2010-07-08

A Probabilistic Morphological Analyzer for Syriac

Peter J. McClanahan
Brigham Young University - Provo

A Probabilistic Morphological Analyzer for Syriac

Peter J. McClanahan

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Eric K. Ringger, Chair
Kevin D. Seppi
Daniel Zappala

Department of Computer Science

Brigham Young University

December 2010

ABSTRACT

A Probabilistic Morphological Analyzer for Syriac

Peter J. McClanahan

Department of Computer Science

Master of Science

We show that a carefully crafted probabilistic morphological analyzer significantly outperforms a reasonable, naïve baseline for Syriac. Syriac is an under-resourced Semitic language for which there are no available language tools such as morphological analyzers. Such tools are widely used to contribute to the process of annotating morphologically complex languages. We introduce and connect novel data-driven models for segmentation, dictionary linkage, and morphological tagging in a joint pipeline to create a probabilistic morphological analyzer requiring only labeled data. We explore the performance of this model with varying amounts of training data and find that with about 34,500 tokens, it can outperform the baseline trained on over 99,000 tokens and achieve an accuracy of just over 80%. When trained on all available training data, this joint model achieves 86.47% accuracy — a 29.7% reduction in error rate over the baseline.

Keywords: Segmentation, dictionary linkage, morphological tagging, Syriac, Semitic languages, probabilistic models, joint pipelines

ACKNOWLEDGMENTS

I am extremely grateful to my advisor, Eric Ringger, for helping me get started with this thesis, giving me guidance and direction, and helping to shape this work into what it has become. Thanks also goes to Kevin Seppi, who provided another stable support and helped guide and direct in much the same way Dr. Ringger did. Many thanks to Robbie Haertel, who spent numerous hours helping me fix the details, and who also provided countless good ideas and suggestions. George Busby's code provided a solid foundation with his joint pipeline framework. Without him, my contributions would not be possible.

My profound gratitude goes to Kristian Heal, without whom I would not have been introduced to the fascinating world of Syriac, and whose scholarly knowledge of the language helped immensely during my entire thesis. Thanks also goes to Deryle Lonsdale, who augmented my understanding and the course of my thesis with his contributions and linguistic insights. I am grateful to Jay McCarthy, who gave excellent feedback and quick turnarounds on drafts, helping me to gain a very clear picture of what to do. I am also grateful to Daniel Zappala for stepping in on short notice to be my third committee member.

Most of all, my wife, Lia, has been a steady source of encouragement through the ups and downs I have gone through. I could not have done this without her.

Contents

List of Figures

List of Tables

Chapter 1

Morphological Annotation

Scholars at the Neal A. Maxwell Institute for Religious Scholarship at BYU and at the Oriental Institute at Oxford University are jointly working on a project called the Comprehensive Syriac Corpus project, with the goal of creating a comprehensive, labeled corpus of classical Syriac. Syriac is an under-resourced Semitic language of the Christian Near East and a dialect of Aramaic. It is currently almost entirely a liturgical language but was a true spoken language up until the eighth century. During that time period many prolific authors wrote in Syriac, and even today there are texts still being composed in or translated into Syriac. The goal of the Comprehensive Syriac Corpus project is to annotate this corpus with information to facilitate systematic study of Syriac and its grammar. Such studies would be useful to linguists, Syriac students, and scholars of Syriac, the Near East, and Eastern Christianity.

Labeling Syriac (and other languages) can be extremely time consuming. The Way International Foundation, a Biblical research, teaching, and fellowship ministry, spent 15 years labeling the Syriac New Testament by hand. It was labeled with grammatical information similar to that desired for the comprehensive corpus (see Kiraz [1994]). The larger goal in which this thesis plays a part is to aid the creators of the corpus in developing a system by which human annotators can more efficiently label large corpora. Benefits are expected to come through utilization of machine-learned models that will enable annotators to correct the mistakes of the models rather than label each word from scratch. In addition, active learning will be a key in further reducing the cost of this endeavor. The scope of this thesis is to aid the creators of the corpus by developing a model that can be used to assist human annotation. This thesis is that a probabilistic data-driven frame-

Figure 1.1: An example of a Syriac word. This word is transliterated *LMLCCON* and means "to your king".

work can be used for segmentation, dictionary linkage, and morphological tagging for Syriac, and can significantly outperform a reasonable, naïve baseline. In the following sections we will define each part of this thesis statement and motivate its importance.

1.1 Syriac

Syriac is an ancient language of the Christian Near East, a Semitic language, and a dialect of Aramaic. As a Semitic language, Syriac is closely related to Arabic and Hebrew. It is written right to left, and its alphabet consists of 22 consonants. An example word is shown in Figure 1.1. This word means "to your (masculine plural) king" and is transliterated as *LMLCCON* according to the transliteration developed by Kiraz [1994]. In this transliteration, all capital letters and symbols are mapped to consonants, including *A* (olaph), *O* (waw), and the symbol of a semi-colon (*;*), representing yod. In order to simplify the preparation of this thesis, we transliterate all Syriac according to this mapping. Syriac is an abjad, which means that the writing system does not require vowels; in most cases vowels are omitted entirely. The data we use in this thesis does not contain vowels.

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

- word token - The contiguous characters delimited by whitespace. This is what most people would think of as a word.
- stem - The main part of the word token. It is to this that prefixes and suffixes can be attached.

- prefix - The characters placed before the stem.
- suffix - The characters placed after the stem.
- baseform - The form from which the stem is derived.[1]
- root - The form from which the baseform is derived. This is typically the triliteral root.[2]

To clarify, we will continue with the example word token *LMLCCON*, which means "to your (masculine plural) king". For this word, the stem is *MLC*; the baseform is *MLCA* "king"; and the root is *MLC*. In English, the stem is an inflected baseform, and is not necessarily a word. For example, the stem of "producing" is "produc". The root has no English equivalent; however, the same root *MLC* is the foundation for other Syriac words in addition to king, such as promise, counsel, deliberate, reign, queen, kingdom, and realm.

1.2 Segmentation

Segmentation is the process of dividing a word token into its prefixes, suffixes, and stem. For Syriac, each word token has exactly one stem. This word token may have from zero to three prefixes, with each prefix being exactly one character in length. Since it is trivial to take a group of prefixes and divide them into one-character elements, we treat the concatenated group of prefixes as a single prefix. Each word token has zero or one suffix. The suffixes are more complex than the prefixes. They may be multiple characters in length, and the set of possible suffixes is much larger than the set of possible prefixes.

Suffixes also encode the morphological attributes of suffix gender, suffix person, suffix number, and suffix contraction. The attribute of suffix contraction encodes whether the suffix is normal or contracted. Morphological attributes of a word are used to describe its morphemes (semantic meaning) and other syntactic functions. For example, the English word *unbreakable* has

[1] Linguistic derivation is the formation of one word form from another word (e.g., glorify and glory, happiness and happy).

[2] Almost all words in Syriac are derived from a triliteral root, a root consisting of 3 consonants.

three morphemes: (1) *un*, (2) *break*, and (3) *able*. Each of these parts would be the value of some morphological attribute.

Segmentation, therefore, takes in a word token as input and produces, as the output, the prefixes, stem, and suffix. For the word token *LMLCCON*, the input is the word token itself, and segmentation produces a prefix *L*, a stem *MLC*, and a suffix *CON*. Another way of viewing the segmentation process is the identification of the prefix-stem boundary and the stem-suffix boundary.

1.3 Dictionary Linkage

Dictionary linkage is the process of linking a stem to its associated baseform and root. Each Syriac stem has exactly one baseform from which it is derived, and each baseform has exactly one root from which it is derived. There may be many stems linked to each baseform, and many baseforms may link to one root. This linkage may be thought of as two separate processes: (1) a baseform linkage, where the stem is mapped to a baseform and (2) a root linkage, where the baseform is mapped to the root.

The baseform linkage task takes a stem as input and produces a baseform. Root linkage takes a baseform and produces a root. For the example word *LMLCCON*, the baseform linker takes the stem *MLC* and returns the baseform *MLCA*. The root linker takes *MLCA* and returns the root *MLC*. As shown in this example, the transition from the stem to the baseform or from baseform to root often requires deleting and inserting characters, in this case, "*A*".

1.4 Morphological Tagging

Morphological tagging is the process of labeling each word token with its morphological attributes. For Syriac, scholars have defined this set of morphological attributes, consisting of 12 attributes for each stem and 4 attributes for each suffix. A list of the morphological attributes and their possible values is found in Table 1.3. Each prefix has exactly the same morphological information (namely, that it is a prefix), and is therefore trivial to label. As with dictionary linkage, this problem may be thought of as two separate tagging tasks: tagging the stem and tagging the suffix.

Tag	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 1.1: The values for the morphological tags of the stem *MLC* (king).

Tag	Value
Suffix Gender	masculine
Suffix Person	second
Suffix Number	plural
Suffix Contraction	normal suffix

Table 1.2: The values for the morphological tags of the suffix *CON* (your, masculine plural).

Morphological Tag	Values
Grammatical Category	verb, participle, noun, pronoun, number, adjective, particle, adverb, idiom
Verb Conjugation	n/a, peal, ethpeal, pael, ethpael, aphel, ettaphal, shaphel, eshtaphal, saphel, estaphal, p, ethp, palpel, ethpalpal
Aspect	n/a, perfect, imperfect, imperative, infinitive, participle
State	n/a, absolute, construct, emphatic
Number	n/a, singular, plural
Person	n/a, first, second, third
Gender	n/a, common, masculine, feminine
Pronoun Type	n/a, personal, demonstrative, interrogative
Demonstrative Category	n/a, near, far
Noun Type	n/a, proper, common
Numeral Type	n/a, cardinal, ordinal, cipher
Participle Type	n/a, active, passive
Suffix Contraction	n/a, normal suffix, contracted suffix
Suffix Gender	n/a or common, masculine, feminine
Suffix Person	n/a, first, second, third
Suffix Number	n/a or singular, plural

Table 1.3: Possible values for the morphological attributes of Syriac words.

The stem tagger produces all 12 morphological attributes (or tags) for the given stem. The suffix tagger produces the 4 tags for a given suffix. For the word *LMLCCON*, the stem tagger takes *MLC* and produces the attributes found in Table 1.1. Table 1.2 shows the results for the suffix tagger, which takes the suffix *CON* as input.

The values for verb conjugation, aspect, person, pronoun type, demonstrative category, numeral type, and particle type in Table 1.1 are "N/A." This is because certain morphological attributes are defined only for certain grammatical categories. Verb conjugation, for example, is defined only for verbs. Since this particular word is a noun, the value for that attribute is "N/A". Table 1.4 shows the possible morphological attributes for each grammatical category.

1.5 Data-Driven Framework

We refer to the implementations that produce segmentation, dictionary linkage, and morphological tagging as segmenters, linkers, and taggers, respectively. A data-driven framework means that the

Attribute	Verb	Participle	Noun	Pron.	Num.	Adj.	Particle	Adv.	Idiom
Verb Conjugation	X	X	X			X	X		
Aspect	X	X	X			X	X		
State	X	X	X		X	X	X	X	X
Number	X	X	X	X	X	X	X	X	X
Person	X	X	X	X		X	X		
Gender	X	X	X	X	X	X	X	X	X
Pronoun Type				X					
Demonstrative Category				X					
Noun Type			X						
Numeral Type					X				
Participle Type	X	X	X			X	X		

Table 1.4: Representation of the applicable morphological attributes for the stem tagger (X's mean the field applies for the given grammatical category)

segmenter, linkers, and taggers are built from data already annotated with segmentation, linkage, and tagging information. Aside from this annotated data, the framework does not require additional language tools that most previous works use. In a machine learning framework, this means that the implementations (segmenter, linkers, and taggers) are models that are trained from annotated data and evaluated on test data.

1.6 The Low-Resource Scenario

Since Syriac is an under-resourced language, data is scarce. To compensate, as part of the Comprehensive Syriac Concordance project we are investigating the process of active learning for gathering annotated data. Active learning is a process where a model chooses which items (sentences) the humans should annotate. The learned model is used to select and request annotations for maximal benefit in its learning process. During the annotation process, automatic pre-annotation, or the predictions from the model developed in this thesis, will be shown to the annotators in order to expedite their annotation process. For active learning, as the amount of annotated data increases, the models become increasingly accurate, thus increasing the quality of the pre-annotation.

We explore the performance of models with varying amounts of training data in order to gain a better understanding of how well the models perform with different amounts of data. Although learning curves (line graphs of accuracy plotted against increasing amounts of training data) are not not directly comparable with the performance of active learning, they roughly show how these models could perform in an active learning setting.

1.7 Validation

We test the performance of the probabilistic morphological analyzer (which we call **prob-morph**) in a joint pipeline framework, where the models (segmenter, linkers, and taggers) are combined to create a single joint model. This joint model is explained in more detail in chapter 3. The reasonable baseline model is created by inserting a baseline model for each of the **prob-morph** sub-models into the same joint pipeline. By comparing these two approaches (baseline and **prob-morph**), we will demonstrate that the **prob-morph** approach can significantly outperform the naïve approach.

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 the largest suffix that is possible for that word from the list of prefixes and suffixes seen during training.

A similar baseline exists for dictionary linkage. It is the same most-frequent model with an altered unknown word approach. For unknown words, this baseline predicts a baseform equal to that of the stem. The baseline root linker uses a most-frequent model to predict roots for those baseforms that were seen during training. The approach for unknown words for this model predicts the first three characters of the baseform as the root, since almost all roots contain only three characters.

For both the suffix and stem taggers, there are two most-frequent tagger baselines. The first, which we call the monolithic baseline, treats the morphological attributes (all seven for the stem tagger) as a unit. For the stem tagger in our example, the monolithic tag is

noun#N/A#N/A#emphatic#singular#N/A#masculine#N/A#N/A#common#N/A#N/A, where # is the attribute delimiter. The most-frequent tagger then works as expected using the monolithic tag as a basic unit. For unknown words, the model predicts the attributes to be those from the monolithic tag that were seen most in training.

The second baseline we call an independent baseline because each morphological attribute is assumed to be independent. A separate most-frequent tagger is trained for each attribute with the task of predicting solely that attribute. This model is ignorant of the other morphological attributes. The best prediction of each of these taggers (12 for stem tagging) is combined naïvely with no notion of what combinations may be valid or invalid.

In chapter 2, we review related work. Chapter 3 consists of a separate, stand-alone paper submitted for publication at the 2010 Conference on Empirical Methods on Natural Language Processing (EMNLP). This paper contains the heart of this thesis and presents our solution and methodology. As this is a stand-alone publication, parts of chapter 1 and chapter 2 will be repeated, as necessary. Chapter 4 contains additional and more comprehensive results that did not fit in chapter 3 including statistical validation. Chapter 5 offers conclusions and makes recommendations for future work.

Chapter 2

Related Work

There is much previous work related to the different aspects of this task, namely segmentation, dictionary linkage, and morphological tagging. We will review prior work in these areas. We will then briefly mention previous or current projects that have had or have similar goals to those of the larger scope of this project (outside this thesis), since we believe they will help motivate our approach.

2.1 Segmentation

Segmentation is generally not considered a task in and of itself. It is mostly used only as a preliminary approach, or as an approach coupled with other tasks, where the segmentation is only part of the output. Stemming is a related task that is almost identical to segmentation. The difference is that segmentation returns the segmented text, while stemming removes any prefixes and suffixes and returns only the stem.

Lee et al. [2003] is the most relevant work for segmentation, since they segment Arabic, a language related to Syriac, with a data-driven approach. Because of the linguistic similarities between Arabic and Syriac, there are only subtle differences in segmenting them. Lee et al. use manually segmented data with an unsupervised algorithm which learns the segmentation for Arabic without any additional language resources. At the heart of the algorithm is a word-level trigram language model, which they use to learn the correct weights for prefixes and suffixes. They report an accuracy of 97%.

Rogati et al. [2003] use an unsupervised learning approach to stemming. They claim their approach is low-resource, as its only language tools are an English stemmer, a small parallel corpus (10,000 words), and where available, simple language rules. Their approach agreed with a state-of-the-art Arabic segmenter (their method of validation) on 87.5% of the words.

Chinese and Japanese texts are not delimited by whitespace, and segmentation lends itself nicely to finding word boundaries for these languages. Like Semitic languages, these languages require segmentation as an early step for further computational linguistic analysis. Peng et al. [2004] segment Chinese text into word boundaries using a linear-chain conditional random field (CRF). They add 24 word- and character-based lists to aid segmentation, as well as introduce a model for dealing with unseen words. This additional language knowledge in conjunction with the new model allows them to achieve state-of-the-art results for Chinese segmentation.

There are many other approaches that combine segmentation with other tasks (see Bar-haim et al. [2008], Diab et al. [2004], Habash and Rambow [2005], Kudo et al. [2004], Petkevič [2001], Smith et al. [2005]). These typically utilize a morphological analyzer. A morphological analyzer is a linguistic tool that, given a word, produces all possible segmentations that that word could have according to the grammar of the language. In addition, most morphological analyzers also give corresponding baseforms and morphological information. These approaches that utilize a morphological analyzer are different from under-resourced approaches in that they choose among a limited number of possible segmentations given by a language tool instead of considering all possible segmentations that were previously seen during training. If the correct output is not enumerated by the morphological analyzer, it cannot be predicted.

Of these approaches, only Diab et al. [2004] and Habash and Rambow [2005] were applied to a Semitic language. As of June 2010, Habash et al. currently have the state-of-the-art segmentation for Arabic, achieving an accuracy of 99.7%. Lee et al. have the best approach not requiring additional language tools, with an accuracy of 97%.

2.2 Dictionary Linkage

Most previous work in dictionary linkage is found under the term lemmatization. Baseforms are often called lexemes or lemmas, and in the literature all three are used interchangeably. Lemmatization is therefore strictly baseform dictionary linkage. Little work has been done in classifying the root, possibly because the notion of root is used mainly in Semitic languages, and comparatively little work has been done in Semitic languages.

Chrupała et al. [2008] is the only previous low-resource approach for dictionary linkage. They have developed a system called Morfette that couples morphological tagging and lemmatization. This system builds off of Chrupała's [2006] data-driven lemmatization approach and combines lemmatization and morphological analysis for Spanish, Polish, and Romanian. These languages have distinct differences from Syriac, such as the fact that no segmentation is required, and they are read left to right. Chrupała's lemmatization is a two-pass method that first finds lemma-classes, and then assigns each lemma-class to a word the way a typical sequence labeler would. A lemma-class is formed by looking at the minimum edit distance between the baseform (lemma) and the word token. Since Spanish, Polish, and Romanian do not require segmentation, their lemmatization task maps from word token to baseform instead of stem (word token stripped of prefixes and suffix) to baseform, as baseform linkage does in Syriac. The transformation from word token to lemma is then used as the class the model tries to predict. For example, *pedir* is the baseform of the Spanish word *pidieron*. Instead of a model predicting *pedir* given *pidieron*, it would predict the lemma class, which is the transformation from *pidieron* to *pedir*. Chrupała has encoded that transformation for this example as {(D, i, 2), (I, e, 3), (D, e, 5), (D, o, 7), (D, n, 8)} where (D, i, 2) means delete *i* at position 2, and (I, e, 3) means insert *e* at position 3. All transformations are encoded using insertions and deletions and are created by first reversing both baseform and word.

Since Chrupała et al. did not have text labeled with baseforms, they used the minimum edit distance technique to acquire the baseforms. They then used a maximum entropy classifier

to predict these labels. Their lemmatization accuracies range from 93.0% – 98.5% depending on language and dataset.

Al-Shammari and Lin [2008] produced the first Arabic lemmatizer in a resource-rich setting by using language rules in a series of eight steps: (1) simple noun identification, (2) suffix and prefix removal, (3) noun dictionary generation, (4) verb identification, (5) verb dictionary generation, (6) finding all noun tokens, (7) stop word removal, and (8) root extraction for verbs. They evaluate by comparing the quality of clustering using words stemmed with a previously existing Arabic stemmer with the quality of clustering using words lemmatized by their method. Since a goal of lemmatization is to get the baseform of the word, and not just the stem, the hope is that lemmatization does a better job of finding the underlying base than stemming. Their hope was that this, in turn, would produce purer clusters and a higher cluster quality. They report a cluster quality of 70.8% using lemmatized documents and a cluster quality of 58% using stemmed documents. Unfortunately, no mention of lemmatization accuracy is presented.

Daya et al. [2008] use the notion of roots and patterns in their framework to find roots for Semitic texts. In this framework, words are created by combining a pattern and a root, or inserting the root letters into a pattern. Since roots generally have three consonants, a pattern will often have three spaces where the consonants of the root fit. The downside to this approach is that it works only for words conforming to the patterns available. While their approach does not utilize a morphological analyzer, it does require a list of roots, lists of common prefixes and suffixes, and "knowledge of word-formation processes, and behavior of the weak roots in certain paradigms," placing it in the resource-rich category. On their best models, they achieve an F-score (another measure of accuracy) ranging between 86.92% and 91.64%.

2.3 Morphological Disambiguation

Almost all morphological disambiguation approaches require the use of a morphological analyzer, making them resource-rich approaches. Habash and Rambow [2005] even claim that Arabic morphological disambiguation "cannot be done successfully using methods for English." In addition,

Wintner [2004] noted that morphological analyzers were "possible — indeed, necessary" to further morphological disambiguation of Hebrew. Where morphological analyzers are available, the benefits are large. In addition to providing linguistically possible morphological information for each word, morphological analyzers also give segmentation and baseform information. Unfortunately, these tools are time consuming to build and require advanced linguistic knowledge of the language. For Syriac, a morphological analyzer is not available. Kiraz [2000] created a Syriac morphological analyzer; however, this morphological analyzer was developed on outdated equipment and is no longer working or available to us.

Hajič and Hladká [1998] were among the first to do morphological disambiguation. They worked with Czech and treated the problem as a sequence-labeling problem, given the many similarities between morphological tagging and part-of-speech (POS) tagging. With the aid of a morphological analyzer to reduce the number of possible output classes for a given word, they introduced the notion of ambiguity classes. An ambiguity class is a set of possible ambiguous classes a classifier needs to resolve. For example, the word *jen* has three possible morphological taggings, or set of tags: `TT----------`, `NNIS1-----A--`, and `NNIS4-----A--`. Here, each character represents a value for a morphological attribute (for Czech there are 13 morphological attributes) determined by the position in the string. The dash means that attribute is not applicable. For this word, there are six ambiguity classes (`NT`, `NT`, `-I`, `-S`, `-14`, `-A`). Ambiguity classes are determined by the possible unique values for each morphological attribute. For example, the first ambiguity class (`NT`) is determined by the number of unique values for the first morphological attribute in the three morphological taggings shown above. The second ambiguity class is the same as the first for this word. The third has unique values `-` and `I`, creating the third ambiguity class of (`-I`). Where there is no ambiguity (e.g., the sixth morphological attribute), an ambiguity class is not created. Hajič and Hladká constructed a separate classifier for each ambiguity class that needed to be learned, for a total of 378 models. They combined the output of these models in a way that uses only valid combinations and achieves an accuracy of 92.0% compared to the baseline of 77.8%.

Habash and Rambow [2005] currently have the best-known results for morphological disambiguation on Arabic. Instead of using ambiguity classes, Habash and Rambow took a simpler approach — to use a separate classifier for each morphological attribute. The difficulty is in combining the outputs from the classifiers to make the best possible joint tag. With the aid of a morphological tagger, they showed different methods for combination that most closely matched a possible tagging given by the morphological analyzer. This approach proved quite successful, as they achieved an overall tag accuracy of 97.6%.

Hlaváčová [2001] looks at solving the problem of unknown words (specifically those for which a morphological analyzer does not provide any results) for Czech. He improved the number of words to which the morphological analyzer could offer analyses by 60% – 70% depending on the corpus.

Feldman and Hana [2010] provide an overview of current resource-poor NLP-related work. They also show work on Russian, Czech, and Romance languages. They approach morphological tagging with a morphological analyzer of Czech to tag Russian, to approach the problem from a cross-language point of view. To further augment the system, they provide a list of cognates and alter the Czech training data to look more like Russian. With these improvements, they report accuracies of up to 80%.

Others (Bar-haim et al. [2008], Kudo et al. [2004], Mansour et al. [2007], Petkevič [2001], and Smith et al. [2005]) have used similar approaches that also utilize morphological analyzers to reduce the number of output classes. These approaches are done on a variety of languages, with differing techniques, but still, all use morphological analyzers.

There is little previous work regarding morphological disambiguation in an under-resourced setting. To our knowledge, only Chrupała et al. [2008] and Kübler [2010a, 2010b] do not use a morphological analyzer or any other language resources. Chrupała et al. used maximum entropy classifiers to predict baseforms and morphological attributes. Depending on the language and dataset, their morphological disambiguation accuracies range from 84.9% – 98.8%, with Span-

ish and Romanian both achieving accuracies over 95%. Polish accuracies are not as strong, since Polish is a Slavic language with more morphological complexity than Spanish or Romanian.

Mohamed and Kübler 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 low when first segmenting word tokens. They use TiMBL, a memory-based learner, as their model and report an accuracy of 94.74%.

2.4 Similar Projects

As mentioned above, this thesis is a small part of a larger project being undertaken by the Neal A. Maxwell Institute in which they will gather texts and annotate them via active learning. Many other organizations are interested in developing labeled corpora, and they use varied amounts of computational assistance in their tasks.

The Perseus Project began in 1998 and has continued since then gathering and annotating Greek and Latin texts. The project uses machine learning to a small degree, but does not use it for annotation prediction. It utilizes machine learning models for data and text mining. The Thesaurus Linguae Graecae (TLG) is a similar project, with the aim of creating a digital Greek library. The TLG, begun in 1972, created a Greek morphological analyzer to do its annotation.

The Maxwell Institute previously created a Dead Sea Scrolls Electronic Corpus, but this project also did not leverage machine learning. Their annotations were collected by scholars over a number of years. Turgama is yet another project that uses manual annotations. This project focuses on Syriac texts and does use computers to aid annotation (in the form of linguistic rules), but it uses additional Syriac data, such as a lexicon and descriptions of morphology. Also, the scope of Turgama is rather limited compared to that of the comprehensive Syriac concordance project.

The Comprehensive Syriac Corpus project plans to use active learning and pre-annotation to aid human annotation. For this cause, we are interested in exploring the performance of models

with varying amounts of training data. Additionally, the project hopes to utilize computational linguistic methods as much as possible.

Chapter 3

A Probabilistic Morphological Analyzer for Syriac

Paper Submitted to EMNLP 2010

3.1 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 and connect novel probabilistic models for segmentation, dictionary linkage, and morphological tagging 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.

3.1.1 Syriac Background

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 Semitic language of the Christian Near East and a dialect of Aramaic. It is currently almost entirely 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 annotating these texts with useful information, we will facilitate systematic study by scholars of Syriac, the Near East, and Eastern Christianity.

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 British and Society [1920] and a concordance thereof. We aim to replicate this on a much larger scale with more modern tools, building up from the labeled New Testament data, our only resource. As such, our learning and annotation framework requires only labeled data. Additionally, the approach described in this paper may be useful for the analysis of other morphologically rich languages.

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. Since Syriac is an abjad, its writing system does not require vowels. As a dialect of Aramaic, it is written right-to-left and has a templatic morphology, based on a system of triliteral roots, with prefixes, suffixes, infixes, and enclitic particles.

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 aid in the definitions of segmentation, dictionary linkage, and morphological tagging:

- word token - contiguous characters delimited by whitespace
- stem - the main part of a word token to which prefixes and suffixes can be attached

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

- baseform - the form from which the stem is derived; 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 *LMLCCON*, which means "to your (masculine plural) king". For this word, the stem is *MLC*; the baseform is *MLCA* "king"; and the root is *MLC*. The stem is an inflected baseform and does not necessarily form a word on its own. In Syriac, the same root *MLC* is the foundation for other words such as promise, counsel, deliberate, reign, queen, kingdom, and realm.

3.1.2 Sub-tasks

Segmentation, or tokenization as it is sometimes called (e.g., Habash and Rambow [2007]), is the process of dividing a word into its prefixes, a stem, and suffixes. 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. While segmenting Syriac, we can treat all prefixes as a single unit. Suffixes may be multiple characters in length and encode the morphological attributes of suffix gender, suffix person, suffix number, and suffix contraction. The attribute of suffix contraction encodes whether the suffix is normal or contracted. For the example word *LMLCCON*, the prefix is *L* "to", the stem is *MLC* "king", and the suffix is *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 has exactly one baseform from which it is derived, and each baseform has exactly one root from which it is derived. As such, linkage may be thought of as two separate processes: (1) baseform linkage, where the stem is mapped to a baseform; and (2) root linkage, where the baseform is mapped to a root. For our example *LMLCCON*, baseform linkage would map stem *MLC* to baseform *MLCA*, and root linkage would map baseform *MLCA* to root *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

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 3.1: The values for the morphological attributes of the stem *MLC*, "king".

Attribute	Value
Gender	masculine
Person	second
Number	plural
Contraction	normal suffix

Table 3.2: The values for the morphological attributes of the suffix *CON*, "(masculine plural) your".

stem and (2) tagging the suffix. For Syriac, scholars have defined for this task a set of morphological attributes consisting of 12 attributes for the stem and 4 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 4 morphological attributes for the suffix are suffix contraction, suffix person, suffix gender, and suffix number. 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 with a value (i.e., "tag" or "label") for each of the 12 stem attributes, and the suffix is labeled with a value for each of the 4 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 3.1 and 3.2 show the correct stem and suffix tags for the word *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 contains previous work relevant to our approach. Finally, in Section 7 we briefly conclude and offer directions for future work.

3.2 The Syromorph Approach

Since we have no language tools, but we do have labeled data, we focus on automatically annotating Syriac text in a data-driven fashion. We desire the sub-models to influence each other, since segmentation, linkage, and morphological tagging are not mutually independent tasks. 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.

3.2.1 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 3.1 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 — specifically the stem.

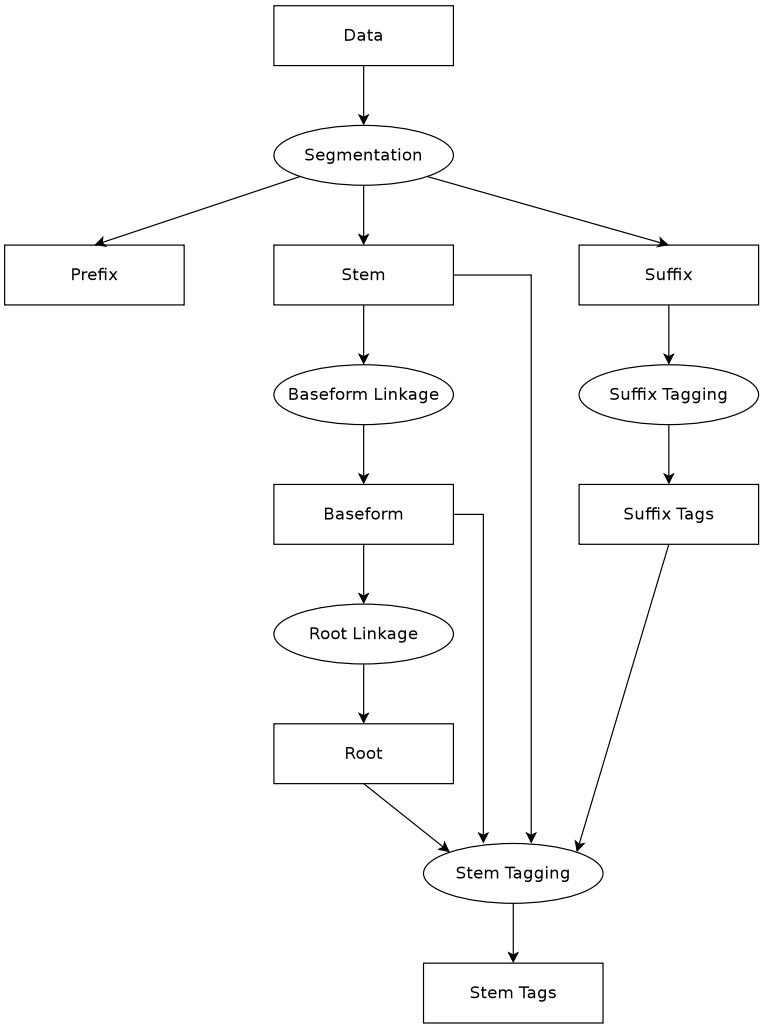

Figure 3.1: The **syromorph** model. Each rectangle is an input or output and each oval is a process employing a sub-model.

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 upstream sub-models. The upstream sub-models of a given sub-model are those sub-models upon which the given sub-model depends. We created the dependencies shown in Figure 3.1 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. When the stem tagger is being trained, for example, the local context consists of the predicted segmentation, predicted baseforms, predicted roots, and predicted suffix tags for each word in the sentence. In addition, the local context for stem tagging contains the words of the sentence and the stem labels for the previous o stems. Since features are extracted from the local context: for stem tagging, for example, we extract features such as current stem, previous stem, current baseform, previous baseform, current root, previous root, current suffix tags, and previous suffix tags. "Previous" in this case refers to labels on the immediately preceding word token.

3.2.2 Segmentation

The **syromorph** segmentation model is a hybrid word- and consonant-level model adapted from a model used for data-driven diacritization Haertel et al. [2010]. Like Haertel et al., we employ maximum entropy Markov models for each of our probabilistic sequence models. Their models distinguish between words that are rare and words that are not rare, and they showed that the distribution over labels is different for known words and unknown words. In this work, we consider only unknown words to be rare and words seen during training to be non-rare. For each of the non-rare word types, a separate model is trained with the purpose 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].

For those words that are considered rare, since there is no word type–specific model, we back off to a consonant-level model. Our consonant-level segmentation model uses the notion of BI (Beginning and Inside) tags, which have proved 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 a maximum entropy Markov model to predict one of these six tags for each consonant. Furthermore, we constrain the model to allow only legal possible transitions given the current prediction, so that prefixes must come before stems and stems before suffixes. We then transform the guessed consonant-level hypotheses to a word-level tag. 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.

3.2.3 Dictionary Linkage

For dictionary linkage, we divide the problem into two separate tasks: baseform linkage and root linkage. For each of these tasks, we explore two different hybrid models. These hybrid models are similar to **hybrid** for segmentation, in that they use a similar type of model for non-rare (known) words, namely a collection of separate models for each word type (either a stem or baseform, depending on the linker). The approach for non-rare words is the same for both hybrid models.

For rare (unknown) words, there are two separate approaches to linkage. The first rare approach 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 formed 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. For example, the transformation of *XE;N* to *XEA* would be

{(D, *;*, 2), (I, *A*, 2), (D, *N*, 3)}. This transformation means delete *;* and *N* from positions 2 and 3, respectively, and insert *A* into position 2. Applying this transformation to *XE;N* results in *XEA*.

We transform all baseforms into their corresponding lemma-classes in baseform linkage, and we do the same for the roots in root linkage. The predicted transformation is then applied to the stem or baseform in order to construct the actual prediction (baseform or root, respectively). The advantage here is that common transformations are grouped into a single class, thereby allowing the model to adequately predict baseforms and roots that have not been seen during training, but whose transformations have been seen. We call this hybrid model **hybrid-morfette**. This model is trained on all words in order to capture as many transformations as possible.

The second approach for rare (unknown) words uses a maximum entropy Markov model trained on all words seen in training. One disadvantage of this model is that it predicts only baseforms and roots that were seen in training data. This approach is just like the unknown-word approach used by Toutanova and Manning [2000] for POS tagging. This approach is called **hybrid-maxent**.

3.2.4 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, where the label is the concatenation of all the morphological attribute values. In Table 3.3, the stem tag and suffix tag columns contain the monolithic tags for stem tagging and suffix tagging. We use a maximum entropy Markov model to predict a monolithic tag for each stem or suffix and call this model **maxent-mono**.

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 maximum entropy Markov models for each of the single-attribute taggers. This model is called **maxent-ind**.

3.2.5 Decoding

Whereas Viterbi decoding produces the most probable sequence of labels given a sequence of unlabeled words, our per-task decoders are beam decoders, with beam-size equal to b. In particular, we limit the number of per-stage back-pointers to b due to the large size of the tagset for many of our sub-models.

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.

3.3 Experimental Setup

We are using the Syriac Peshitta New Testament in the form compiled by Kiraz [1994]. The Way International, a Biblical research ministry, annotated this version of the New Testament by hand and required 15 years to do so. 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.

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

Table 3.3: A labeled Syriac sentence *OEBDT ANON LALHN MLCOTA OCHNA OMLCA*, "And you have made them a kingdom and kings and priests for our God." (Revelation 5:10)

Table 3.3 shows part of a tagged Syriac sentence using this tagset. The suffix and stem tag values are indices representing morphological attributes. In the example sentence the suffix tag `1011` represents the values "masculine", "N/A", "plural", "normal suffix" for the attributes suffix gender, suffix person, suffix number, and suffix 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 3.1 from top to bottom. The following describes the interpretation of the stem values represented in Table 3.3. Grammatical category values `0`, `2`, and `3` represent "verb", "noun", and "pronoun", respectively. 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.

This dataset consists of 109,640 word tokens. Since we are focusing on under-resourced circumstances, we sweep our results for varying amounts of data to better understand how our models perform in such circumstances. For each point on a learning curve, we take a percentage of the data as training, and test on a fixed development test set.

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.

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 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 sub-tasks.

3.4 Experimental Results

We evaluate our method by calculating the average accuracy of ten-fold cross validation. 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.

3.4.1 Baseline Results

Each of the baselines performs surprisingly well. Table 3.4 is grouped by sub-task and reports the results of each of the baseline sub-tasks in the first row of each group. The accuracies of most of

	Model	Total	Known	Unk
SEG	baseline	96.75	99.64	69.11
	hybrid	**98.87**	**99.70**	**90.83**
BF	baseline	95.64	98.45	22.28
	hybrid-morfette	**96.19**	98.05	**78.4**
	hybrid-maxent	**96.19**	99.15	67.86
RT	baseline	98.84	**99.56**	80.20
	hybrid-morfette	**99.05**	99.44	**88.86**
	hybrid-maxent	98.34	**99.45**	69.30
SUF	monolithic baseline	98.75	98.75	N/A
	independent baseline	96.74	98.78	**0.01**
	maxent-mono	**98.90**	**98.90**	N/A
	maxent-ind	**98.90**	**98.90**	N/A
STEM	monolithic baseline	83.08	86.26	0.01
	independent baseline	53.24	86.90	0.00
	maxent-mono	**89.48**	**92.87**	**57.04**
	maxent-ind	88.43	90.26	40.59

Table 3.4: Results of the individual sub-models used in our approach.

the tasks are so high because the ambiguity of the labels given the instance is quite low. Ambiguity rates for segmentation, baseform and root linkage, and stem and suffix tagging are 1.01, 1.05, 1.02, 1.47, and 1.35, respectively. Preliminary experiments indicated that if we had not separated these tasks, the tagging accuracy would have been lower. The unknown tagging accuracy for the monolithic suffix tagger is not applicable, because there were no suffixes that were not seen during training. This makes sense, considering that the baseline segmenter chooses only suffixes that were seen during training.

3.4.2 Individual Models

Table 3.4 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, 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

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

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 word that was never seen was predicted, then the weights 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.

3.4.3 Joint Model

Table 3.5 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 **hybrid** / **hybrid-maxent** / **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, yielding a 35.11% reduction in error rate; 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 3.2, 3.3, and 3.4 show learning curves for total, known, and unknown accuracies for the joint pipeline model. As can be seen in Figure 3.2, 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.

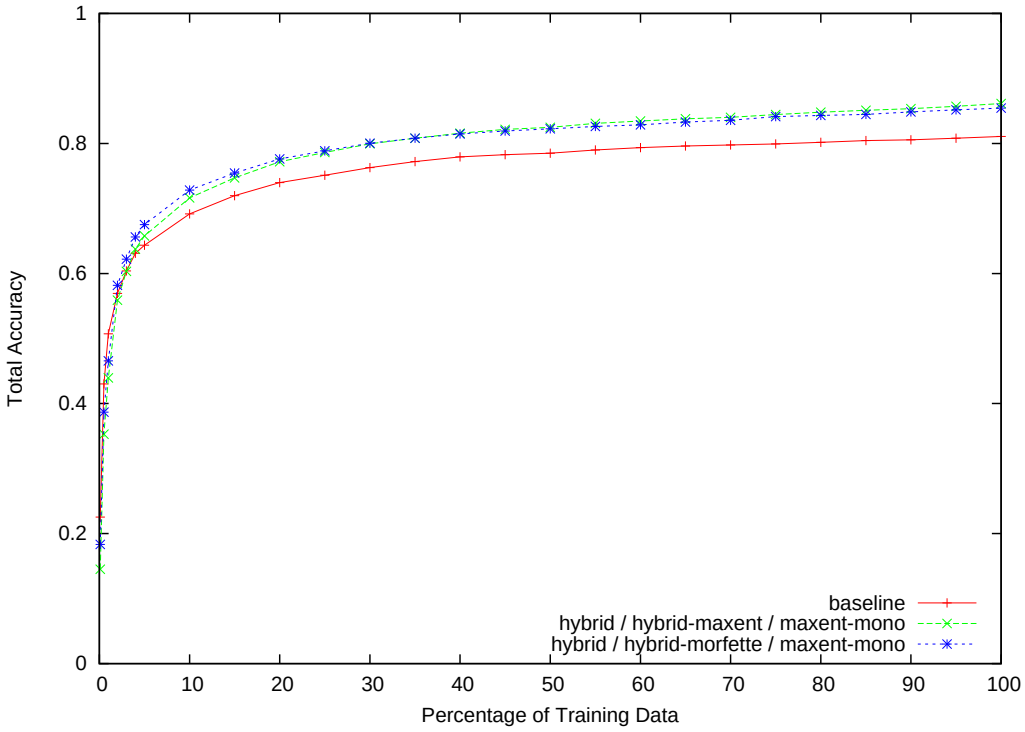

Figure 3.2: Learning curve for the total accuracy of the joint model.

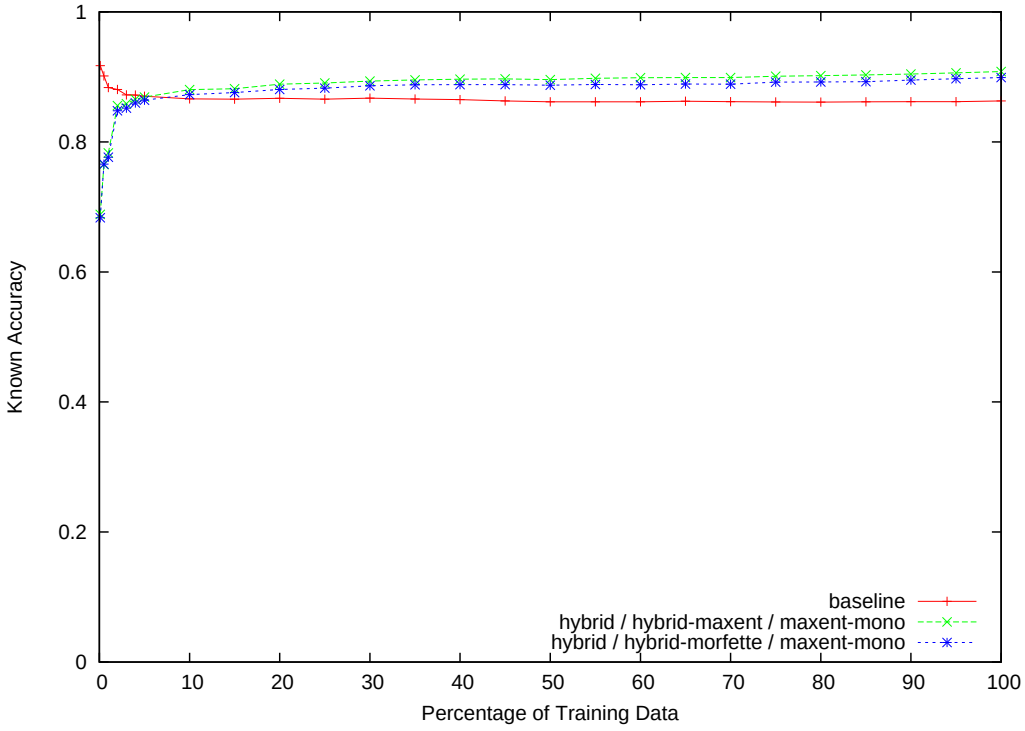

Figure 3.3: Learning curve for the known accuracy of the joint model.

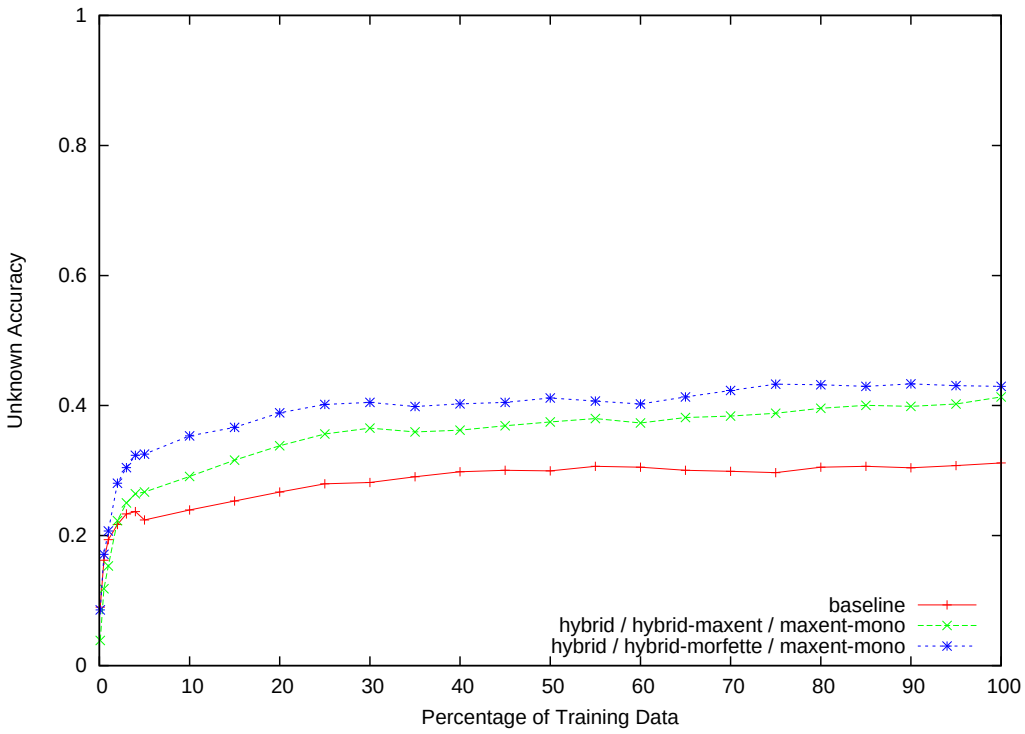

Figure 3.4: Learning curve for the unknown accuracy of the joint model.

Figure 3.3 shows the baseline performing quite well 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 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 3.4 clearly shows that **hybrid-morfette** linkers outperform **hybrid-maxent**; however, Figure 3.2 shows that between 25% and 40% of the data, the **hybrid-morfette**'s advantage on unknown words (and disadvantage on known words) allows **hybrid-maxent** to surpass it.

3.5 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 maximum entropy Markov models 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.

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 resources. 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), but get segmentation for free 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 (10). 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] used 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, this morphological analyzer was developed on outdated equipment and is no longer working or available to us.

3.6 Conclusions and Future Work

We have shown that we can effectively model segmentation, linkage to headwords in a dictionary, and morphological tagging with a joint model called **syromorph**. We have introduced novel approaches for segmentation, dictionary linkage, and morphological tagging. 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 pre-annotation 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 entire tag accuracy closer to the accuracy of individual decisions. Feature engineering for the stem tagger and the exploration of possible new morphological tagging techniques we leave for future work.

Chapter 4

Additional Results

In this chapter, we expound on results highlighted in chapter 3 and discuss those which did not fit in chapter 3. We will discuss clustering for stem tagging, then further examine the individual models. Finally, we will describe statistical validation.

4.1 Clustering

One experiment not mentioned in Chapter 3 is the clustering of morphological attributes for the stem tagger. With 12 attributes (or tags) to predict per stem, assuming total independence or no independence (monolithic) among the attributes both have drawbacks. The independent approach sometimes forms combinations that are linguistically impossible, due to poor independence assumptions. The disadvantages of the monolithic approach are that it is memory-intensive to train (since there are over 2,500 tag types seen during training) and that it suffers from data sparsity.

To try to offset these two problems, we use hierarchical agglomerative clustering to group the morphological attributes into clusters, allowing a compromise between independence assumptions and data sparsity problems. We then train one model for each cluster, assuming independence between clusters, but allowing for dependence within clusters. Again, we combine the outputs of the models in a naïve way, choosing the best prediction from each model.

We examine single-link, complete-link, and average-link hierarchical agglomerative clustering techniques in which we use pairwise mutual information between the morphological attributes as the distance for clustering. For each of the clustering techniques, we create a dendrogram (see Figure 4.1 for the complete-link dendrogram). By drawing a horizontal cut through any

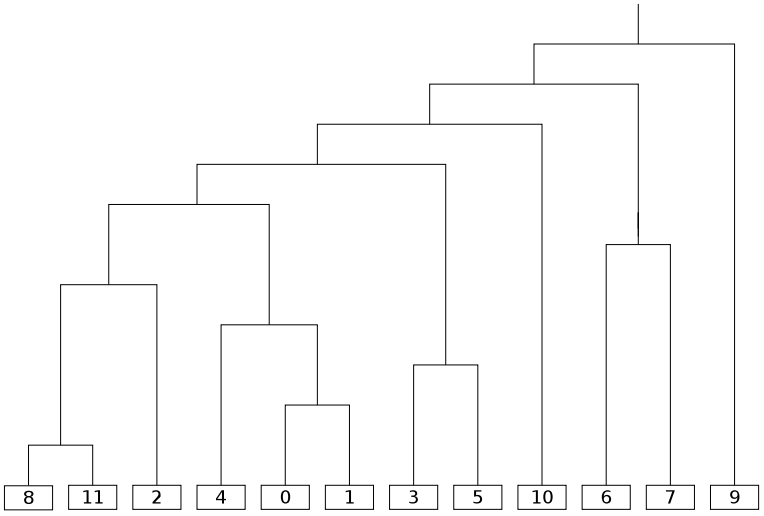

Figure 4.1: The dendrogram for complete link hierarchical agglomerative clustering of stem morphological attributes. The indices of the nodes represent the following morphological attributes in order, from 0 to 11: verb conjugation, aspect, state, number, person, gender, pronoun type, demonstrative category, noun type, numeral type, participle type, grammatical category.

part of the dendrogram, one can arrive at any number of clusterings (groups of clusters) ranging from one to the number of leaf nodes. Clusters consist of the groups of nodes that are connected in the graph below the horizontal threshold. Notice that at the extremes — a horizontal line just above the leaf nodes or through the trunk of the dendrogram — the clustering is equivalent to our independent or monolithic approaches, respectively. There exist many clusterings for which single-, complete-, and average-link yielded the same clusterings. These clusterings, are perhaps stronger, since the same clusterings were achieved by different techniques. We refer to these models by the first letter of their algorithm (A, C, or S) followed by the number of clusters. For example, S-4 would be single-link with four clusters and C-11 would be complete-link with eleven clusters. Table 4.1 shows the accuracies for each clustering according to these clustering algorithms.

Clustering the stem morphological attributes was not effective on the front end. As the table reveals, Each time independence is introduced, by breaking the monolithic tag into at least two clusters, the accuracy drops. Although the unknown accuracy is higher on some of the more independent approaches, the increase in unknown accuracy is not enough to increase the total

Cluster	Total Accuracy	Known Accuracy	Unknown Accuracy
ACS - 1	**89.48**	**92.87**	57.04
ACS - 2	89.41	92.83	56.60
AC - 3	89.46	**92.87**	56.85
S - 3	89.44	**92.87**	56.56
AC - 4	88.95	92.41	55.89
S - 4	88.92	92.35	56.08
AS - 5	88.95	92.41	55.89
C - 5	88.91	92.09	58.49
AS - 6	88.91	92.09	58.49
C - 6	88.73	91.96	57.82
AC - 7	88.73	91.96	57.82
S - 7	88.87	92.08	58.20
AC - 8	88.69	91.34	57.63
S - 8	88.64	91.92	57.24
AC - 9	88.61	91.88	57.33
S - 9	88.34	91.55	57.72
ACS - 10	88.33	91.54	57.62
ACS - 11	88.53	91.60	**59.17**
ACS - 12	88.43	91.54	58.59

Table 4.1: Results for different clusterings of the stem morphological attributes

accuracy. No clustering at all worked best; hence, that scenario receives primary focus in chapter 3, with the independent scenario for contrast.

4.2 Additional Sub-Model Results

Results not mentioned in chapter 3 include how well individual models performed using predicted values instead of truth as input. The following figures show the accuracies of each sub-model within the context of the **prob-morph** joint model. Figures 4.2, 4.3, 4.4, 4.5, and 4.6 show learning curves for the sub-models for total tag accuracy on the token level. Figures 4.7, 4.8, 4.9, 4.10, and 4.11 show similar results, but for known tag accuracy. Finally, Figures 4.12, 4.13, 4.14, 4.15, and 4.16 show learning curves for the sub-models for unknown tag accuracy. It is clear that for each sub-model, after the 10% of training data mark, the **prob-morph** models are superior to the naïve approach.

4.2.1 Total Accuracy

Figures 4.2, 4.3, 4.4, 4.5, and 4.6 show the learning curves for each of the sub-models for total tag accuracy. Figure 4.2 shows that the linkage models **hybrid-maxent** and **hybrid-morfette** have very little effect on the **hybrid** segmenter, which performs considerably better than the naïve segmenter baseline. As mentioned previously, Figure 4.3 indicates that **hybrid-morfette** does better than **hybrid-maxent** for lower amounts of training data (where there is a greater percentage of unknown words). The crossover point, where **hybrid-maxent** starts performing equally well, or better, is at about 60% of the data, where the percentage of unknown words is about 12.2%. The root accuracy learning curve (see Figure 4.4) corroborates the relationship between **hybrid-maxent** and **hybrid-morfette** discussed in chapter 3, and in this case is more pronounced, with **hybrid-morfette** dominating the entire learning curve. Figure 4.5 shows that the baseline simply does not perform as well as either of the joint models, which for this sub-model are indistinguishable. Figure 4.6 looks most similar to the total joint accuracy, since stem tagging is the sub-model

that performs the worst of the sub-models and pulls the total accuracy down the most. The picture here is clear: **hybrid-maxent** is superior to **hybrid-morfette**, and both outperform the baseline.

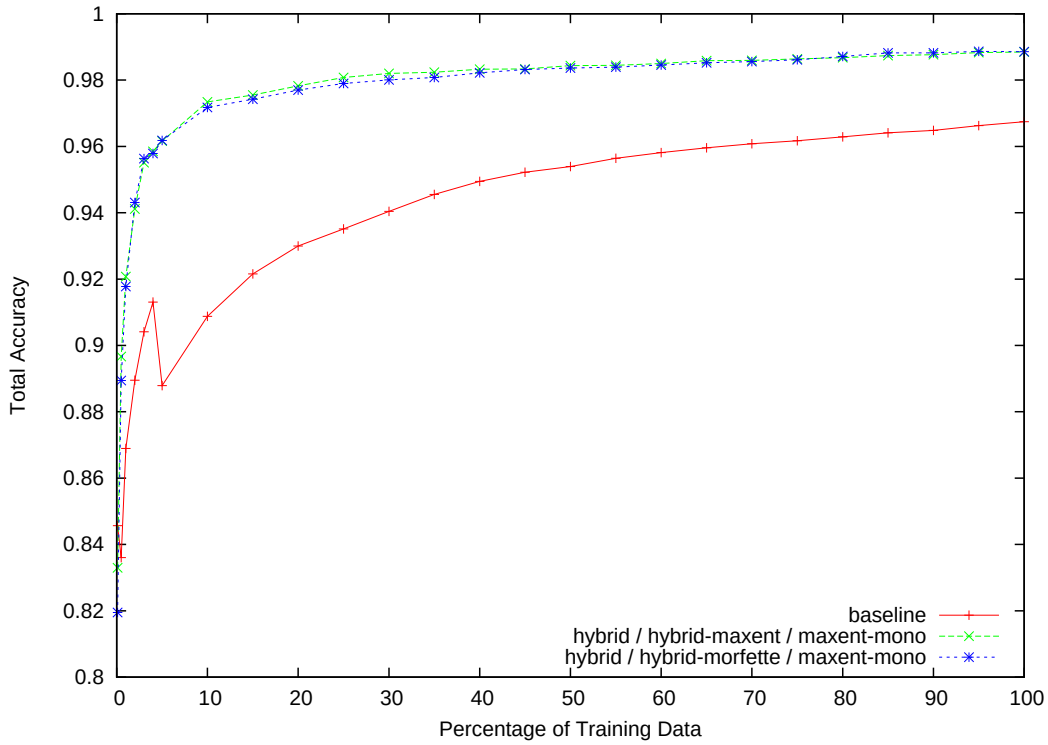

Figure 4.2: The learning curve for the total accuracy of segmentation models in the joint pipeline.

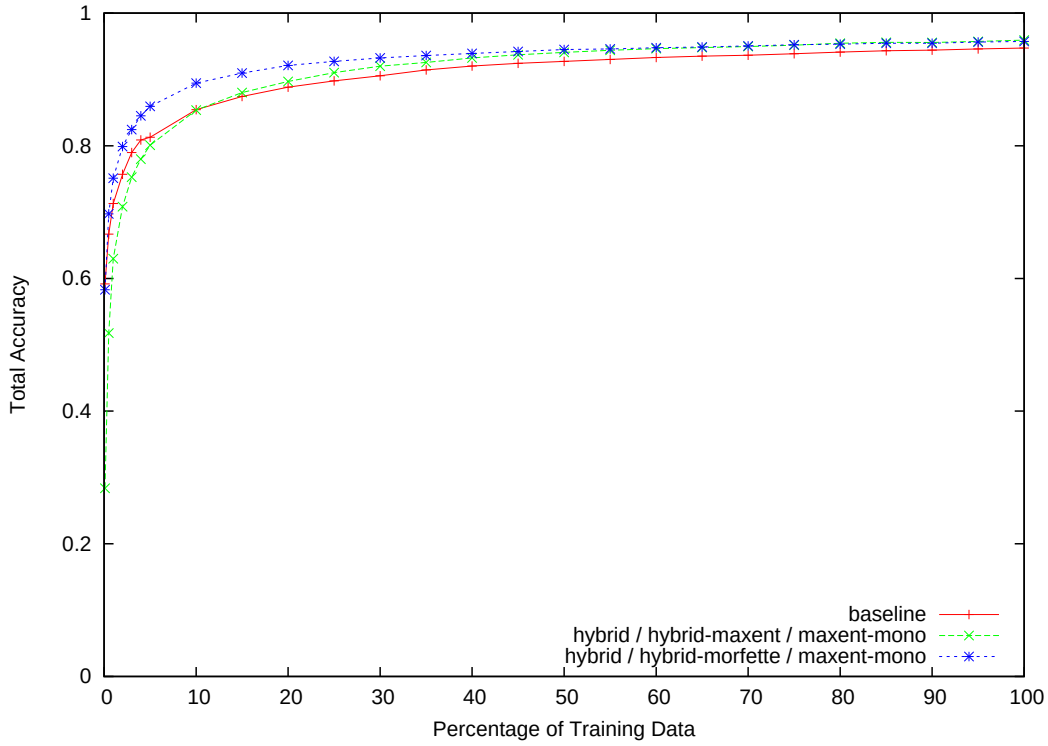

Figure 4.3: The learning curve for the total accuracy of baseform linker in the joint pipeline.

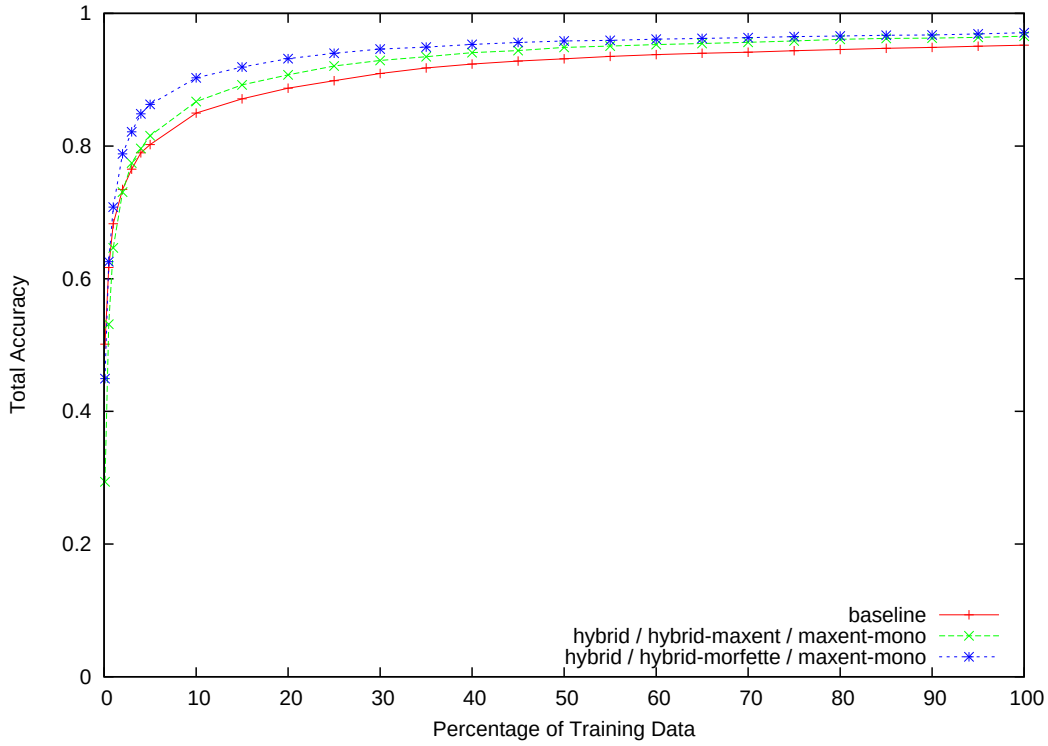

Figure 4.4: The learning curve for the total accuracy of root linker in the joint pipeline.

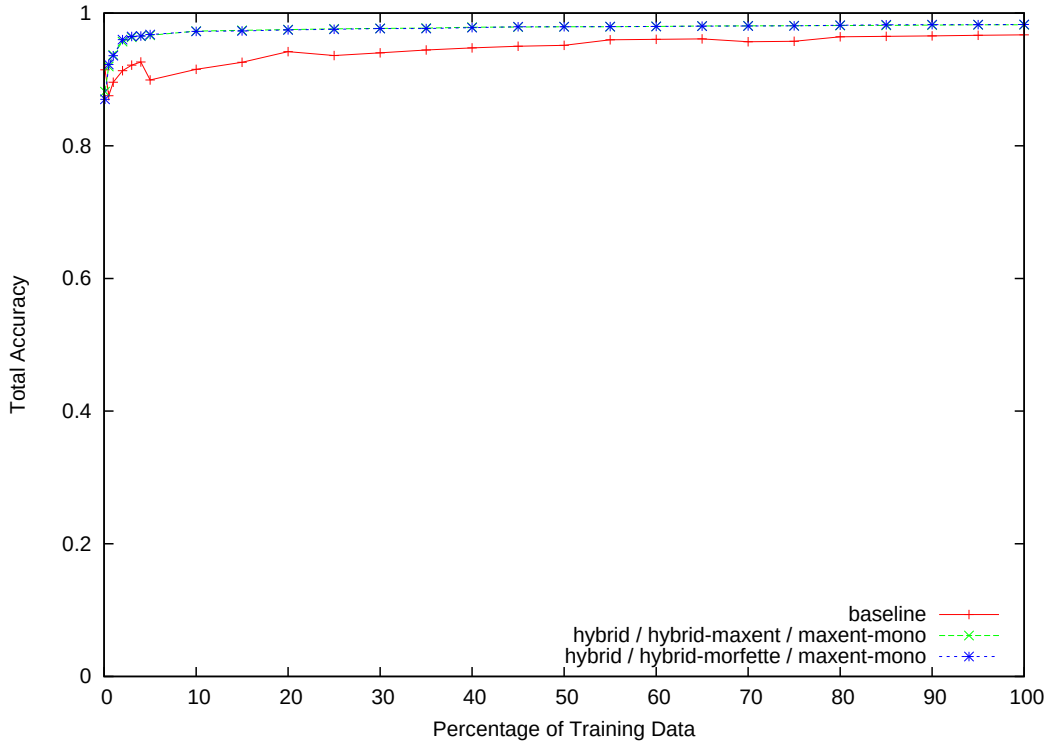

Figure 4.5: The learning curve for the total accuracy of suffix tagger in the joint pipeline

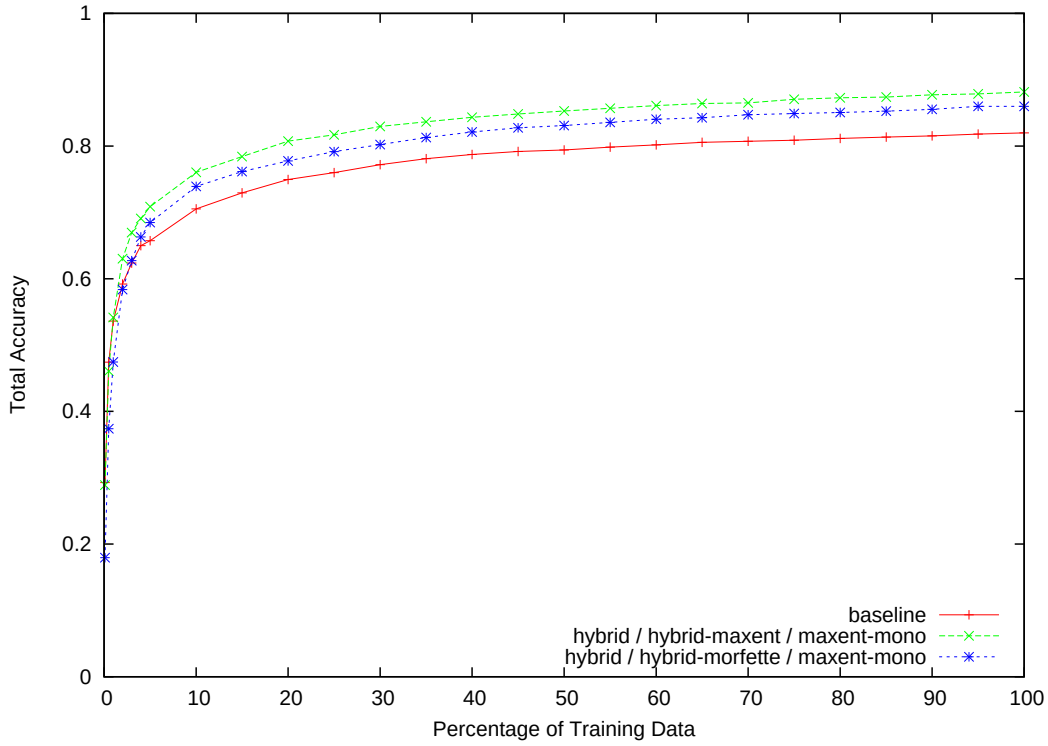

Figure 4.6: The learning curve for the total accuracy of stem tagger in the joint pipeline.

4.2.2 Known Accuracy

Figures 4.7, 4.8, 4.9, 4.10, and 4.11 show the learning curves for each of the sub-models for known tag accuracy. The learning curve for segmentation known accuracy (Figure 4.7) looks noisy, but the scale on the y-axis is very small. It seems as though the **prob-morph** approaches edge out the baseline, but the difference may not be significant. Figure 4.8 again shows how **hybrid-morfette** does well on unknown words, but suffers with known accuracy. The root linkage does not appear to have this problem, as both models consistently outperform the baseline (see Figure 4.9). Figure 4.10 shows the superiority of our approaches over the baseline for suffix tagging, while Figure 4.11 shows the interesting behavior explained in chapter 3 for the total accuracy of the joint model.

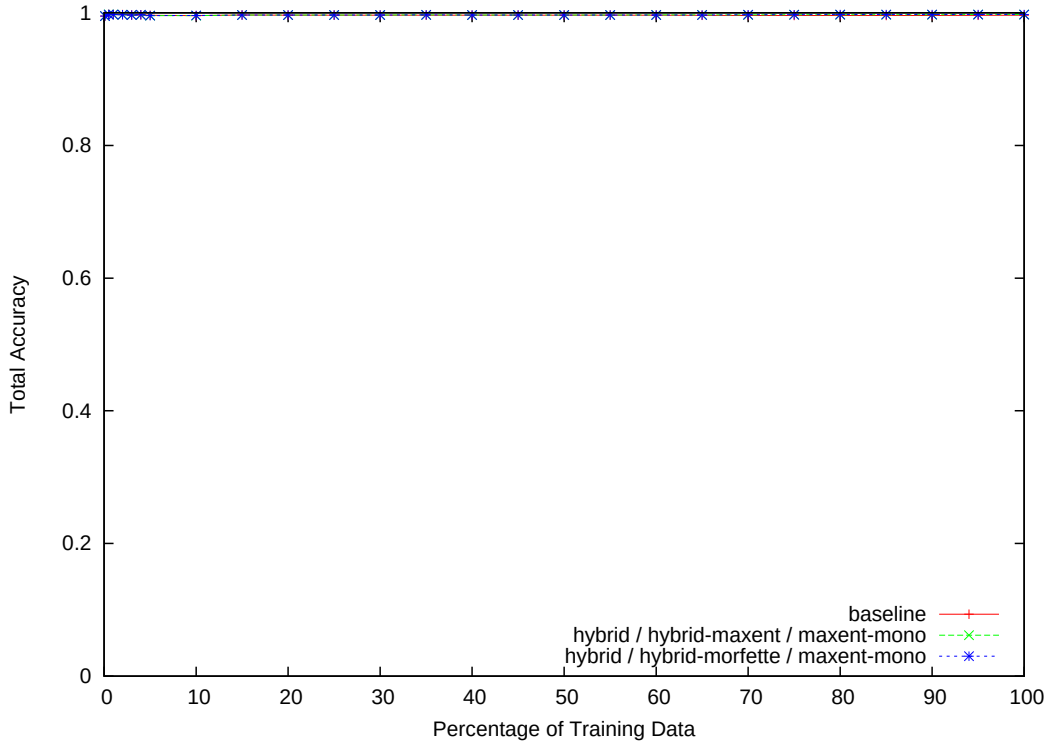

Figure 4.7: The learning curve for the known accuracy of segmentation models in the joint pipeline.

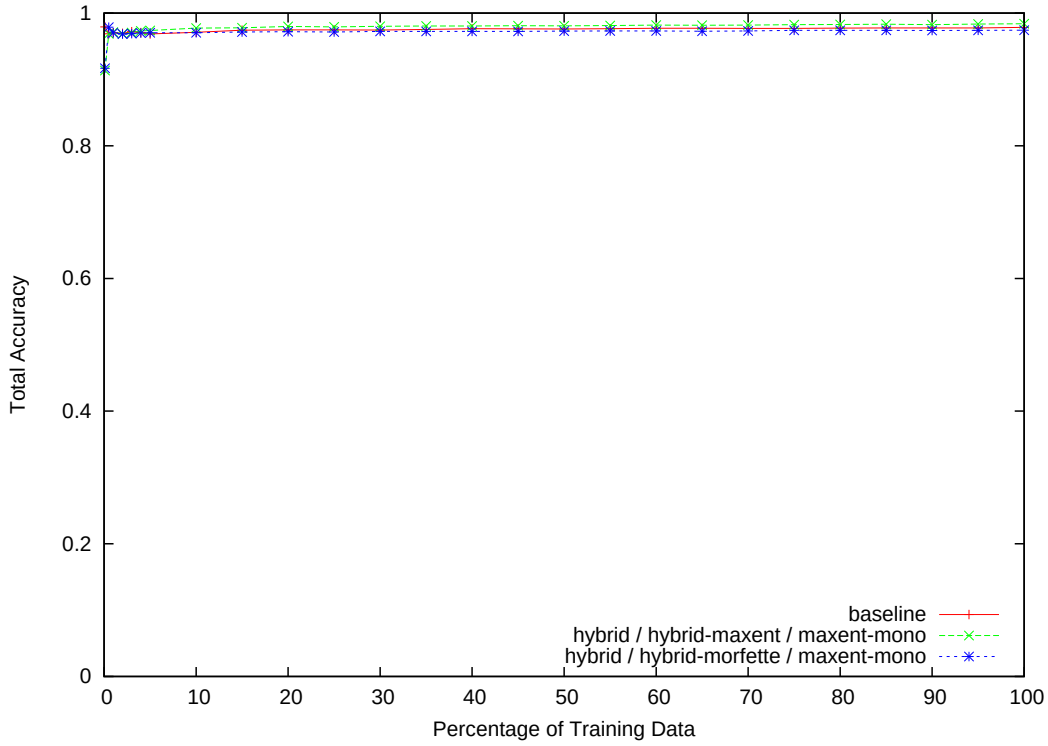

Figure 4.8: The learning curve for the known accuracy of baseform linker in the joint pipeline.

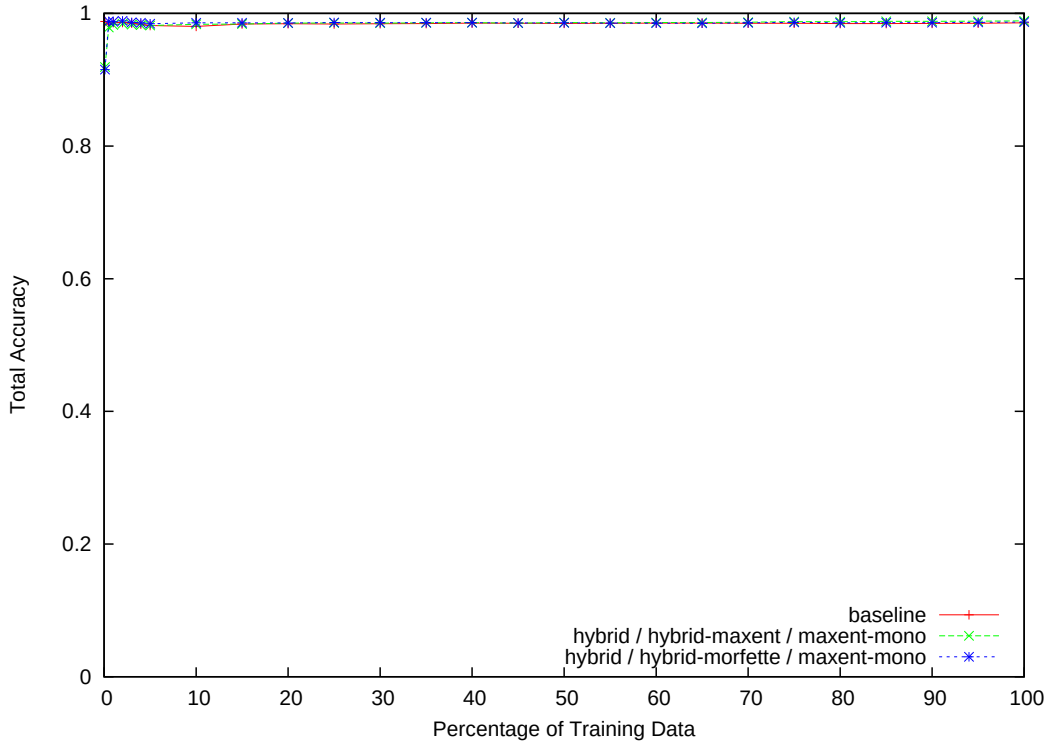

Figure 4.9: The learning curve for the known accuracy of root linker in the joint pipeline.

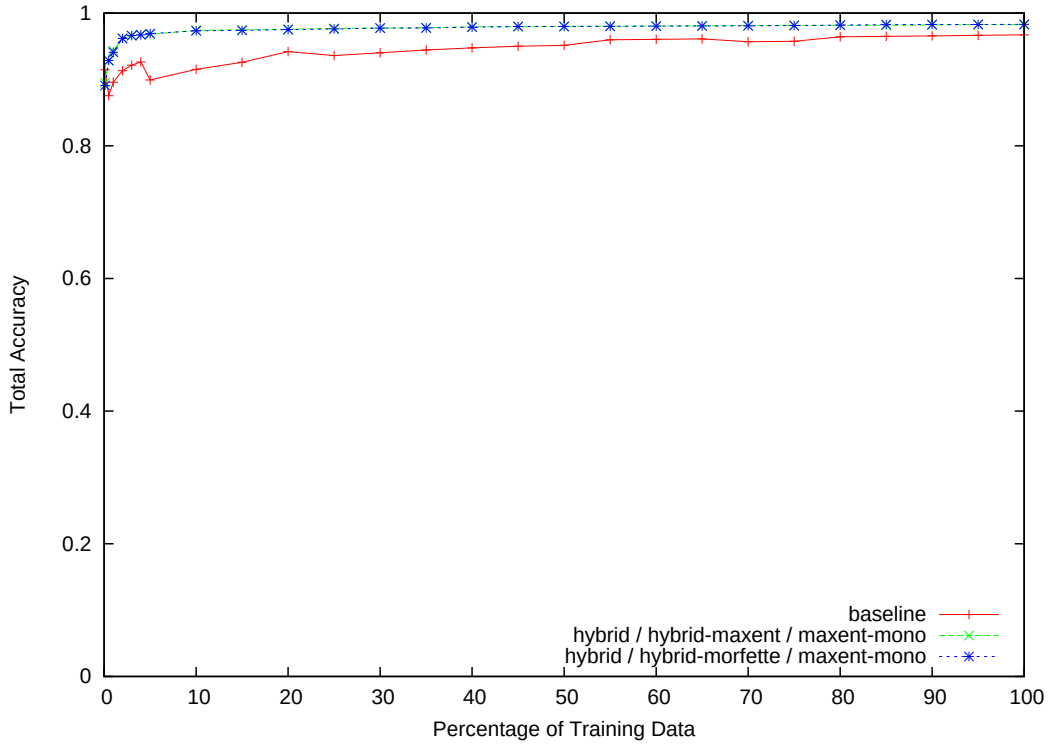

Figure 4.10: The learning curve for the known accuracy of suffix tagger in the joint pipeline

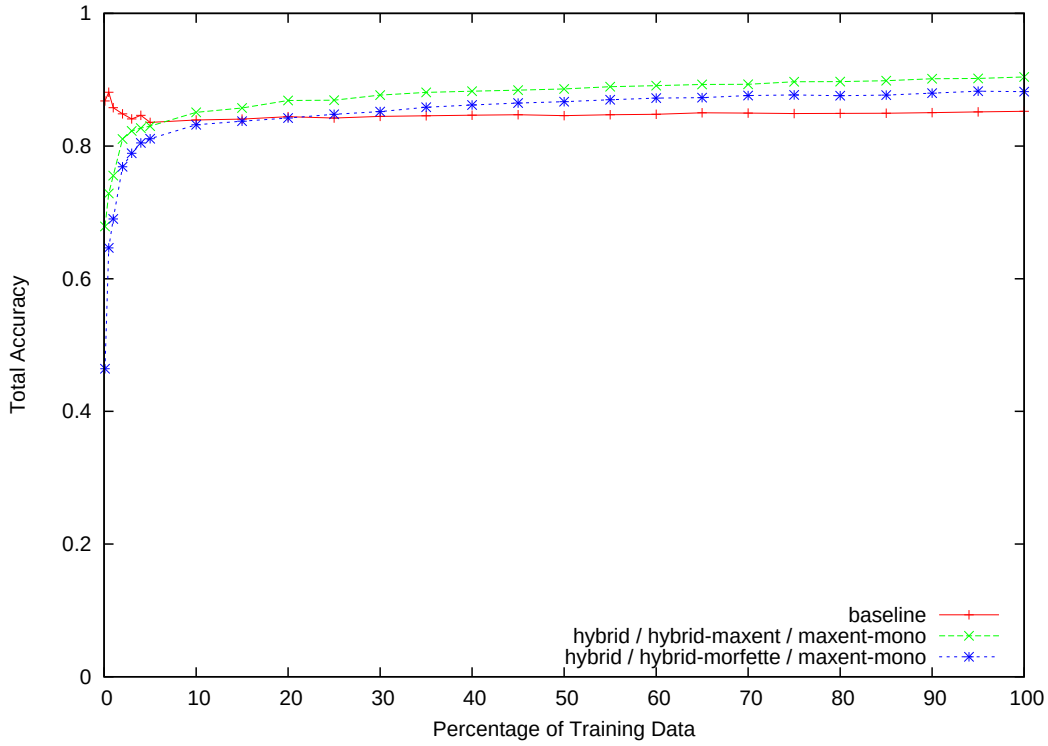

Figure 4.11: The learning curve for the known accuracy of stem tagger in the joint pipeline.

4.2.3 Unknown Accuracy

The final five Figures (4.12, 4.13, 4.14, 4.15, 4.16) show the learning curves of unknown accuracy for each of the sub-models. It is in these learning curves that the greatest difference between baseline and **prob-morph** approaches can be observed. Figure 4.12 shows a very large difference between the baseline approaches and the joint models; the joint models seem to perform equally well for segmentation. For baseform linkage (Figure 4.13), the baseline performs very well with little data, because the unknown approach for baseline baseform linkage is to predict the stem as the baseform. This is a good naïve approach and is true for many non-rare words. With little training data, however, many words which are generally known happen to be unknown, due to the small amount of training data. Therefore, as the number of unknown words diminishes over time (as training data increases), this naïve approach worsens. The same sort of effect is seen in root linkage in Figure 4.14. These two figures also show just how much better **hybrid-morfette** performs than **hybrid-maxent**. For suffix tagging, both joint pipeline approaches are nearly equal, and the baseline is non-existent, since all suffixes in the test data had been seen during training. The final graph of unknown accuracy for stem tagging shows the same decay in the baseline as the amount of training data grows larger and larger.

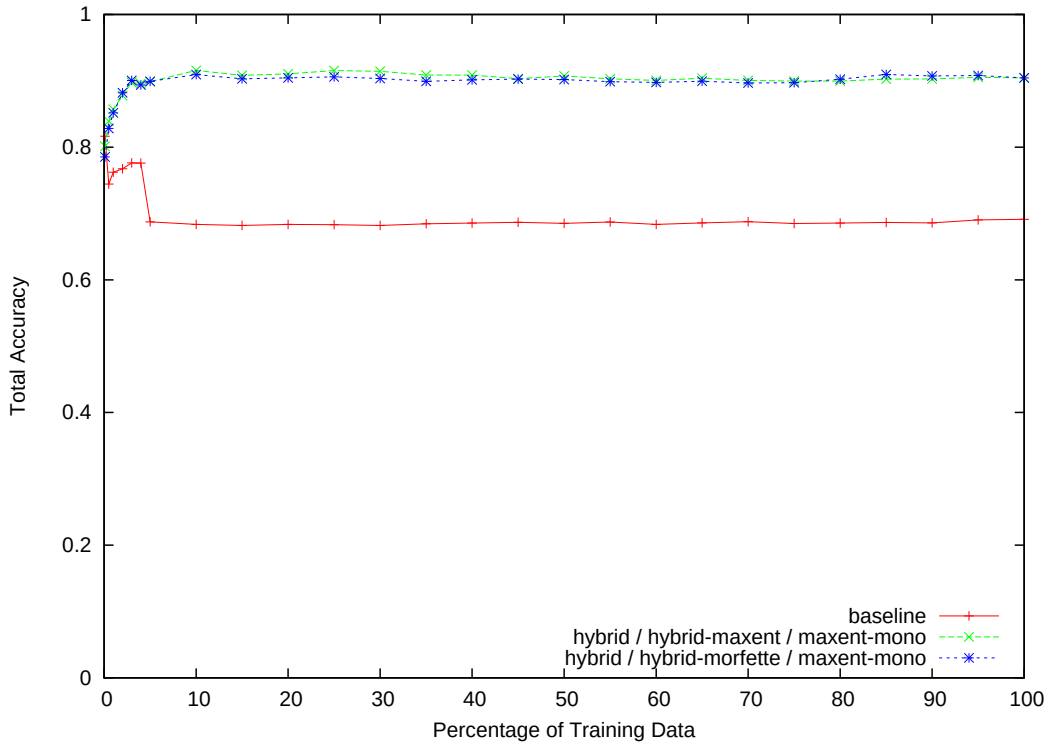

Figure 4.12: The learning curve for the unknown accuracy of segmentation models in the joint pipeline.

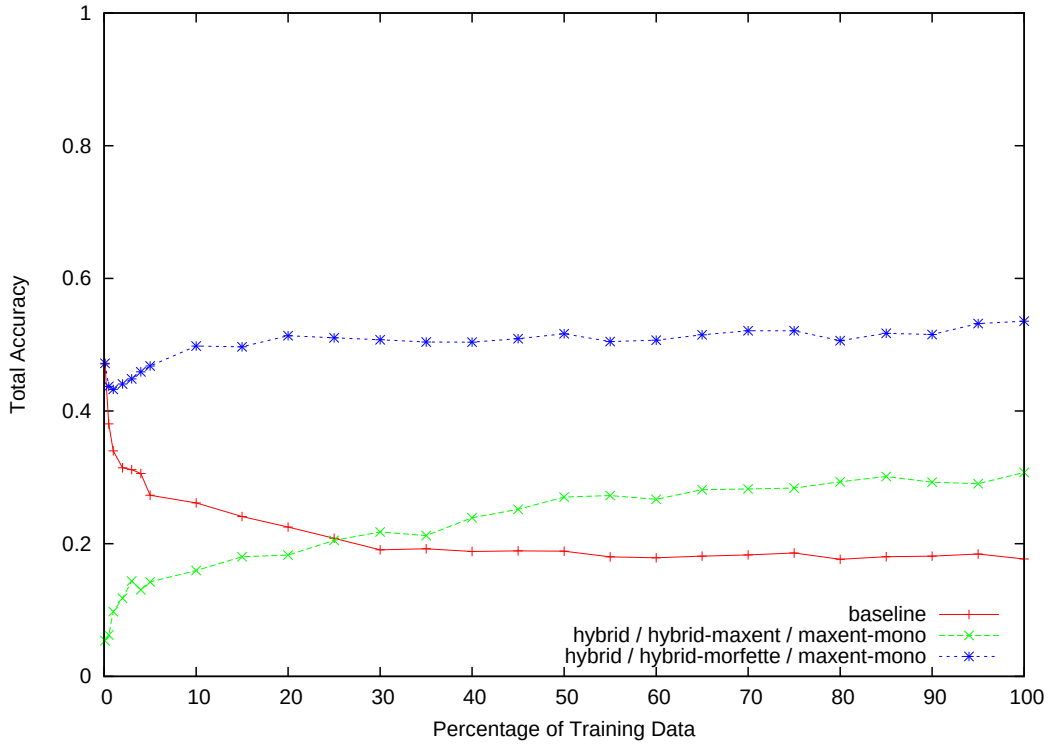

Figure 4.13: The learning curve for the unknown accuracy of baseform linker in the joint pipeline.

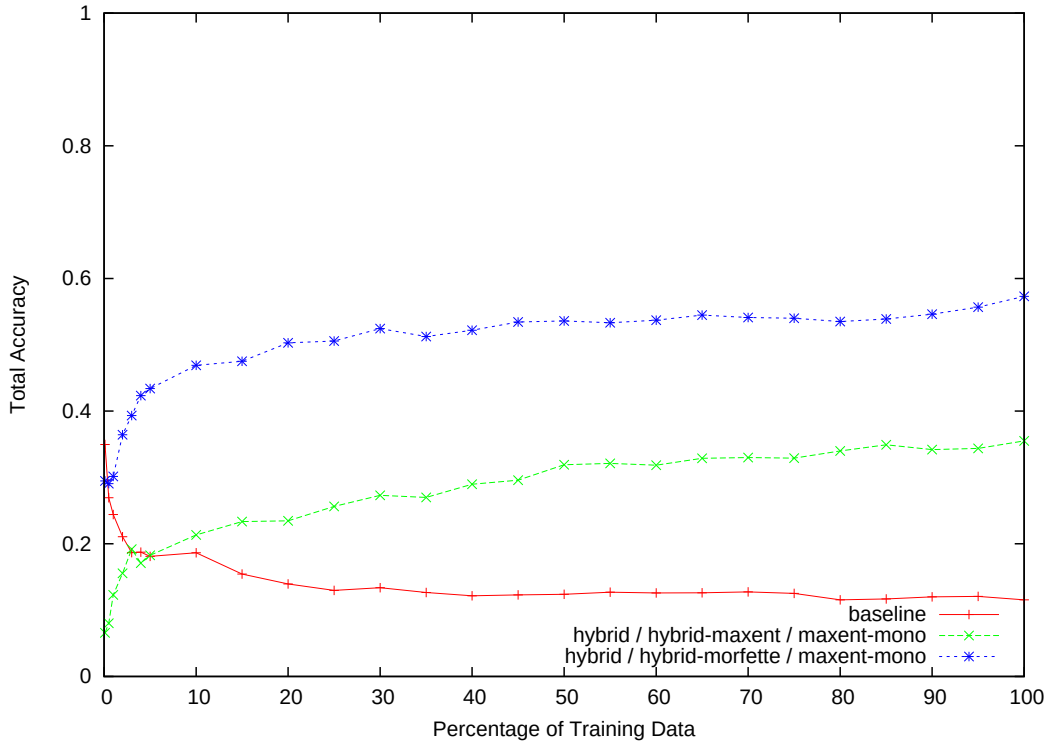

Figure 4.14: The learning curve for the unknown accuracy of root linker in the joint pipeline.

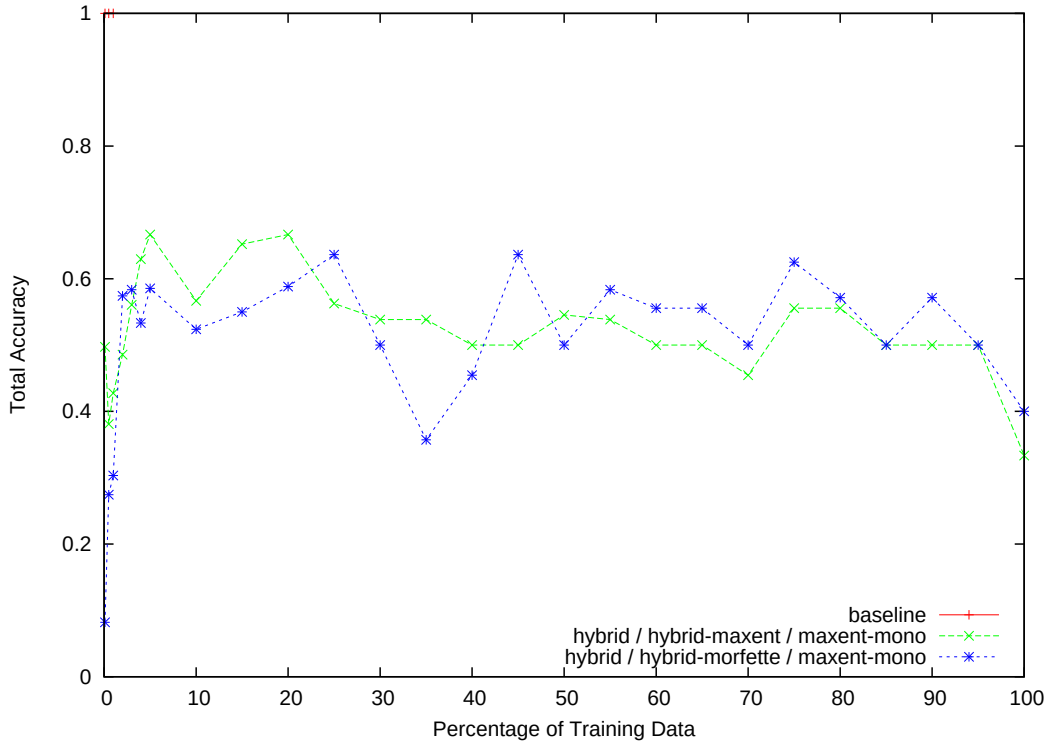

Figure 4.15: The learning curve for the unknown accuracy of suffix tagger in the joint pipeline

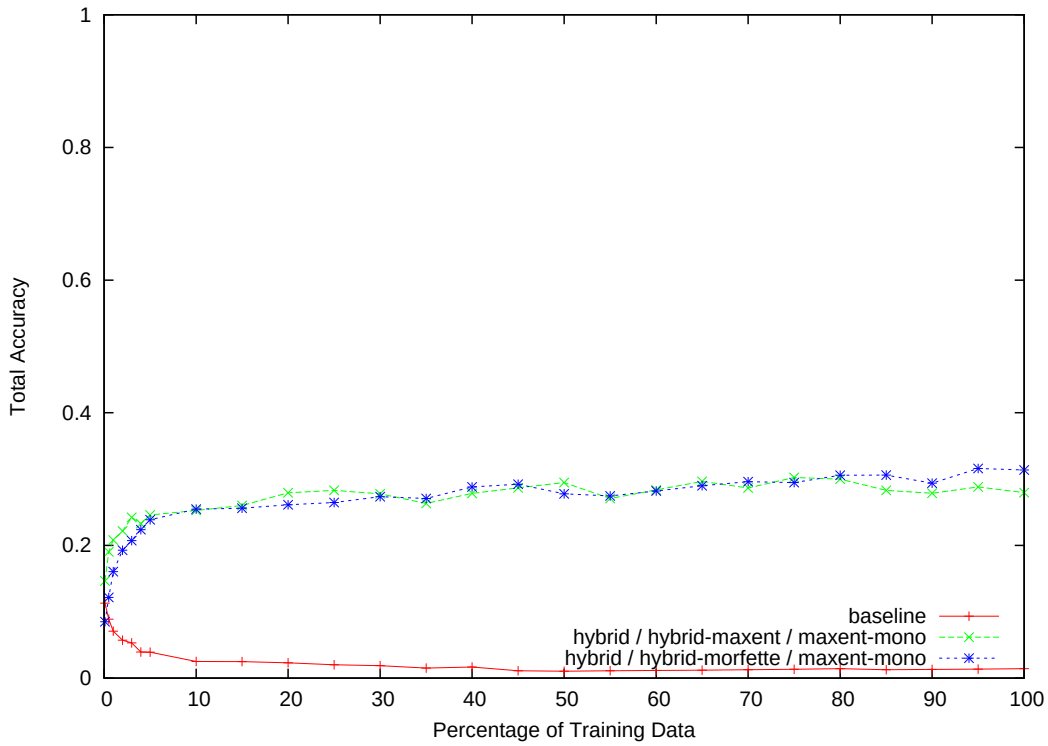

Figure 4.16: The learning curve for the unknown accuracy of stem tagger in the joint pipeline.

4.3 Arc N and State N

Briefly mentioned in chapter 3 was the notion of two parameters b and n that we now refer to as Arc N (b) and State N (n). Arc N is the parameter that represents the beam with during Viterbi decoding. State N represents the number of contexts to keep after each sub-model has finished predicting. We swept each of these values between 2 and 40 and computed the decision-level metrics (**total-decisions**, **na-coverage**, **na-accuracy**), as well as the token-level accuracies (total, known, and unknown). Figures 4.17, 4.18, and 4.19 show the results for the decision-level metrics and Figures 4.20, 4.21, and 4.22 show the total, known, and unknown accuracy, respectively.

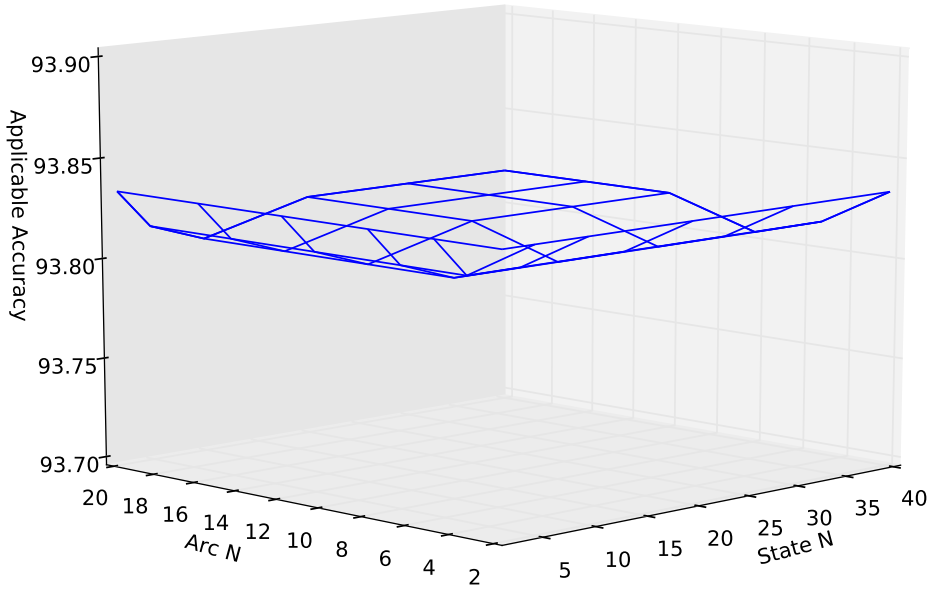

Figure 4.17: The decision-level accuracy (**total-decisions**) for various values of Arc N and State N.

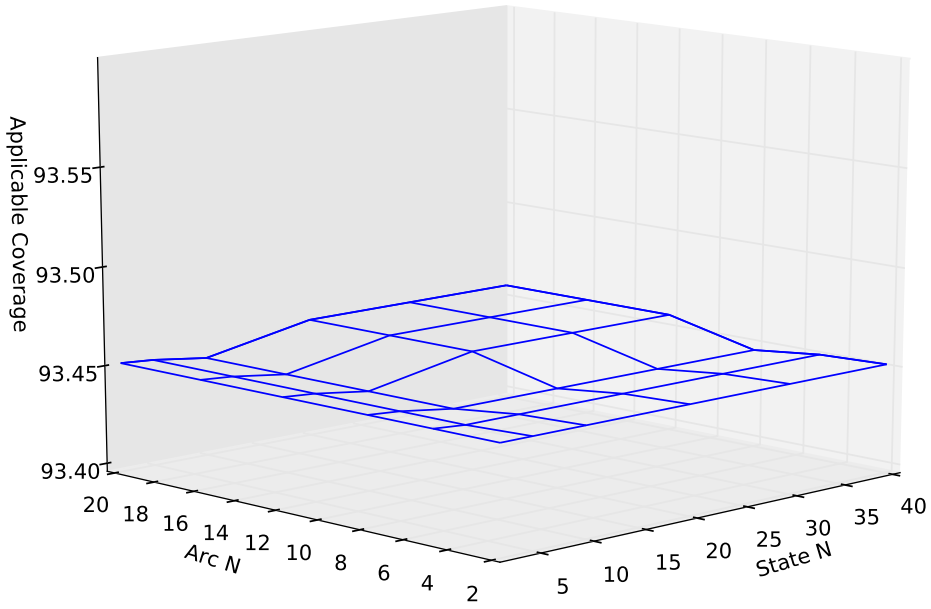

Figure 4.18: The decision-level coverage (**na-coverage**) for various values of Arc N and State N.

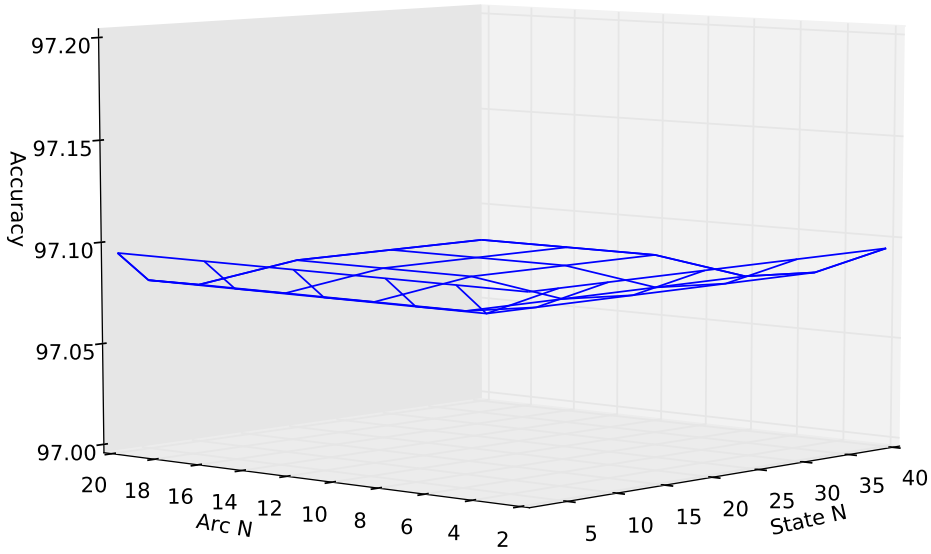

Figure 4.19: The decision-level N/A accuracy (**na-accuracy**) for various values of Arc N and State N.

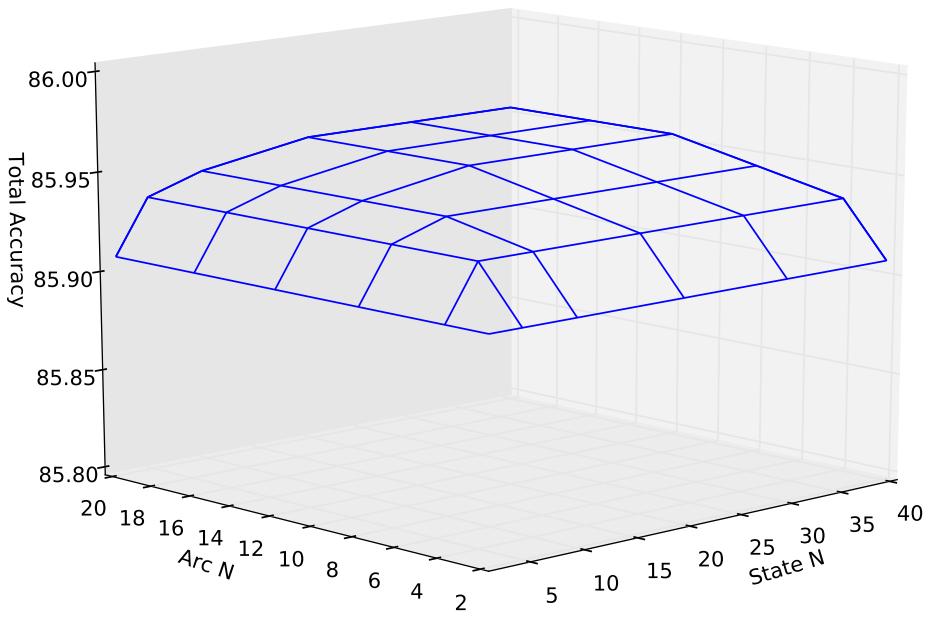

Figure 4.20: The total accuracy for various values of Arc N and State N.

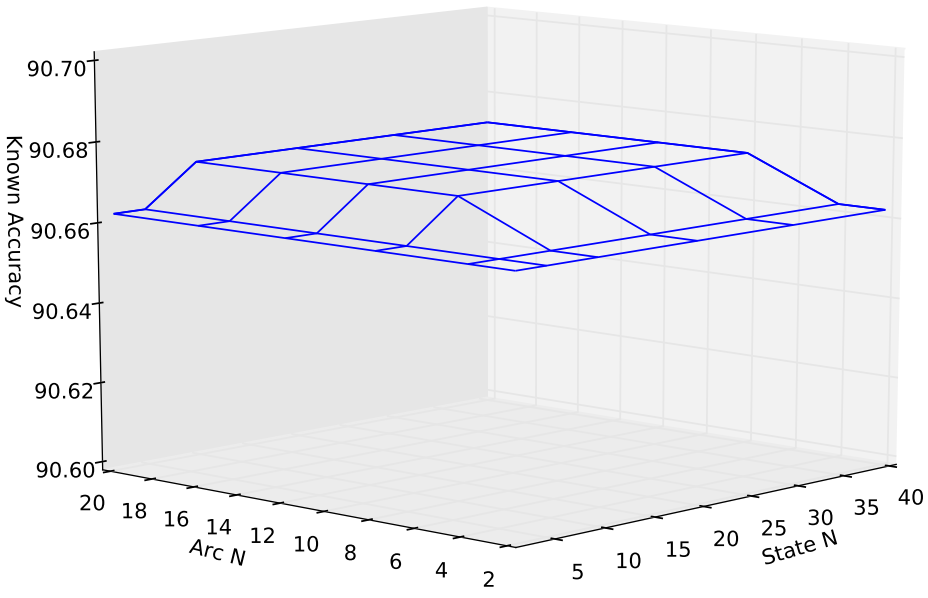

Figure 4.21: The known accuracy for various values of Arc N and State N.

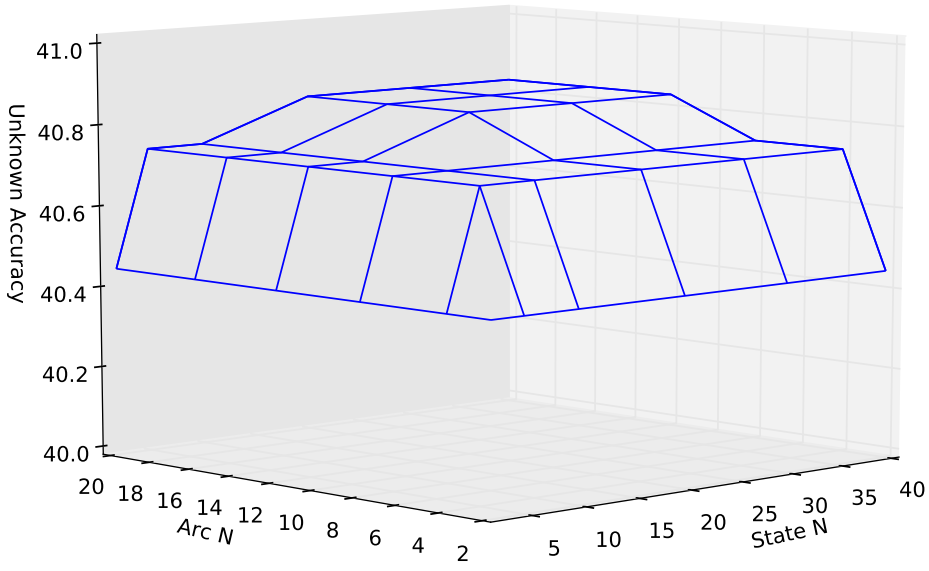

Figure 4.22: The unknown accuracy for various values of Arc N and State N.

4.4 Decision-Level Analysis

Learning curves for the decision-level accuracies are not shown in chapter 3 due to space constraints. In this section we present the learning curves for all three decision-level metrics. For each of these metrics, there exists a learning curve for total accuracy, known accuracy, and unknown accuracy. These are shown in Figures 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.30, and 4.31. These results are quite similar to the learning curves of the token-level accuracies; however, with respect to decision-level metrics, **hybrid-maxent** only slightly outperforms **hybrid-morfette** with the full amount of training data. The crossover point between **hybrid-maxent** and **hybrid-morfette** is also much later on the learning curve. As seen before, **hybrid-morfette** is clearly better for unknown words throughout the entire process. All **prob-morph** models also outperform the baseline after 10%. This indicates that it is advantageous to use **prob-morph** models over the baseline when considering the number of decisions correctly chosen.

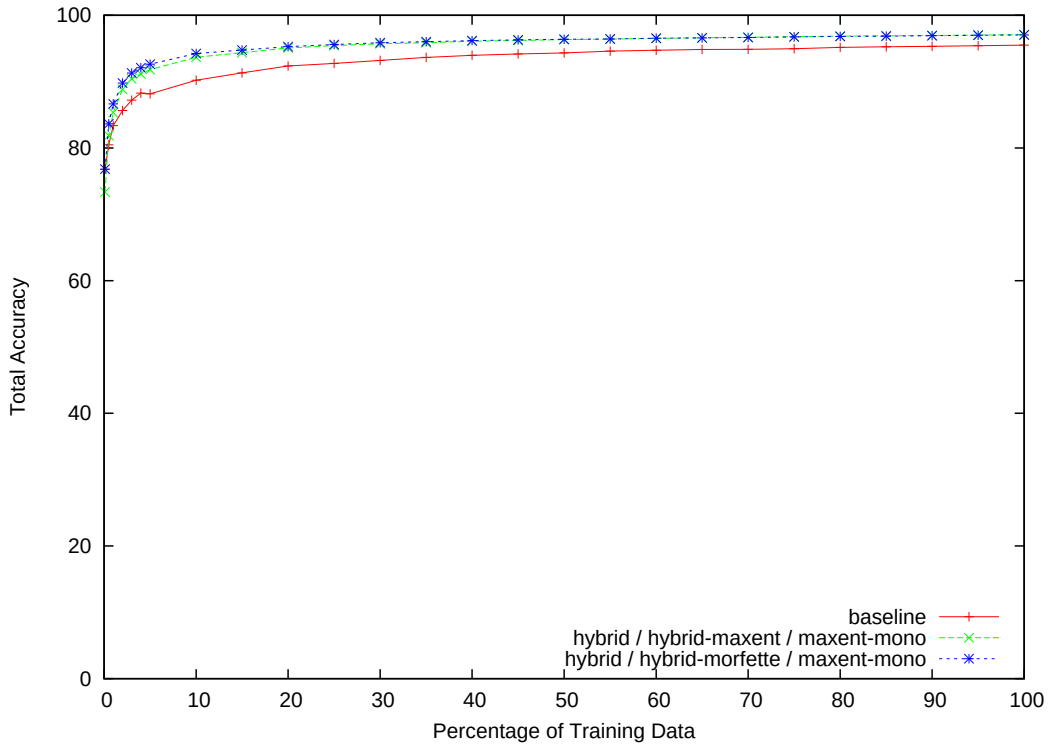

Figure 4.23: The learning curve for decision-level total accuracy.

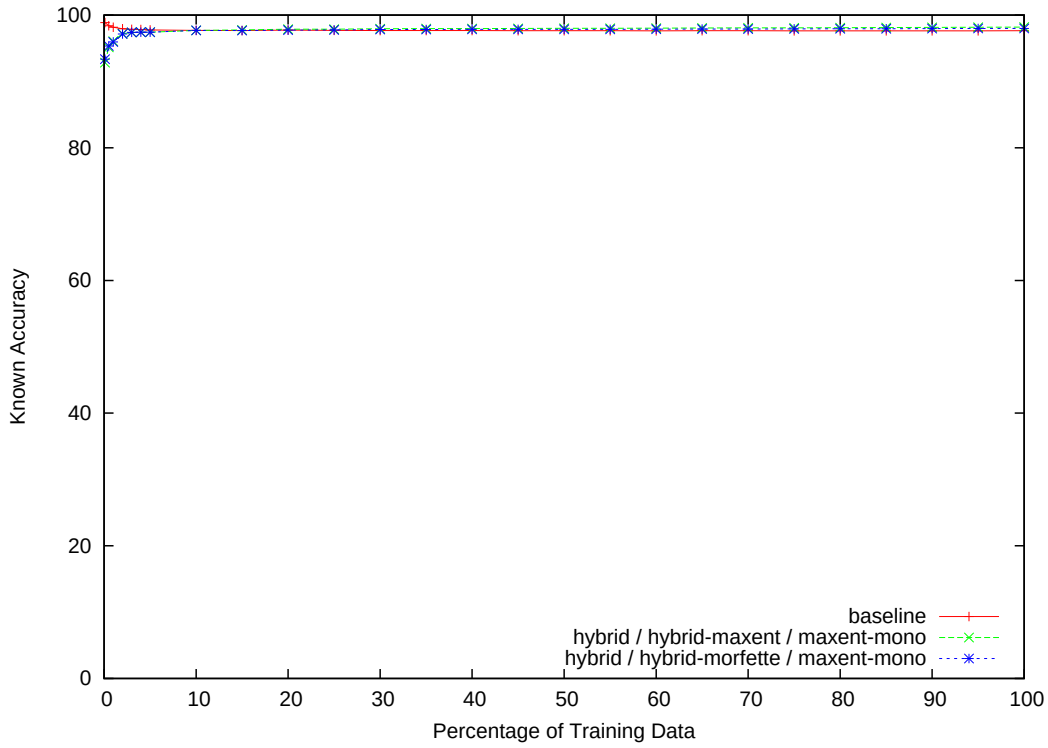

Figure 4.24: The learning curve for decision-level known accuracy.

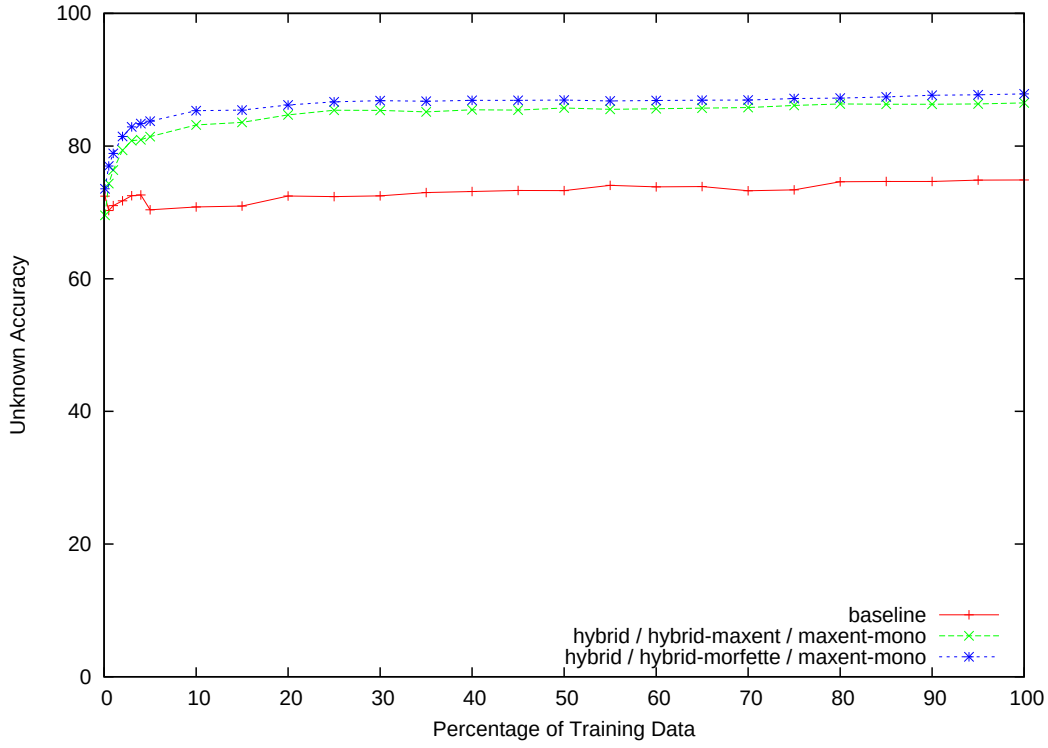

Figure 4.25: The learning curve for decision-level unknown accuracy.

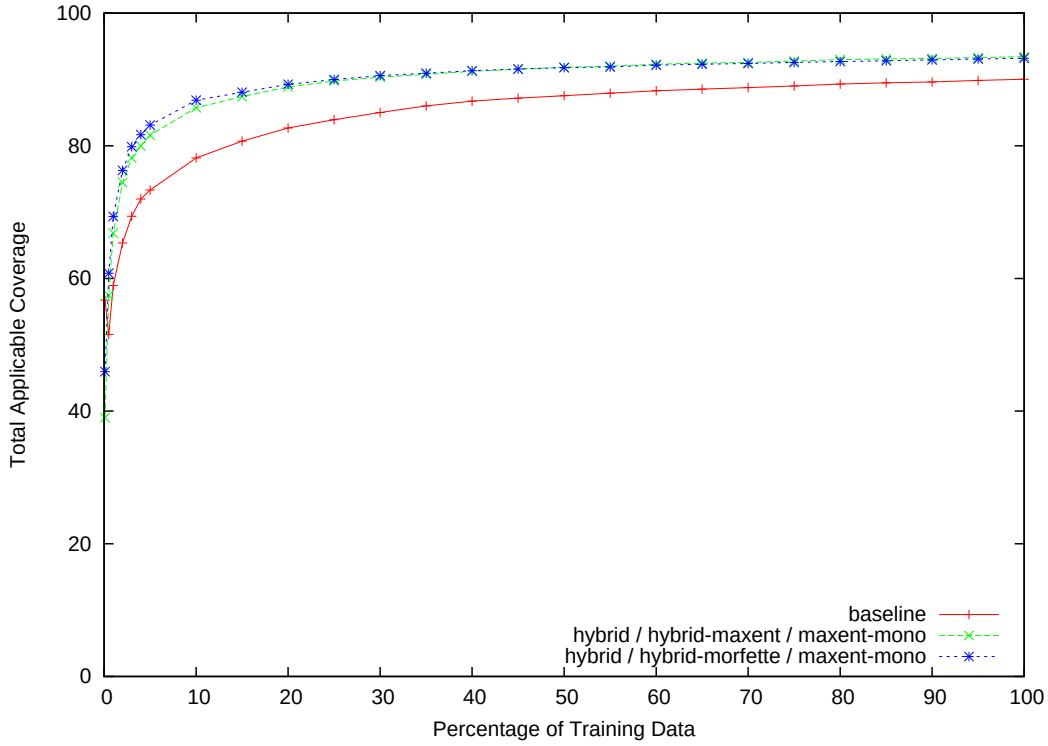

Figure 4.26: The learning curve for decision-level total applicable coverage.

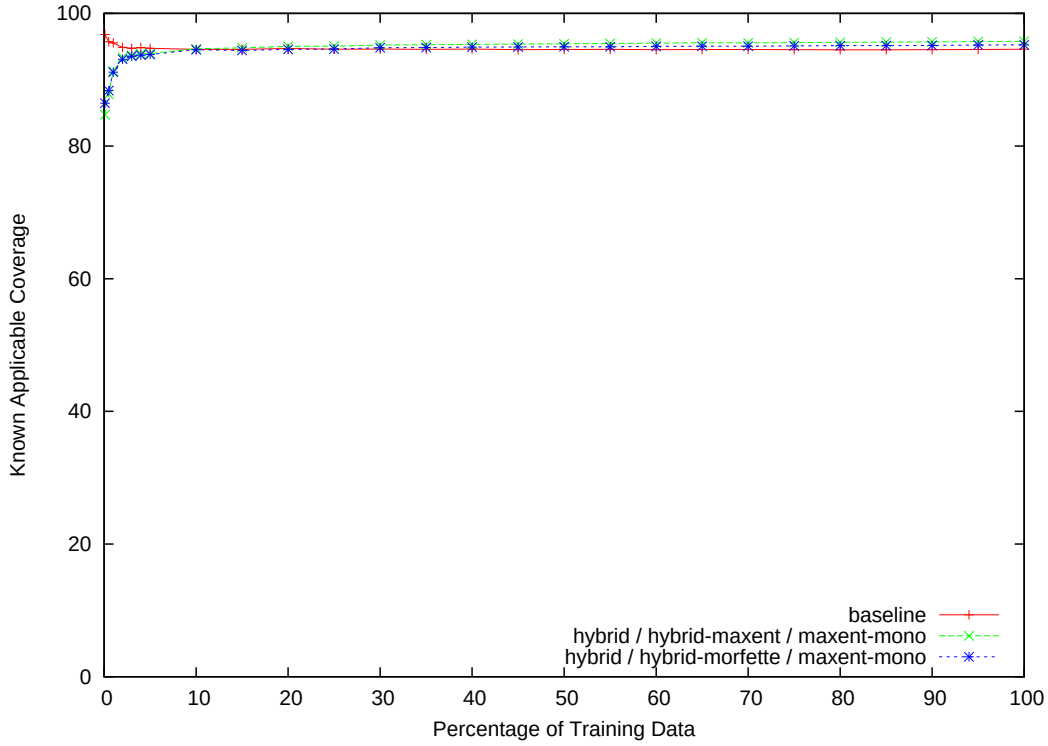

Figure 4.27: The learning curve for decision-level known applicable coverage.

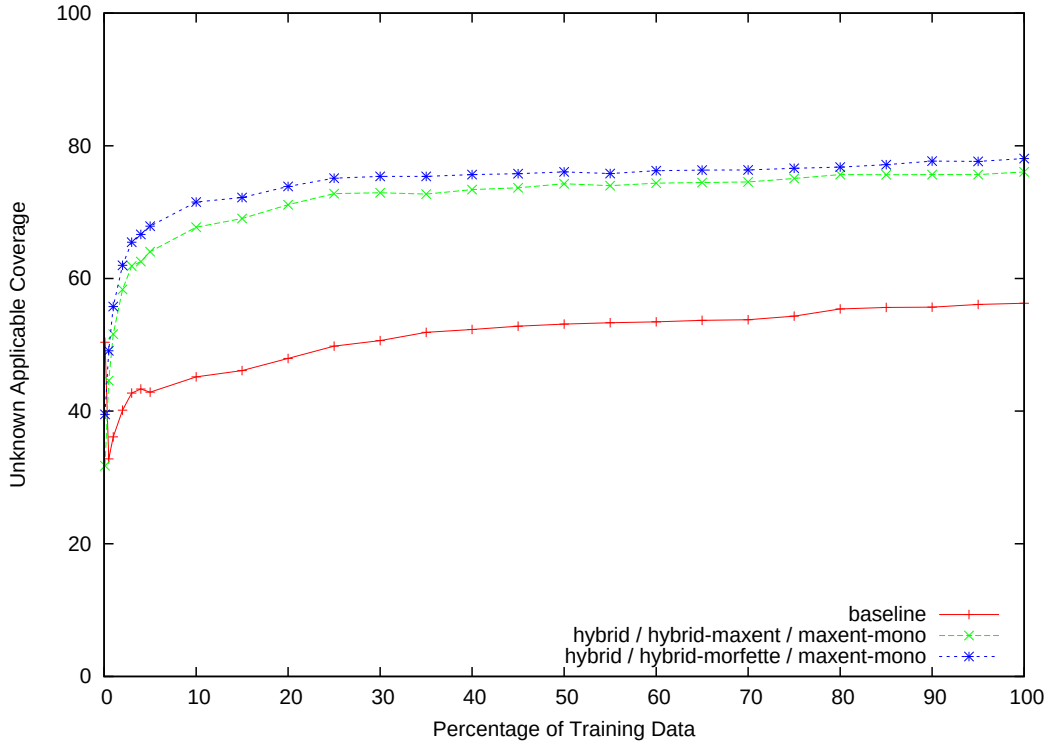

Figure 4.28: The learning curve for decision-level unknown applicable coverage.

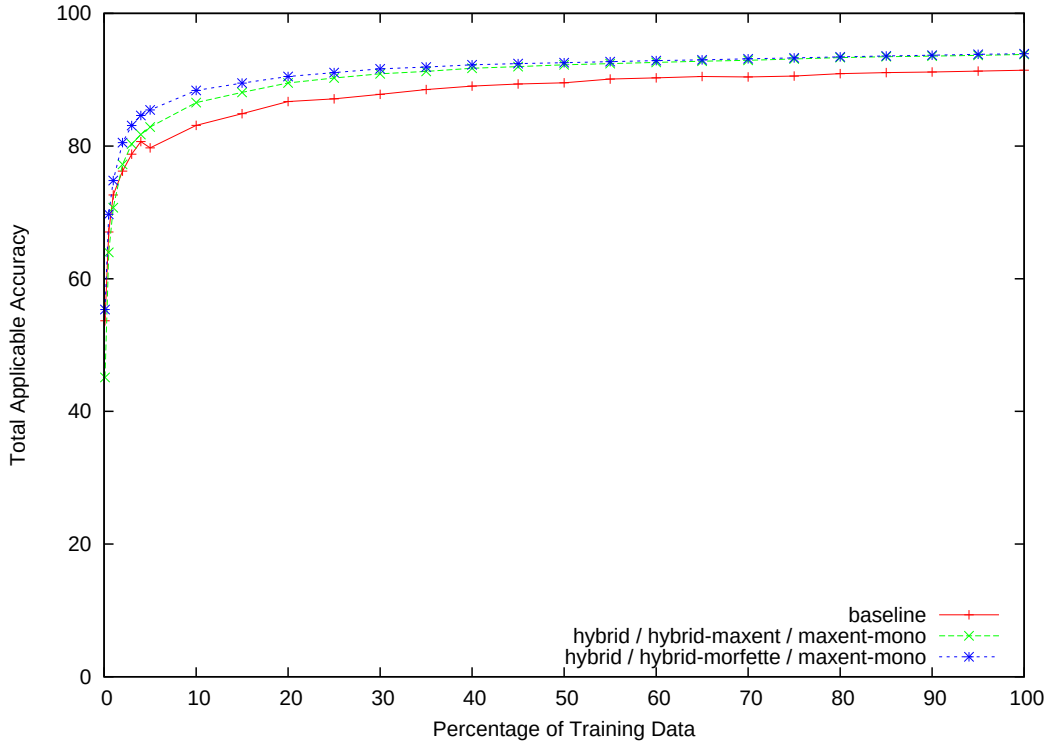

Figure 4.29: The learning curve for decision-level total applicable accuracy.

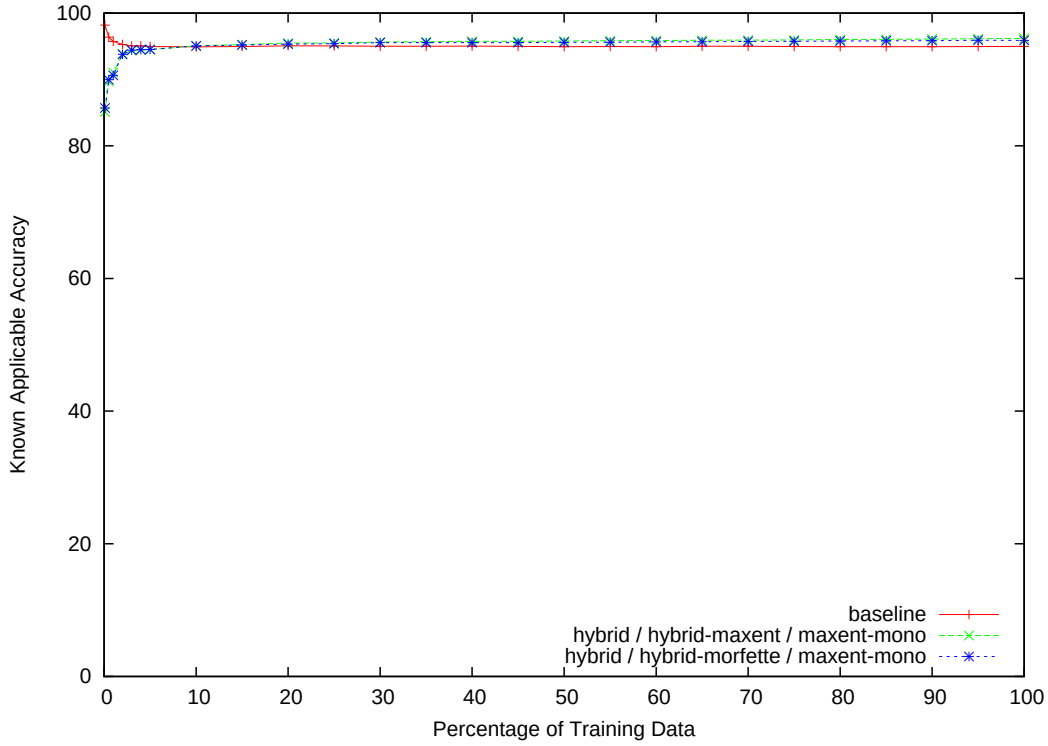

Figure 4.30: The learning curve for decision-level known applicable accuracy.

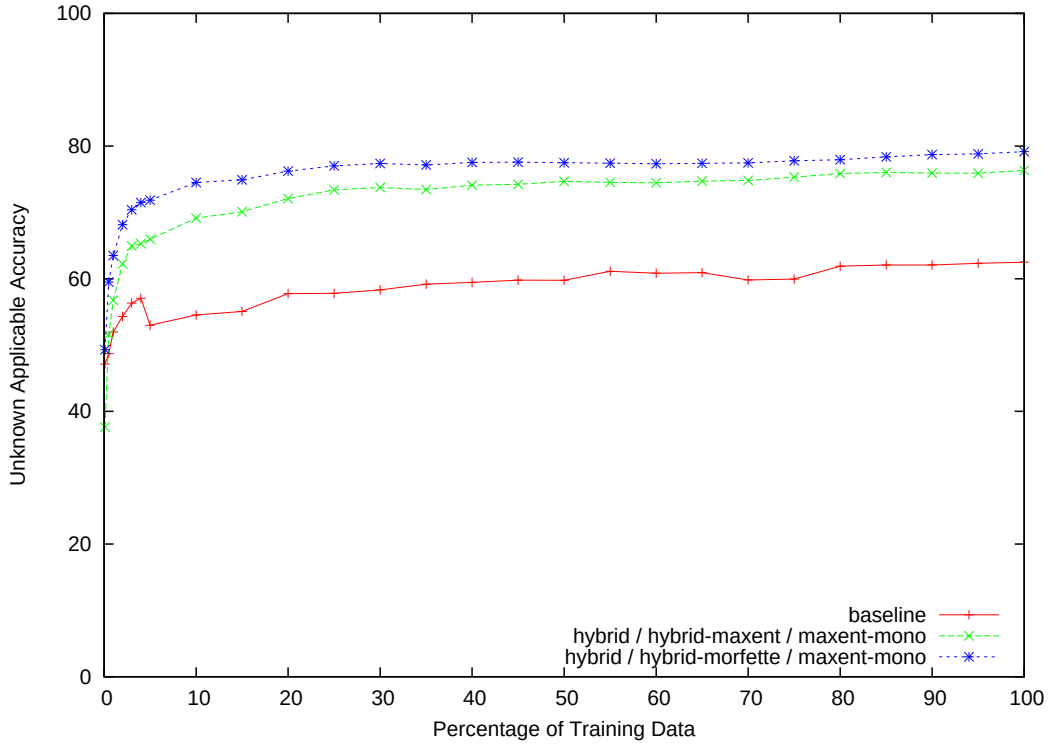

Figure 4.31: The learning curve for decision-level unknown applicable accuracy.

4.5 Statistical Validation

In order to determine if the **prob-morph** models out-perform the naïve approaches, we use statistical validation methods to compare the different algorithms. We compare the results of the best-known joint pipeline model (**hybrid** segmenter, **hybrid-maxent** baseform and root linkers, and **maxent-mono** taggers) to the baseline pipeline model. The baseline pipeline model uses the respective baselines in place of the **prob-morph** models, keeping the same dependencies between sub-tasks.

For validation, we use a paired t-test, with the pair of each sample as the accuracy for each model. For each model, we perform 10-fold cross validation, giving 10 samples for each accuracy level. We found that from training-data sizes of 10% (about 9,900 word tokens) and above, both Morfette and maxent approaches significantly outperform the naïve approach. Therefore, as long as there are approximately 9,900 or more annotated tokens, it makes sense to use the **prob-morph** models. Until that point, the baseline approach performs as well as the more **prob-morph** approach.

Chapter 5

Conclusions and Future Work

We have shown that a probabilistic data-driven framework can be used for segmentation, dictionary linkage, and morphological tagging for Syriac, and that it can significantly outperform the naïve approach. We have introduced novel approaches for segmentation, dictionary linkage, and morphological tagging. Each of these approaches has outperformed its corresponding baseline. Furthermore, for Syriac, a data-driven approach seems to be an appropriate way to solve the problem of segmentation, dictionary linkage, and morphological annotation in an under-resourced setting, especially when considering the total number of correct decisions, and not solely the accuracy of the entire label.

The Maxwell Institute and the Oriental Institute at Oxford University will use this combined model for pre-annotation 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 these issues. In addition, there is much to do in getting the entire tag accuracy closer to the accuracy of individual decisions. Feature engineering, specifically the stem tagging, or introducing new morphological tagging techniques seem the logical place to start, as this sub-task under performs all others.

References

Eiman Al-Shammari and Jessica Lin. A novel Arabic lemmatization algorithm. In *AND '08: Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*, pages 113–118, New York, NY, USA, 2008. ACM.

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

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

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

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

Gregory Crane. The Perseus project and beyond: How building a digital library challenges the humanities and technology. *D-Lib Magazine*, 1998.

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

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

Anna Feldman and Jirka Hana. *A Resource-Light Approach to Morpho-Syntactic Tagging*. Rodopi, Amsterdam/New York, NY, 2010.

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

Nizar Habash and Owen Rambow. 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, 2005.

Nizar Habash and Owen Rambow. 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, April 2007. URL `http://www.aclweb.org/anthology/N/N07/N07-2014`.

Robbie Haertel, Peter McClanahan, and Eric K. Ringger. 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, June 2010. URL `http://www.aclweb.org/anthology/N10-1076`.

Jan Hajič and Barbora Hladká. 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, 1998.

Jaroslava Hlaváčová. Morphological guesser of Czech words. In *TSD '01: Proceedings of the 4th International Conference on Text, Speech and Dialogue*, pages 70–75, London, UK, 2001. Springer-Verlag. ISBN 3-540-42557-8.

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

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

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In *EMNLP 2004: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, 2004.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. 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, 2003.

Saib Mansour, Khalil Sima'an, and Yoad Winter. 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, 2007.

Marianne McDonald. The Thesaurus Linguae Graecae: Project history, October 2009. http://www.tlg.uci.edu/about/history.php.

Emad Mohamed and Sandra Kübler. 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, June 2010.

Emad Mohamed and Sandra Kbler. Arabic part of speech tagging. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), May 2010.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. Chinese segmentation and new word detection using conditional random fields. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 562. Association for Computational Linguistics, 2004.

Vladimír Petkevič. 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, London, UK, 2001. Springer-Verlag.

Wido Van Peursen, November 2009. http://www.hum.leiden.edu/religion/research/research-programmes/antiquity/turgama.html.

Monica Rogati, Scott McCarley, and Yiming Yang. Unsupervised learning of Arabic stemming using a parallel corpus. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 391–398. Association for Computational Linguistics, 2003. doi: http://dx.doi.org/10.3115/1075096.1075146.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 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, 2005.

K. Toutanova and C. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP 2010: Proceedings of the 2000 Conference on Empirical Methods*

in Natural Language Processing, pages 63–70, 2000. doi: http://dx.doi.org/10.3115/1117794.1117802.

Emanuel Tov, editor. *The Dead Sea Scrolls Electronic Library*. Brill, 2007.

Shuly Wintner. Hebrew computational linguistics: Past and future. *Artificial Intelligence Review*, 21(2):113–138, 2004.