHNA	O	CHNA		CHNA	CHN	0000	200320200200
ܘܡܠܟܐ	OMLCA	O	MLCA		MLCA	MLC	0000	200320200200

Table 3: Part of a labeled Syriac sentence ܘܡܠܟܐ ܘܟܗܢܐ ܠܐܠܗܝ ܗܘ ܐܒܕܗ, "And you have made them a kingdom and priests and kings for our God." (Revelation 5:10)

3 Experimental Setup

We are using the Syriac Peshitta New Testament in the form compiled by Kiraz (1994).[2] This data is segmented, annotated with baseform and root, and labeled with morphological attributes. Kiraz and others in the Syriac community refined and corrected the original annotation while preparing a digital and print concordance of the New Testament. We augmented Kiraz's version of the data by segmenting suffixes and by streamlining the tagset. The dataset consists of 109,640 word tokens.

Table 3 shows part of a tagged Syriac sentence using this tagset. The suffix and stem tags consist of indices representing morphological attributes. In the example sentence, the suffix tag 1011 represents the values "masculine", "N/A", "plural", "normal suffix" for the suffix attributes of gender, person, number, and contraction. Each value of 0 for each stem and suffix attribute represents a value of "N/A", except for that of grammatical category, which always must have a value other than "N/A". Therefore, the suffix tag 0000 means there is no suffix.

For the stem tags, the attribute order is the same as that shown in Table 1 from top to bottom. The following describes the interpretation of the stem values represented in Table 3. Grammatical category values 0, 2, and 3 represent "verb", "noun", and "pronoun", respectively. (Grammatical category has no "N/A" value.) The verb conjugation value 1 represents "peal conjugation". Aspect value 1 represents "perfect". State value 3 represents "emphatic". Number values 1 and 2 represent "singular" and "plural". Person values 2 and 3 represent "second" and "third" person. Gender values 2 and 3 represent "masculine" and "feminine". Pronoun type value 2 represents "demonstrative". Demonstrative category value 2 represents "far". Finally, noun type 2 represents "common". The last two columns of 0 represent "N/A" for numeral type and particle type.

We implement five sub-tasks: segmentation, baseform linkage, root linkage, suffix tagging, and stem tagging. We compare each sub-task to a naïve approach as a baseline. In addition to desiring good sub-models, we also want a joint pipeline model that significantly outperforms the naïve joint approach, which is formed by using each of the following baselines in the pipeline framework.

The baseline implementation of segmentation is to choose the most-frequent label: for a given word, the baseline predicts the segmentation with which that word appeared most frequently during training. For unknown words, it chooses the largest prefix and largest suffix that is possible for that word from the list of prefixes and suffixes seen during training. (This naïve baseline for unknown words does not take into account the fact that the stem is often at least three characters in length.)

For dictionary linkage, the baseline is similar: both baseform linkage and root linkage use the most-frequent label approach. Given a stem, the baseline baseform linker predicts the baseform with which the stem was seen most frequently during training; likewise, the baseline root linker predicts the root from the baseform in a similar manner. For the unknown stem case, the baseline baseform linker predicts the baseform to be identical to the stem. For the unknown baseform case, the baseline root linker predicts a root identical to the first three consonants of the baseform, since for Syriac the root is exactly

[2]The Way International, a Biblical research ministry, annotated this version of the New Testament by hand and required 15 years to do so.

three consonants in a large majority of the cases.

The baselines for stem and suffix tagging are the most-frequent label approaches. These baselines are similar to **maxent-mono** and **maxent-ind**, using the monolithic and independent approaches used by **maxent-mono** and **maxent-ind**. The difference is that instead of using maximum entropy, the naïve most-frequent approach is used in its place.

The joint baseline tagger uses each of the component baselines in the n-best joint pipeline framework. Because this framework is modular, we can trivially swap in and out different models for each of the subtasks.

4 Experimental Results

Since we are focusing on under-resourced circumstances, we sweep the amount of training data and produce learning curves to better understand how our models perform in such circumstances. For each point in our learning curves and for all other evaluations, we employ ten-fold cross-validation. The learning curves use the chosen percentage of the data for training and a fixed-size test set from each fold and report the average accuracy.

The reported task accuracy requires the entire output for that task to be correct in order to be counted as correct. For example, during stem tagging, if one of the sub-tags is incorrect, then the entire tag is said to be incorrect. Furthermore, for **syromorph**, the outputs of every sub-task must be correct in order for the word token to be counted as correct.

Moving beyond token-level metrics, in order to understand performance of the system at the level of individual decisions (including N/A decisions), we compute decision-level accuracy: we call this metric **total-decisions**. For the **syromorph** method reported here, there are a total of 20 decisions: 2 for segmentation (prefix and suffix boundaries), 1 for baseform linkage, 1 for root linkage, 4 for suffix tagging, and 12 for stem tagging. This accuracy helps us to assess the number of decisions a human annotator would need to correct, if data were pre-annotated by a given model. Excluding N/A decisions, we compute per-decision coverage and accuracy. These metrics are called **applicable-coverage** and **applicable-accuracy**.

We show results on both the individual sub-tasks and the entire joint task. Since previous sub-tasks can adversely affect tasks further down in the pipeline, we evaluate the sub-models by placing them in the pipeline with other (simulated) sub-models that correctly predict every instance. For example, when testing a root linker, we place the root linker to be evaluated in the pipeline with a segmenter, baseform linker, and taggers that return the correct label for every prediction. This gives an upper-bound for the individual model, removes the possibility of error propagation, and shows how well that model performs without the effects of the other models in the pipeline.

For our results, unknown accuracy is the accuracy of unknown instances, specific to the task, at training time. In the case of baseform linkage, for example, a stem is considered unknown if that stem was not seen during training. It is therefore possible to have a known word with an unknown stem and vice versa. As in other NLP problems, unknown instances are a manifestation of training data sparsity.

4.1 Baseline Results

Table 4 is grouped by sub-task and reports the results of each of the baseline sub-tasks in the first row of each group. Each of the baselines performs surprisingly well. The accuracies of the baselines for most of the tasks are high because the ambiguity of the labels given the instance is quite low: the average ambiguity across word types for segmentation, baseform linkage, root linkage, suffix tagging, and stem tagging are 1.01, 1.05, 1.02, 1.35, and 1.47, respectively.

Preliminary experiments indicated that if we had trained a baseline model using a single prediction (a monolithic concatenation of the predictions for all tasks) per token rather than separating the tasks, the baseline tagging accuracy would have been lower. Note that the unknown tagging accuracy for the monolithic suffix tagger is not applicable, because there were no test suffixes that were not seen during training.

4.2 Individual Model Results

Table 4 also shows the results for the individual models. In the table, SEG, BFL, RTL, SUFFIX, and STEM represent segmentation, baseform linkage, root linkage, suffix tagging, and stem tagging,

	Model	Total	Known	Unk
SEG	baseline	96.75	99.64	69.11
	hybrid	**98.87**	**99.70**	**90.83**
BFL	baseline	95.64	98.45	22.28
	hybrid-morfette	**96.19**	98.05	**78.40**
	hybrid-maxent	**96.19**	**99.15**	67.86
RTL	baseline	98.84	**99.56**	80.20
	hybrid-morfette	**99.05**	99.44	**88.86**
	hybrid-maxent	98.34	99.45	69.30
SUFFIX	mono. baseline	98.75	98.75	N/A
	ind. baseline	96.74	98.78	**0.01**
	maxent-mono	**98.90**	**98.90**	N/A
	maxent-ind	**98.90**	**98.90**	N/A
STEM	mono. baseline	83.08	86.26	0.01
	ind. baseline	53.24	86.90	0.00
	maxent-mono	**89.48**	**92.87**	**57.04**
	maxent-ind	88.43	90.26	40.59

Table 4: Word-level accuracies for the individual sub-models used in the **syromorph** approach.

Model	Total	Known	Unk
Baseline	80.76	85.74	28.07
Morfette Monolithic	85.96	89.85	**44.86**
Maxent Monolithic	**86.47**	**90.77**	40.93

Table 5: Word-level accuracies for various joint **syromorph** models.

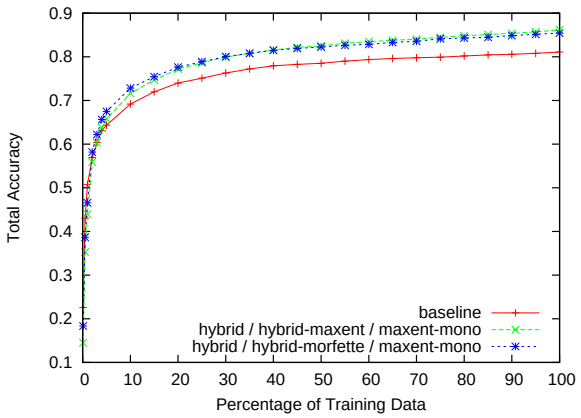

Figure 2: The total accuracy of the joint model.

respectively. Even though the baselines were high, each individual model outperformed its respective baseline, with the exception of the root linker. Two of the most interesting results are the known accuracy of the baseform linkers **hybrid-maxent** and **hybrid-morfette**. As hybrid models, the difference between them lies only in the treatment of unknown words; however, the known accuracy of the morfette model drops fairly significantly. This is due to the unknown words altering the weights for features in which those words occur. For instance, if the previous word is unknown and a baseform that was never seen was predicted, then the weights on the next word for all features that contain that unknown word will be quite different than if that previous word were a known word.

It is also worth noting that the stem tagger is by far the worst model in this group of models, but it is also the most difficult task. The largest gains in improving the entire system would come from focusing attention on that task.

4.3 Joint Model Results

Table 5 shows the accuracies for the joint models. The joint model incorporating "maxent" variants performs best overall and on known cases. The joint model incorporating the "morfette" variants performs best on unknown cases.

Decision-level metrics for the SEG:**hybrid** / BFL and RTL:**hybrid-maxent** / SUFFIX and STEM:**maxent-mono** model are as follows: for **total-decisions**, the model achieves an accuracy of 97.08%, compared to 95.50% accuracy for the baseline, amounting to a 35.11% reduction in error rate over the baseline; for **applicable-coverage** and **applicable-accuracy** this model achieved 93.45% and 93.81%, respectively, compared to the baseline's 90.03% and 91.44%.

Figures 2, 3, and 4 show learning curves for total, known, and unknown accuracies for the joint pipeline model. As can be seen in Figure 2, by the time we reach 10% of the training data, **syromorph** is significantly better than the baseline. In fact, at 35% of the training data, our joint pipeline model outperforms the baseline trained with all available training data.

Figure 3 shows the baseline performing quite well on known words with very low amounts of data. Since the x-axis varies the amount of training data, the meaning of "known" and "unknown" evolves as we move to the right of the graph; consequently, the

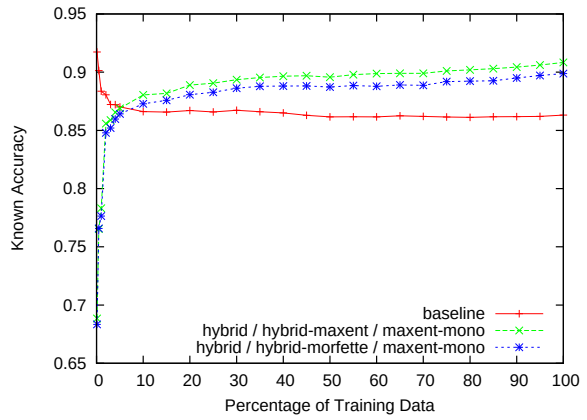

Figure 3: The accuracy of the joint model on known words.

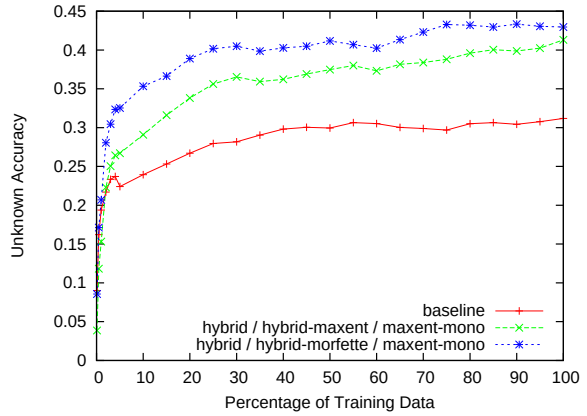

Figure 4: The accuracy of the joint model on unknown words.

left and right sides of the graph are incomparable. When the percentage of training data is very low, the percentage of unknown words is high, and the number of known words is relatively low. On this dataset, the more frequent words tend to be less ambiguous, giving the most-frequent taggers an advantage in a small random sample. For this reason, the baseline performs very well on known accuracy with lower amounts of training data.

Figure 4 clearly shows that **hybrid-morfette** linkers outperform **hybrid-maxent** linkers on unknown words. However, Figures 2- 4 show that **hybrid-morfette**'s advantage on unknown words is counteracted by its lower performance on known words; therefore, it has slightly lower overall accuracy than **hybrid-maxent**.

5 Related Work

The most closely related work to our approach is the Morfette tool for labeling inflectional morphology (Chrupała et al., 2008). Chrupała et al. created a tool that labels Polish, Romanian, and Spanish with morphological information as well as baseforms. It is a supervised learning approach that requires data labeled with both morphological tags and baseforms. This approach creates two separate models (a morphological tagger and a lemmatizer) and combines the decoding process in order to create a joint model that predicts both morphological tags and the baseform. Morfette uses MEMMs for both models and has access to predicted labels in the feature set. Reported accuracy rates are 96.08%, 93.83%, and 81.19% for joint accuracy on datasets trained with fewer than 100,000 tokens for Romanian, Spanish, and Polish, respectively. The major difference between this work and ours is the degree of morphological analysis required by the languages. Chrupała et al. neglect segmentation, a task not as intuitive for their languages as it is for Syriac. These languages also require only linkage to a baseform, as no root exists.

Also closely related is the work of Daya, Roth, and Wintner (2008) on Hebrew. The authors use the notion of patterns into which root consonants are injected to compose Semitic words. They employ linguistic knowledge (specifically, lists of prefixes, suffixes, and "knowledge of word-formation processes" combined with SNoW, a multi-class classifier that has been shown to work well in other NLP tasks. The major difference between this approach and the method presented in this paper is that this method does not require the extra knowledge required to encode word-formation processes. A further point of difference is our use of hybrid word- and consonant-level models, after Haertel et al. (2010). Their work builds on the work of Shacham and Wintner (2007), which is also related to that of Habash and Rambow, described below.

Work by Lee et al. (2003) is the most relevant work for segmentation, since they segment Arabic, closely related to Syriac, with a data-driven approach. Lee et al. use an unsupervised algorithm bootstrapped with manually segmented data to learn the segmentation for Arabic without any additional language re-

sources. At the heart of the algorithm is a word-level trigram language model, which captures the correct weights for prefixes and suffixes. They report an accuracy of 97%. We opted to use our own segmenter because we felt we could achieve higher accuracy with the **hybrid** segmenter.

Mohamed and Kübler (2010a, 2010b) report on closely related work for morphological tagging. They use a data-driven approach to find the POS tags for Arabic, using both word tokens and segmented words as inputs for their system. Although their segmentation performance is high, they report that accuracy is lower when first segmenting word tokens. They employ TiMBL, a memory-based learner, as their model and report an accuracy of 94.74%.

Habash and Rambow (2005) currently have the most accurate approach for Arabic morphological analysis using additional language tools. They focus on morphological disambiguation (tagging), given morphological segmentation in the output of the morphological analyzer. For each word, they first run it through the morphological analyzer to reduce the number of possible outputs. They then train a separate Support Vector Machine (SVM) for each morphological attribute (ten in all). They look at different ways of combining these outputs to match an output from the morphological analyzer. For their best model, they report an overall tag accuracy of 97.6%.

Others have used morphological analyzers and other language tools for morphological disambiguation coupled with segmentation. The following works exemplify this approach: Diab et al. (2004) use a POS tagger to jointly segment, POS tag, and chunk base-phrases for Arabic with SVMs. Kudo et al. (2004) use SVMs to morphologically tag Japanese. Smith et al. (2005) use SVMs for segmentation, lemmatization, and POS tagging for Arabic, Korean, and Czech. Petkevič (2001) use a morphological analyzer and additional simple rules for morphological disambiguation of Czech. Mansour et al. (2007) and Bar-haim et al. (2008) both use hidden Markov models to POS tag Hebrew, with the latter including segmentation as part of the task.

For Syriac, a morphological analyzer is not available. Kiraz (2000) created a Syriac morphological analyzer using finite-state methods; however, it was developed on outdated and now inaccessible equipment and is no longer working or available to us.

6 Conclusions and Future Work

We have shown that we can effectively model segmentation, linkage to headwords in a dictionary, and morphological tagging using a joint model called **syromorph**. We have introduced novel approaches for segmentation, dictionary linkage, and morphological tagging, and each of these approaches has outperformed its corresponding naïve baseline. Furthermore, we have shown that for Syriac, a data-driven approach seems to be an appropriate way to solve these problems in an under-resourced setting.

We hope to use this combined model for preannotation in an active learning setting to aid annotators in labeling a large Syriac corpus. This corpus will contain data spanning multiple centuries and a variety of authors and genres. Future work will require addressing issues encountered in this corpus. In addition, there is much to do in getting the overall tag accuracy closer to the accuracy of individual decisions. We leave further feature engineering for the stem tagger and the exploration of possible new morphological tagging techniques for future work.

Finally, future work includes the application of the **syromorph** methodology to other under-resourced Semitic languages.

Acknowledgments

We would like to thank David Taylor of the Oriental Institute at Oxford University for collaboration on the design of the simplified tagset. We also recognize the assistance of Ben Hansen of BYU on a subset of the experimental results. Finally, we would like to thank the anonymous reviewers for helpful guidance.

References

Roy Bar-haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of modern hebrew text. *Natural Language Engineering*, 14(2):223–251.

British and Foreign Bible Society, editors. 1920. *The New Testament in Syriac*. Oxford: Frederick Hall.

Grzegorz Chrupała, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. In *Procesamiento del Lenguaje Natural*, volume 37, pages 121 – 127.

Ezra Daya, Dan Roth, and Shuly Wintner. 2008. Identifying Semitic roots: Machine learning with linguistic constraints. *Computational Linguistics*, 34(3):429–448.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proceedings of the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, pages 149–152.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 53–56. Association for Computational Linguistics.

Robbie Haertel, Peter McClanahan, and Eric K. Ringger. 2010. Automatic diacritization for low-resource languages using a hybrid word and consonant cmm. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 519–527. Association for Computational Linguistics.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 483–490. Association for Computational Linguistics.

George Kiraz. 1994. Automatic concordance generation of Syriac texts. In R. Lavenant, editor, *VI Symposium Syriacum 1992*, pages 461–.

George Anton Kiraz. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. *Computational Linguistics*, 26:77–105.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of EMNLP*, pages 230–237.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. 2003. Language model based Arabic word segmentation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 399–406. Association for Computational Linguistics.

Saib Mansour, Khalil Sima'an, and Yoad Winter. 2007. Smoothing a lexicon-based pos tagger for Arabic and Hebrew. In *Semitic '07: Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages*, pages 97–103. Association for Computational Linguistics.

Emad Mohamed and Sandra Kübler. 2010a. Arabic part of speech tagging. In *Proceedings of the Seventh International Language Resources and Evaluation (LREC'10)*.

Emad Mohamed and Sandra Kübler. 2010b. Is Arabic part of speech tagging feasible without word segmentation? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 705–708. Association for Computational Linguistics.

Vladimír Petkevič. 2001. Grammatical agreement and automatic morphological disambiguation of inflectional languages. In *TSD '01: Proceedings of the 4th International Conference on Text, Speech and Dialogue*, pages 47–53. Springer-Verlag.

Danny Shacham and Shuly Wintner. 2007. Morphological disambiguation of Hebrew: a case study in classifier combination. In *Proceedings of EMNLP-CoNLL 2007, the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 475–482. Association for Computational Linguistics.

K. Toutanova and C. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*, pages 63–70.