UNIVERSITY OF CALIFORNIA, SAN DIEGO

Part of Speech Tagging of Levantine

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Azadeh Monirabbassi

Committee in charge:

Professor Garrison W. Cottrell, Chair
Professor Roger Levy
Professor Sanjoy Dasgupta

2008

The thesis of Azadeh Monirabbassi is approved and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2008

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT OF THE THESIS

Part of Speech Tagging of Levantine

by

Azadeh Monirabbassi

Master of Science in Computer Science

University of California San Diego, 2008

Professor Garrison W. Cottrell, Chair

The goal for this project is to explore strategies in adapting a Part of Speech (POS) tagger that was trained on Modern Standard Arabic sentences for tagging Levantine sentences, a dialect of Modern Standard Arabic, leveraged by methods of morphological analysis. I propose a tagging model that supports an explicit representation of the root-template patterns of Arabic. I will analyze the functionality and performance of the algorithms, and will compare the results. In leveraging the MSA POS tagger for tagging Levantine data, I achieved a peak accuracy of 73.28% which is 6% higher than the baseline for a standard Hidden Markov Model based tagger.

1

Introduction

1.1 Goals of This Work

Part-of-Speech Tagging is the process of labeling or tagging the words of a text corpus according to a set of grammatical rules based on the individual and contextual roles of the words. Here are some part-of-speech tag examples: singular proper nouns such as *Obama* (NNP), possessive personal pronouns such as *mine* (PRP$), and adjectives such as *rebellious* (JJ). This task is subject to all of the hurdles encountered by any other statistical classification method. Part-of-speech tagging is not an easy task due to the fact that different words can take on different tags at different times because of their semantic ambiguity. For example, the word 'plays' is usually viewed as the third person of the verb 'play', but it could also be the plural of the noun 'play' in the context of a theatrical act. In order to best tackle this problem we need a good model to represent the data and a good

methodology for part-of-speech tagging.

The Arabic language is a language of multiple dialects with just one writing standard. For this reason there are not enough standard corpora of the different dialects of Arabic for natural language processing purposes other than informal blogs, emails, and other online materials. Also, these informal materials are not written with a uniform or standard rule. The news, the media and all of the official affairs are documented in Modern Standard Arabic (MSA) which is not spoken by anyone and is not acquired as a child's first language. While there is sufficient annotated MSA data, very little annotated data exists in the dialects.

Here I consider Levantine Arabic, a group of Arabic dialects spoken in the 100 km-wide eastern-Mediterranean coastal strip known as the Levant, i.e. in Syria, Palestine, western Jordan and Lebanon. In this paper I address the problem of tagging transcribed spoken Levantine Arabic (LA). This work is based on the assumption that it is easier to manually create new resources that relate LA to MSA than it is to manually create syntactically annotated corpora in LA. This approach assumes the existence of annotated LA corpus only for development and testing.

This report is organized as follows. I first discuss related work and available corpora and present linguistic issues in LA and MSA. I then discuss the approaches I took for part-of-speech tagging of Levantine using MSA corpora and different methods of morphological analysis that boost tagging accuracy. I summarize and discuss the results in section 3.2.

1.2 Linguistic Facts

Levantine is a dialect of Arabic spoken in Jordan, Syria, Lebanon, and Palestinian territories. Although all native Arabic speakers speak a regional derivation of Arabic, Modern Standard Arabic is the official language and literary standard used in books, newspapers, official documents, as well as the language taught in all levels of school. Written documents of the colloquial dialects do not exist in abundance except for plays, some poetry, blogs, and emails. Except for the primary school books for children, most MSA literature is unvocalized, meaning that the short vowels are missing (i.e. a, e, i, o, and u). In MSA, *qatala* means 'he killed', and *qutila* means 'he was killed'. These examples are both vocalized. The unvocalized version of both of the previous examples is *qtl*. Here I discuss some of the differences and similarities between the Levantine dialect and MSA. I will touch on the differences in the sound system, and emphasize grammatical and lexical differences, which have a significant influence on the accuracy of this task. The following are the most significant comparisons between these two languages.

MSA's word order is Verb-Subject-Object whereas the usual word order for Levantine is Subject-Verb-Object. The following examples, transliterated with the Buckwalter transliteration scheme, show the same sentence in VSO and SVO word orders: *yaktubuwna Almodar~isuwna Alr~isAlAti* 'write the teachers the letters' is in VSO form and *Almodar~isuwna yaktubuwna Alr~isAlAti* 'the teachers write the letters' is in SVO form.

MSA nouns have three cases, nominative, genitive, and accusative. Normally singular nouns take the endings *u*, *i*, and *a* in the definite form and *N*, *K*, and *F* in the indefinite form for nominative, genetive, and accusative respectively. These endings, which play a grammatical role, are referred to as case endings. Case endings are present only in formal or literary language with the exception of indefinite accusative nouns ending in any letter but *ta marbuta*, or *hamza*, in which case the *F* is written on the top of the letter *alif* added to the end of the word with *alif* still showing up in unvowelled texts. In short, any words ending in the letter *ta marbuta* or the letter *hamza* get an *alif* added with an 'F' on top. This makes our job easier when dealing with the unvocalized version of the MSA and Levantine data sets since for the most part, Levantine has lost the case endings for the nouns with the exception of the *F* that sits on the top of the letter *alif*. Without the case endings the Levantine and MSA versions of a given word are more likely to be similar. In MSA, nouns are marked for definiteness and indefiniteness. The definiteness is marked by the article *Al-*, while indefiniteness is usually indicated by the suffixes *N*, *-F*, and *K*, which follow the case marker as mentioned earlier. Levantine nouns are marked for definiteness and indefiniteness, as they are in all varieties of Arabic. Although there are different variations of the definite article *Al* in Levantine, such as *l-*, *il-*, and *li-* depending on the following consonant, our Levantine development data set only uses the *Al-* article for consistency and convenience purposes. For example *Al-kuwiyt* in MSA becomes *li-kwiyt* in Levantine which means 'Kuwait'. Here are some examples of definite and indefinite markers

in MSA and Levantine. In MSA, *Al+kitAb+a* 'book' is Accusative + definite and *mAdiy+AF* 'materialistic' is Accusative + indefinite. In Levantine, *Al+kteb* 'book' is definite and *mAdiy+AF* 'materialistic' is indefinite.

In my project, I am not using the extended POS tags which represent number, gender, subject, object, etc. If I were, the lack of case endings in Levantine would hurt my project since case endings are indicative of the POS roles represented by the extended POS tags. Instead, in Levantine sometimes the lack of some case endings is compensated by the addition of the same vowel or a different one somewhere before the last consonant. For example, the word *>noti* 'you' (feminine) becomes *<not*. In this case, *>* is phonologically the same as the case ending *i*. There are many such differences between Levantine and MSA at different positions in a word, which make the same word (a word of the same root) look different in the two languages. Fortunately due to the usage of the unvocalized MSA training and Levantine development datasets, this inconsistency does not cause any problems, except in cases where the difference is caused by long vowels which are not omitted in the unvocalized version as in the example of the word book. In MSA unvocalized version of the word book is *ktAb*, and in Levantine it is *ktb*.

MSA nouns have three numbers, singular, dual, and plural. The dual number is missing in Levantine Arabic. For example, instead of the word *$Aqilayn*, which means '(two) workers', *$Aqiliyn*, which means 'workers' is used in Levantine.

Levantine Arabic has two genders: masculine and feminine, which are ex-

pressed as suffixes following nouns and circumfixes surrounding verbs. Gender distinctions are absent in the plural form in Levantine unlike in MSA.

Some of the most significant features of the verb system in Arabic are person, number, tense, and aspect which are marked by prefixes and suffixes. There is one basic stem (form 1) plus nine derived stems, each with a range of meanings, such as reflexivity, and causativity. Each form has its own set of active and passive participles and verbal nouns.

In Levantine, verbs always maintain their plural agreement whereas in MSA the singular form is used when the verb precedes the subject, which is usually the case since MSA's word order is VSO. Also in Levantine, in the imperfect tense, the present aspect marker *b* is used which does not exist in MSA. In addition, the template vowels used for the same verb tense are sometimes different in MSA and Levantine. A combined example for all three is *ya$rabuwna AlrijAl* 'the-men drink' in MSA, which becomes *AlrijAl byi$rabuw* in Levantine. In the Levantine dialect a vowel and a consonant of a word stem can switch positions. This only occurs in the 'they' and 'you' (plural) forms of the imperfect tense, with the exception of cases in which the template vowel of the word is *a*. For example, *byuktubuw* 'they write' becomes *byukutbuw*, but *byi$rabuw* 'they drink' remains unchanged.

In the Levantine dialect, among the negative particles are *mi$* as in *huwe mi$ huwn* or 'He is not here', and a set of conjugated words such as *manni* and *manna* meaning 'I am not' and 'we are not' respectively. These particles are absent in MSA and instead the verb *layosa*, or 'is not' is conjugated for all pronouns to

convey non-existence. The letter *$* is another negative particle in Levantine that negates a verb as in *hum byuHub∼uw $ Al$∼ugl hdA*, 'they do not like this job'. In MSA the negative particle *ma* and *la* are exclusively used for the past and present tenses respectively, and serve as a prefix. In Levantine the negative particle *ma* is used to negate both the past and the future tenses. In addition, in Levantine some verbs are negated using the circumfix *m + $* which is absent in MSA.

As I mentioned earlier, the prefix *b* is a present tense indicator in Levantine, but it is absent in MSA where the template of the verb dictates the tense. The following examples show the difference between the present tense of the verb 'to travel' in Levantine and MSA. *hm bsAfrw* means they travel in Levantine when *hm ysAfrwn* means the same thing in MSA.

1.3 Related Work

In the previous work on Levantine using the MSA training corpus done by Rambow et al.[5], three different goals were met: lexicon induction from corpora, POS tagging, and parsing of Levantine. In the first task the goal was to build a lexical or translation mapping between the two languages' (language and dialect) corpora. Given such a translation lexicon, a probability was assigned to every translation pair taking the statistics of the two corpora into account. The Expectation-Maximization algorithm was used with the assumption that the pair of corpora was a sample from a weighted mixture of two conditional distributions

in which the words of one corpus were generated given the words of the other corpus under the given translation lexicon. The accuracy of the results depended upon the distribution of frequent words which in turn is influenced by the genre similarity of the corpora. This was done on comparable and unrelated corpora and proved to be a difficult task. The content and the size of the corpora are other factors that contribute to the accuracy. One method of partially normalizing for the factors above that was used was extraction from the larger of the two corpora in order to better match the distributions of the two corpora which resulted in an improvement in accuracy.

The second task, POS tagging, dealt with the adaptation problem, adapting the MSA tagger for Levantine, with different sources of information; basic linguistic knowledge about both languages, varying sizes and qualities of MSA-Levantine lexicon, and a small tagged Levantine corpus. The results showed that the small and high-quality lexicon of the most common Levantine words yielded the biggest boost in the accuracy.

The final experiment was parsing Levantine using an MSA parser. To compensate for the lack of parallel corpora, explicit linguistic knowledge was used to perform sentence transduction, treebank transduction, and grammar transduction.

From all of the above approaches it was concluded that better accuracy could be achieved by allowing the models to exploit knowledge about the language, hence the goal of this project is to exploit morphological acquisition.

In the approach taken by Habash and Rambow[6], tokenizing and morpho-

logically tagging are the same operation consisting of three phases: obtaining a list of all possible analyses for the words in a sentence from the morphological analyzer, applying the classifiers for selected morphological features to words of the text, and using the output of the classifiers to choose from the analyses returned by the morphological analyzer. The algorithm used here was the support vector machine (SVM)-based Yamcha (Yet Another Multipurpose CHunk Annotator) instead of an exponential model. Habash and Rambow showed that the use of a morphological analyzer is beneficial in POS tagging, and their results are the best published to date for tokenization of naturally occurring input and POS tagging of MSA (98% on the Penn Arabic treebank).

Habash, Rambow, and Kiraz[7] present a morphological analyzer and generator for the Arabic language family. Their work is novel in that it explicitly addresses the need for processing the morphology of the dialects. Their method provides an analysis to a root+pattern template representation, separate phonological and orthographic representations, and combination of morphemes from different dialects. They add two general requirements for morphological analyzer. First, the morphological analyzer must be accompanied by a morphological generator. The second requirement is that a representation should be used that is defined in terms of a lexeme and attribute-value pairs for morphological features such as aspect or person. In their approach, there are three levels of representation; lexeme level, morpheme level, and surface level in which words are a string of characters. They implement a multiple automaton of five tiers, a multi-tape

finite state automaton and represent the word using a context-free grammar. The use of a finite state technology makes their technique usable as a generator and an analyzer.

Diab, Hacioglu, and Jurafsky present,[8] an SVM based approach with a polynomial kernel of degree 2, to automatically tokenize, POS tag, and annotate base phrases in Arabic text. They adapt highly accurate tools developed for English text and apply them to Arabic. In the process of word tokenization, each letter in a word was tagged with a label indicating the morphological identity of the token to which it belonged. By token, one means stem+affix, proclitic, or enclitic. The POS tagging was a 1-of-24 classification task from the collapsed tag set in the ATB. In base phrase chunking 9 types of chunked phrases are recognized using a phrase IOB (Inside a phrase, Outside a phrase, and Beginning of a phrase) tagging scheme with an I and a B for each chunk and a single O tag. They report that the SVM-TOK tokenizer achieves an F score of 99.12, the SVM-Part of Speech tagger achieves an accuracy of 95.49%, and the SVM-Base Phrase chunker yields an F score (a measure of accuracy that takes into account both precision and recall) of 92.08.

Adler and Elhadad deal with morphological disambiguation of the Hebrew language by combining morphemes into a word in both agglutinative and fusional ways[9]. They present an unsupervised stochastic model. They use a morphological analyzer that deals with the data sparseness problem caused by the affixational morphology nature of Hebrew. They present a text encoding method for languages

of the affixational morphology nature in which the knowledge of word formation rules helps in the disambiguation. They use a Hidden Markov Model with the Baum-Welch (EM) algorithm in a way that the segmentation and tagging can be learned simultaneously. They use backoff smoothing, suggested by Thede and Harper (1999), with an extension of additive smoothing for the lexical probabilities. Reported results on a large scale corpus (6M words) were 92.32% for POS tagging, and 88.5% for full morphological disambiguation.

Brants' Trigrams 'n' Tags (TnT) [10] was proposed to be an efficient statistical part-of-speech tagger. Brants argues that a tagger based on Markov models performs at least as well as other approaches of the time, including the Maximum Entropy framework. Described in this paper are the basic model of TnT, the techniques used for smoothing and for handling unknown words, and evaluations on two corpora. When Brants' paper was published, a recent independent comparison of 7 taggers [15] had shown that Markov models combined with a good smoothing technique and with handling of unknown words works better than the leading Maximum Entropy framework which had a very strong position among the statistical approaches. Brants shows that this method yields the highest accuracy as well as being the fastest on training and testing. The most effective smoothing paradigm used in [10] is the linear interpolation of unigrams, bigrams, and trigrams with lambda_1, lambda_2, and lambda_3 as weights respectively. Handling the unknown words was done by suffix analysis; tag probabilities are set according to the word's ending. The probability distribution for a particular suffix is gener-

ated from all words in the training set that share the same suffix of a predefined maximum length. Probabilities were smoothed by successive abstraction using a sequence of increasingly more general context that recursively eliminates characters from the suffix. Additionally, different parameters were used for capitalized words than non-capitalized words in order to enhance the accuracy. Also, Brants reduced the processing time of the Viterbi algorithm by using a beam search which excluded a state of a smaller value from further processing. According to Brants in [10] a tagger based on Markov models yields great results and that the only other method that has comparable results is the Maximum Entropy framework. Brants' method yielded a tagging accuracy of 96-97% on the Penn Arabic Treebank.

2

Linguistic Resources

2.1 Corpora

The corpora used in this project were the MSA treebank (ATB) from the Linguistic Data Consortium (LDC) [14], consisting of 625,000 words (750,000 tokens after tokenization) of newspaper and newswire text, and the Levantine Arabic Treebank (LATB), also from the LDC. ATB is morphologically disambiguated by hand and syntactically annotated with reduced and elaborate tag sets. The corpus is split into 10% development data, 80% training data, and 10% test data. The training data consists of 17,617 sentences and 588,244 tokens.

The Levantine treebank consists of 33,000 words of transcribed telephone conversations. This data was collected as part of the LDC CALL HOME project [5]. The speech effects have been removed for more text-like data. As mentioned before the LATB is only used for development and testing, not for training. The

development data consists of 1928 sentences and 11,151 tokens. The test data consists of 2051 sentences and 10,644 tokens.

All the available corpora are transliterated into ASCII characters using the Buckwalter transliteration scheme [13]. For all the experiments in this project I use the collapsed or the reduced POS tag set that focuses on major parts of speech, excluding morphological information, number, and gender, and both the vocalized and non-vocalized versions of the treebanks. I ran all of my experiments on the unvocalized versions because the vocalized versions were hand-voweled and would detract from my goal of minimizing human supervision. All the data is derived from the parsed trees in the treebank. The MSA training data set that was mainly used in my experiments was untokenized, unvocalized, and contained 660,307 words of which 41,659 were unique, 24 unique tags, and 18,970 sentences. The unvocalized and untokenized Levantine development data set contained 15,249 words of which 2091 were unique, 21 unique tags, and 1928 sentences. The average sentence length of the MSA corpus is 33 words while that of the Levantine corpus is 6, due to the informal register of the Levantine corpus. A register is a subset of a language that is used in a specific setting and for a given purpose. The register distinguishes amongst variations in a language with respect to the user and takes into account social background, geography, sex, and age. Even though MSA and LA have a considerable number of words in common, they are quite different from one another. Table 2.1 shows the lexical overlap between the two languages by type, or the number of unique words, and token which is the total number of

words. One can also infer the difference between the frequency distributions of the tag between the two languages.

In the ATB, some words have been given no morphological analysis and have been assigned the POS tag of NO_FUNC. The Arabic treebank has some inconsistencies in addition to the NO_FUNC tagged words. There are more unique tags in the vocalized version (28) than the unvocalized (26). I also noticed that the preposition '-hum', the third person masculine plural personal pronoun, has two tags in the vocalized version, PRP, the reduced version, and CVSUFF_3MP which carries more detailed information such as third person masculine plural. Additionally, there are a couple of tags, VERB, and VB that are comprised of a mixture of past tense, imperative, and present tense verbs, even though there already exist the tags VBD (past), VBP (present) and VBN (past participle). Additionally, most of these verbs tagged as VB and VERB are not vocalized in the vocalized version and some are even misspelled, although most of these misspellings are as a result of a common ommission of the letters *hamza*, *shadda*, *fathatan*, *dammatan*, and *kasratan*.

Table 2.1: Lexical Overlap Between LA Development Set and ATB

	By Type			By Token			
	% in MSA Only	% in Both	% in LA Only	% in MSA Only	% in Both	% in LA Only	Frequency
NN	93.69	5.3	1.01	78.32	21.59	0.09	163586
IN	62.71	31.64	5.65	16.72	83.27	0.01	100738
JJ	95.21	3.94	0.85	84.73	15.14	0.12	58015
NNP	97.56	2.25	0.19	92.07	7.89	0.03	43724
PUNC	90.91	9.09	0	99.4	0.6	0	43105
CC	45.45	54.55	0	2.46	97.54	0	40043
VBD	94.01	4.38	1.6	78.51	21.26	0.23	24313
VBP	90.47	3.16	6.37	83.45	15.08	1.47	21403
NNS	97.78	1.72	0.5	91.39	8.53	0.08	19494
PRP	54.72	26.42	18.87	4.3	95.63	0.07	15127
PRP$	23.08	69.23	7.69	1.66	98.33	0.01	12992
CD	98.26	1.52	0.22	91.94	8.04	0.03	11291
RP	37.31	44.78	17.91	18.66	81.19	0.14	8399
WP	63.33	33.33	3.33	6.89	93.1	0.01	8165
DT	60.71	21.43	17.86	25.31	74.59	0.1	4956
RB	44.29	34.29	21.43	56.98	42.53	0.48	3221
NOFUNC	96.14	2.87	0.99	95.85	3.42	0.72	2629
VBN	95.39	4.48	0.13	93.31	6.65	0.04	2797
WRB	50	25	25	20.1	79.08	0.82	611
UH	48	44	8	28.44	70.62	0.95	211
VB	56.82	13.64	29.55	62.5	14.29	23.21	74
NNPS	100	0	0	100	0	0	31
VERB	88.89	11.11	0	88.89	11.11	0	9
NUMCOM	100	0	0	100	0	0	2

3

Part-of-Speech Tagging

3.1 Methodology

3.1.1 Basic Model

In the original experiments I used a hidden Markov Model (HMM) to represent the data. Let w_i represent the i_{th} word in a sequence of words and let t_i represent the i_{th} tag. We can compute the probability of seeing a sequence of tags, $T = t_1, t_2, \cdots, t_n$ as:

$$P(T) = P(t_1)P(t_2|t_1)P(t_3|t_2, t_1)P(t_n|t_{n-1}, \cdots, t_1)$$

In order to decrease the complexity of this model we can assume that each tag only depends on the immediately preceding tag (also called the Markov property), which gives us:

$$P(T) = P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_n|t_{n-1})$$

This is called a first order Markov model which can also be called a Bigram model when used to represent sequences of words or letters. Note that in our case, we are presented with a sequence of words and asked to determine the most likely sequence of tags. Thus we can consider the tags to be hidden states and the words to be observations. If we assume that the i_{th} word depends on the i_{th} tag alone, using Bayes rule we can compute the probability of a given tag in the i_{th} position as:

$$P(t_i|w_i, t_{i-1}) = \frac{P(w_i|t_i)P(t_i|t_{i-1})}{P(w_i|t_{i-1})}$$

This gives us what is called the Hidden Markov Model. Since for POS tagging we are only concerned about $\arg\max_{t_i} P(t_i|w_i, t_{i-1})$ and $P(w_i|t_{i-1})$ is not a function of t_i we can see that we need to estimate three parameters for our model. The first parameter is $P(w_i|t_i)$, also known as the emission probability. The second parameter is $P(t_i|t_{i-1})$, or the transition probability. Finally, we need to estimate the vector consisting of $p(t)$ for all t, which is called the π vector. The π vector is a distribution on initial states for the tags. Combining all the parameters in our model we get:

$$P(\vec{t}|\vec{w}) = \frac{P(w_1|t_1)P(t_1)\prod_{i=2}^{n} P(w_i|t_i)P(t_i|t_{i-1})}{P(\vec{w})}$$

When given a sequence of words $W = w_1, w_2, \cdots, w_n$, or a sentence, our goal is to find the sequence of tags $T = t_1, t_2, \cdots, t_n$ which maximizes $P(W|T)$. One efficient way of doing this is to use the Viterbi algorithm[1]. In the Viterbi algorithm, $P(t_i|w_1, w_2, \cdots w_n)$ is computed from the transition and emission matrices using Bayes rule for every word in the sequence. The optimal path for each state as well as state probabilities are stored for each word, with the most probable state and path being chosen at the end.

Basic Approach

The simplest way to compute the parameters for our HMM is to use relative frequency estimation, which is to count the frequencies of word/tag and tag/tag combinations as well as the frequencies of tags at the beginning of each sentence. The problem with this approach is that there are many words seen in the test set that are absent in the training set. In these situations, $P(w_i|t_i)$ is zero for all tags which means that there can be no paths of tags corresponding to a sequence of words containing w_i. Note that this method, and all subsequent methods, are supervised because the tagging in the training set is done by humans.

One simple way to avoid this problem is to find all the columns in the emission matrix which contain all zeros, replace them with a constant value $\alpha = 10^{-20}$. This is similar to add-alpha smoothing, except that in my method only the elements in the all-zero columns are affected, whereas in add-alpha smoothing, α is added to every single entry. Note that this method does not yield a true

probability distribution. Experimentally both methods yielded nearly identical results. We will consider the accuracy of this method to be a baseline. For the MSA development set, this was 93.55% and for the LA development set this was 69.21%.

Another possible improvement that I tried was implementing a trigram model for the tags instead of a bigram. However, due to the sparseness of the resulting transition matrix for $P(t_i|t_{i-1}, t_{i-2})$, this method did not work as well as expected, even with add-alpha smoothing.

Using Morphological Properties

In order to deal with the sparsity of the emission model, I tried forming equivalence classes over words based on morphological properties. The circumfixes from each word were extracted (more on how this was done later in this section), and equivalence classes were formed for words sharing the same circumfix. For words that did not appear in the training corpus, the average value of $P(w|t)$ for all words sharing the same circumfix in the training corpus was computed using relative frequency estimation and used to fill the empty (all-zero) columns of the emission matrix. For words that did not share a circumfix with any word in the training corpus, the average value of $P(w|t)$ for the entire corpus was used instead.

In order to test the potential effectiveness of this method I first relied on the circumfixes given in the Treebank (the tokenized version). Using the above described method with these circumfixes raised the accuracy of the tagger when

tested on MSA. However, these circumfixes were found by humans and are not available for use in the untokenized training data.

In order to replicate these results with the untokenized training data, I tried finding the equivalence classes by looking at the first n letters and the last k letters of the words in the untokenized MSA corpus, however, the accuracy was lower than the baseline accuracy found with the basic method described earlier for all combinations of n and k when cross validating on MSA. However, I was also able to obtain an improvement in accuracy on MSA by finding circumfixes using a method similar to that of Schone and Jurafsky[2]. This method was applied to English, German, and Dutch and has never been used on Semitic languages. Specifically, I induce the circumfixes in five steps:

1) Reverse the letters for all the words in the corpus and add them to a word trie. Branch points are found and are assumed to be prefixes (with no maximum length.) These tentative prefixes are extracted and stored.

2) I make another pass through the word database, this time searching our prefix database for all possible prefixes for each word and adding the remaining stems (words with prefixes removed), as well as the original words, to a new word trie. The words and stems are added without reversing their letters. I again find branch points, with the branch points assumed to be suffixes. I then extract the suffixes and store them.

3) Step 2 is repeated, but I reverse the letters in the stems and words before adding them to the new word trie in order to find prefixes.

4) At this point I have a set of pseudo-prefixes (found in 3) and pseudo-suffixes (found in 2). I pass through the word database and find all possible pseudo-prefix and pseudo-suffix pairs that can fit a given word. These are stored as candidate circumfixes.

5) All candidate circumfixes are ranked based on the number of stems to which they appear attached. Stems that have only one circumfix associated with them are not counted. All words are then tokenized based on the highest scoring circumfix to which they can be attached. For example, *YakotubAni* meaning 'they write (dual masculine)' becomes *Ya + kotub + Ani*.

Using Expectation Maximization

Another possible way to improve results from the basic model is to use Expectation Maximization (EM) in order to fine tune the parameters (transition matrix, emission matrix, and π vector) for the test set[3]. The basic concept of EM is to iteratively increase the likelihood of an input sequence given a set of initial parameters. The EM process consists of two steps and assumes that the data is divided into two parts, the observed data and the unobserved data. The combination of the observed and unobserved data is called the complete data. In the expectation step, the expected value of the the complete data given the observed data and the current set of parameters is computed. In the maximization step, parameters are chosen that maximize the likelihood of the complete data.

An efficient way to compute EM for hidden Markov models is to use the

Baum-Welch algorithm[4]. In the Baum-Welch algorithm, $P(t_i = T|w_0 \cdots w_N)$ where t_i is the i_{th} tag in the sequence of N words, is computed for every possible tag T using the forward-backward algorithm. Using the forward-backward algorithm we can also compute $P(t_{i-1} = T_1, t_i = T_2|w_0 \cdots w_N)$ where T_1 and T_2 can be any combination of tags and t_i and w_i are defined as before. Based on these probabilities, improved estimates for the emission matrix, transition matrix, and π vector values are computed as follows:

$$P(T) \approx \frac{\sum_{i=1}^{N} P(t_i = T|w_0 \cdots w_N)}{N} \tag{3.1}$$

$$P(T_1|T_2) \approx \frac{\sum_{i=1}^{N} P(t_{i-1} = T_1, t_i = T_2|w_0 \cdots w_N)}{N \cdot P(T_1)} \tag{3.2}$$

$$P(W|T) \approx \frac{\sum_{i \in \{i:\, W=w_i\}} P(t_i = T|w_0 \cdots w_N)}{N \cdot P(T)} \tag{3.3}$$

This process can be repeated until it converges on a local maximum, as shown in [11]. The difference between the expectation and maximization steps may not be apparent at first sight, however, Lloyd Welch's paper gives a thorough explanation of how the Baum-Welch algorithm falls into the category of EM algorithms [11].

In order to initialize the process I first computed the emission matrix, transition matrix, and π vector values using relative frequency estimation on the training set smoothed with the alpha parameter, as described earlier. I then experimented with iteratively updating the emission and transition probabilities using Baum-Welch as described above on the test set. I first updated only the emission prob-

ability, then the transition probability, and lastly both at once. The best results were obtained by altering both of the parameters, but for only one iteration (see Figure 3.3).

One helpful method that I tried was inserting a dummy tag at the end of every sequence. This allowed the algorithm to have additional knowledge of the structure of the corpus by inserting "end of sentence" tags. For example, certain tags may have different probabilities of occurrence when near sentence boundaries. This led to a slight increase in accuracy. As a result, I used this technique in all of my Levantine experiments.

3.1.2 Two-Layer Markov Model

Since when testing on the LA development set, 65% of errors occur in words that are rare or nonexistent in the training set (occurring fewer than four times), it made sense to focus my efforts on the emission probabilities. This idea is supported by measuring the cross entropy between the various data sets we have at our disposal. The cross entropy between the transition matrices for the MSA training and the MSA development sets is 2.239 nats, whereas the cross entropy between the transition matrices for the MSA training set and the LA development set, which is the same size as the MSA development set, is 2.740 nats. This relatively small difference in cross entropy between the transition matrices for the MSA training and LA development sets leads us to believe that our assumption that the inaccuracy in computing the emission probabilities is the most significant

factor in tagging error is valid.

Another way to utilize the morphological properties of the words in addition to using equivalence classes based on affixes is to model each word as a Markov Model with the characters representing states. We can see this illustrated in Figure 3.1. In training we must compute transition probabilities and π vectors for the letters in the words corresponding to each tag in addition to transition probabilities and π vectors for the tags. In testing, I again used the Viterbi algorithm to find the most likely sequence.

Experimentally it turned out that the best results were obtained by using a combination of the basic model and the two-layer Markov model. For words that existed in the training set, the basic model was used. For words that did not exist in the training set, the emission probability calculated from the 2-layer Markov model was used, effectively filling in the all-zero columns of the emission matrix with better informed numbers than simply alpha. Because in computing the most likely tag sequence with the Viterbi algorithm, we are only concerned with $\arg\max_{t_i} p(w_i|t_i)$, there is no need to do any normalization between the columns.

In my original experiment with the two layer Markov model, I used a bi-gram model to represent the letters in each word and I used relative frequency estimation to compute the parameters. However, because of the limits of a bigram model, I implemented K-gram models with K ranging from 1 to 10. In order to avoid problems with sparseness, I implemented Kneser-Ney smoothing for the letter model as described in Chen and Goodman's paper[12]. Kneser-Ney smoothing

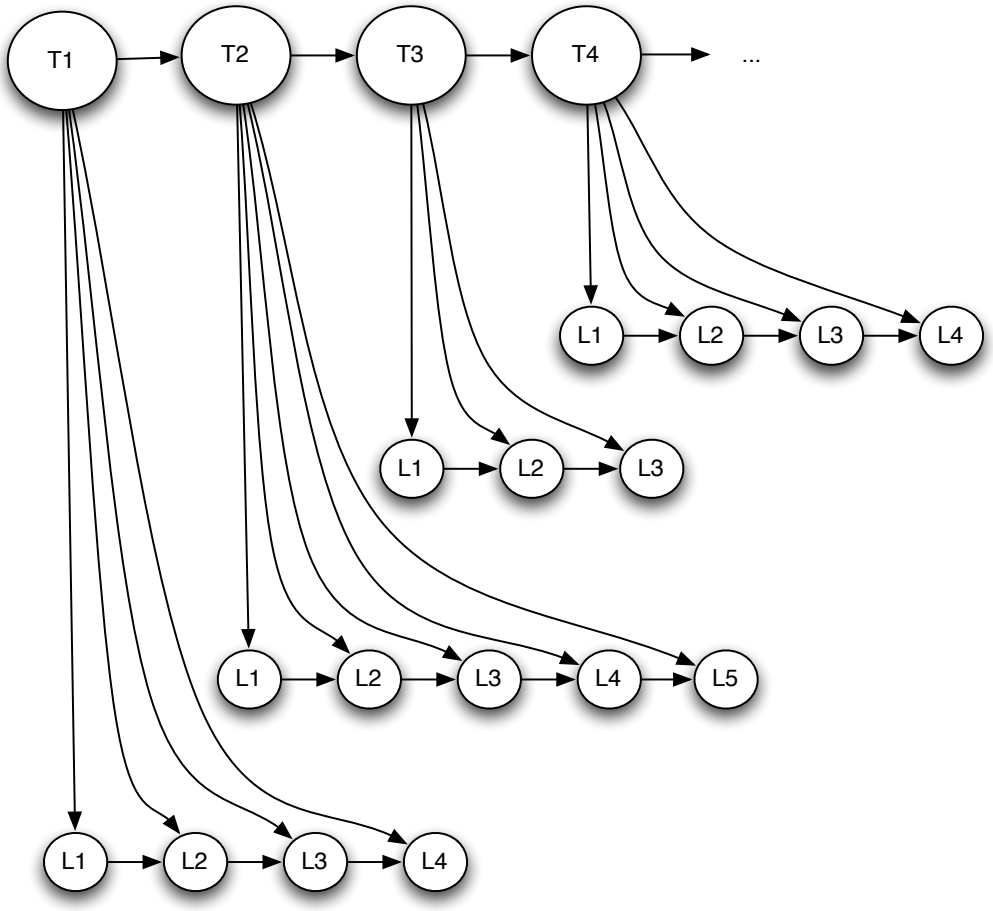


Figure 3.1: Diagram of 2-layer Markov Model

is one of many smoothing methods based on weighted interpolation between a higher and lower order distribution.

Two-Layer MM with EM

Because my best results were obtained by combining the two layer model and the basic model, the easiest way to apply EM and still maintain the benefit of the two-layer Markov model is to use EM to estimate the transition matrix,

emission matrix, and π vector values using Baum-Welch as described earlier for the basic model, and simply substitute the emission probabilities from the 2-layer model only for words that did not exist in the training set.

A more difficult method of using EM is to modify Baum-Welch in order to learn the parameters for the 2-layer model. In order to use EM to learn the transition probabilities for the MMs representing the words, we need to replace $P(w|t)$ with $P(l_1|t)P(l_2|l_1,t)\cdots P(l_n|l_{n-1},\cdots,l_1,t)$ in the forward-backward algorithm stage. Using this, we can again obtain $P(t_i = T|w_0\cdots w_N)$ and $P(t_{i-1} = T_1, t_i = T_2|w_0\cdots w_N)$. We can compute $P(T)$ and $P(T_1|T_2)$ exactly as before. Let M be the total number of letters. Then we can compute the parameters $P(L|T)$, the probability of seeing the letter L, and $P(L_1, L_2|T)$, the probability of seeing the consecutive letters L_1 and L_2 as follows:

$$P(L|T) \approx \frac{\sum_{i\in\{i:\ L\in w_i\}} P(t_i = T|w_0\cdots w_N)}{M\cdot P(T)} \tag{3.4}$$

$$P(L_1, L_2|T) \approx \frac{\sum_{i\in\{i:\ L_1,L_2\in w_i\}} P(t_i = T|w_0\cdots w_N)}{(M-N)\cdot P(T)} \tag{3.5}$$

Thus we have updated both transition probabilities and π vectors for the letters in the words corresponding to each tag and the transition probabilities and π vectors for the tags. Note that in this case, $P(L|T)$ is analogous to the π vector in the simple HMM and $P(L_1, L_2|T)$ is analogous to the transition probability in the simple HMM.

When considering using this model with Kneser-Ney, we must be aware of

certain issues that may arise. In Chen and Goodman's paper on smoothing[12], $p_{KN}(w_i|w_{i-n+2}^{i-1})$, the Kneser-Ney probability of a given n-gram, is defined as

$$p_{KN}(w_i|w_{i-n+2}^{i-1}) = \frac{N_{1+}(\bullet w_{i-n+2}^{i})}{N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet)}$$

where

$$\begin{aligned} N_{1+}(\bullet w_{i-n+2}^{i}) &= |\{w_{i-n+1} : c(w_{i-n+1}^{i}) > 0\}| \\ N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet) &= |\{(w_{i-n+1}, w_i) : c(w_{i-n+1}^{i}) > 0\}| = \sum_{w_i} N_{1+}(\bullet w_{i-n+2}^{i}) \end{aligned}$$

Note that N_{1+} represents the number of words that have one or more counts, and $\bullet$ represents a free variable that is summed over. For example:

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i-1}w_i) > 0\}|$$

where $c(w_{i-n+1}^{i-1}w_i)$ represents the number of times $w_{i-n+1}^{i-1}w_i$ occurs in the corpus. From this we can see that we may have a problem adapting this for use with EM. Since when we iterate in EM we have probabilities instead of counts, we are forced to replace expressions like $c(w_i)$ with $\sum_{w_i} P(t|w_i)$. It is then not clear how one must deal with instances in which one must evaluate expressions like:

$$c(w_{i-n+1}^{i}) > 0$$

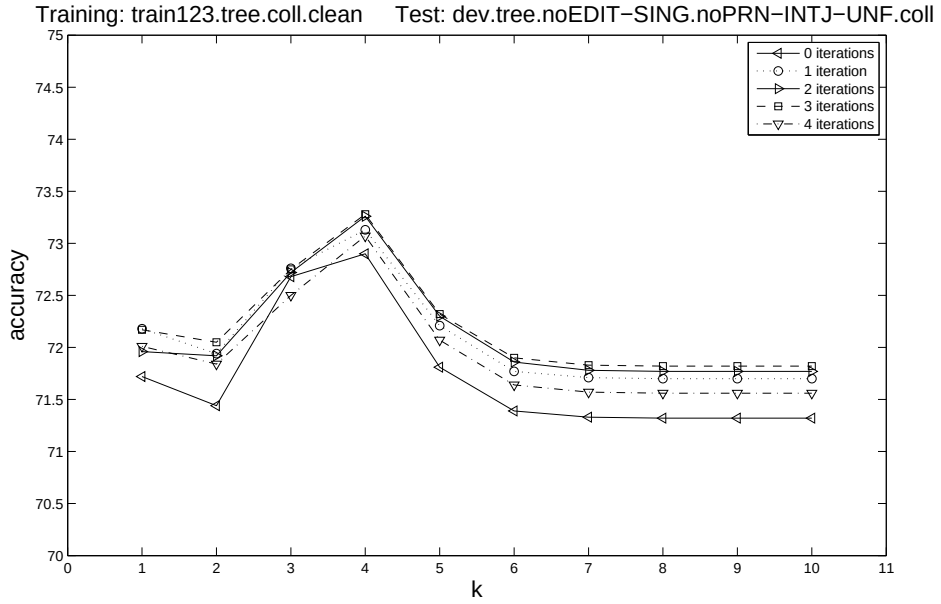
or the even more ambiguous instances in which one must evaluate expressions like:

$$c(w_{i-n+1}^{i}) = 1$$

which occur frequently when calculating probabilities using Kneser-Ney. This results in very poor performance when adapting Kneser-Ney for use with EM in the 2-layer Markov Model, as was seen in my experiments in which I ended up assigning different ranges of $\sum_{w_i} P(t|w_i)$ to integer values of $c(w_i)$. Because of this, my best results were obtained by using the simple substitution method described earlier.

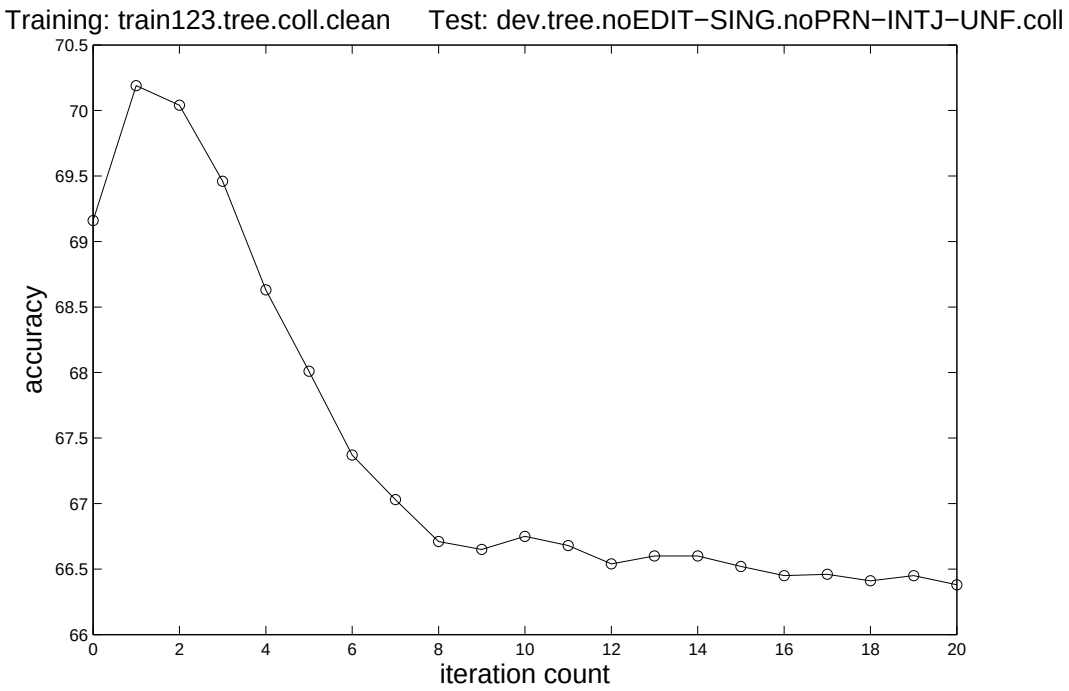
3.2 Results and Discussion

We have adapted a Part of Speech tagger that was trained on Modern Standard Arabic sentences in order to tag Levantine sentences. For a side by side comparison of the accuracies obtained with various methods used on the LA development set, please refer to table 3.7. My experiments show that the best results on the development set were achieved by the two-layer Markov / basic Markov mixture model with simple EM and Kneser-Ney smoothing, with a peak accuracy of 73.28% with three iterations and $k = 4$, where k is the length of the K-grams used in Kneser-Ney smoothing (please see Figure 3.2). This method involved initializing with the ATB and using EM on the LA treebank. As we can see in table 3.1, most LA words are four letters in length, which makes sense given that $k = 4$ performs the best. This accuracy is slightly higher than the 73% accuracy achieved



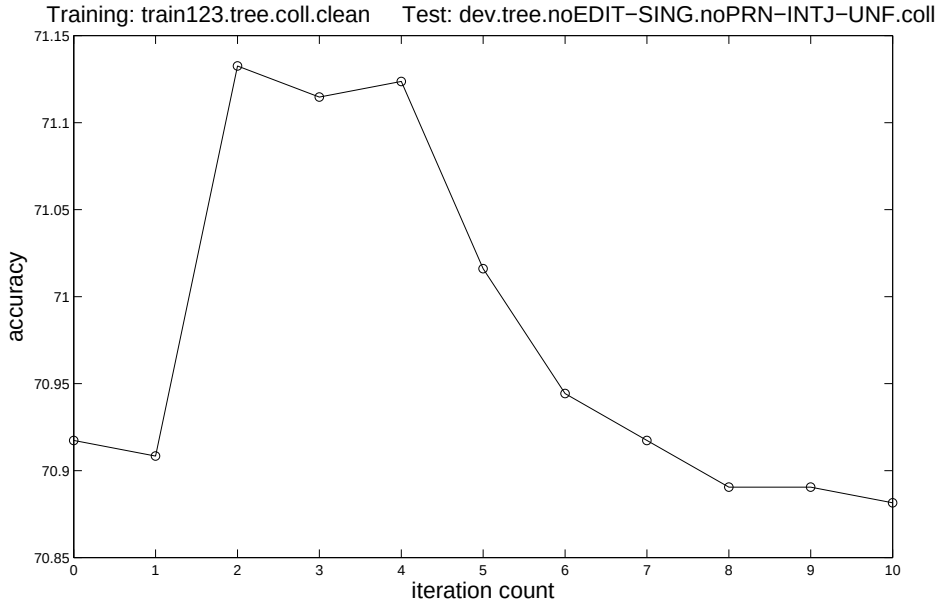
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
0 iterations	71.72	71.44	72.68	72.9	71.81	71.39	71.33	71.32	71.32	71.32
1 iteration	72.18	71.94	72.76	73.13	72.21	71.77	71.71	71.7	71.7	71.7
2 iterations	71.96	71.92	72.72	73.26	72.3	71.86	71.78	71.77	71.77	71.77
3 iterations	72.18	72.05	72.75	73.28	72.32	71.9	71.83	71.82	71.82	71.82
4 iterations	72.01	71.84	72.5	73.07	72.07	71.64	71.57	71.56	71.56	71.56

Figure 3.2: Accuracy of 2-Layer Markov/Basic Markov Mixture Model Versus k



n_iterations	0	1	2	3	4	5	6	7	8	9
% accuracy	69.16	70.19	70.04	69.46	68.63	68.01	67.37	67.03	66.71	66.65
n_iterations	10	11	12	13	14	15	16	17	18	19
% accuracy	66.75	66.68	66.54	66.60	66.60	66.52	66.45	66.46	66.41	66.45

Figure 3.3: Accuracy of Basic HMM with EM on Emission and Transition Versus EM Iteration Count



n_iterations	0	1	2	3	4	5	6	7	8	9	10
% accuracy	70.92	70.91	71.13	71.11	71.12	71.02	70.94	70.92	70.89	70.89	70.88

Figure 3.4: Accuracy of 2-Layer Markov Model Versus EM Iteration Count

Table 3.1: Frequency of Words in LA Versus Length

Length	% in LA Corpus
0	0.67
1	3.96
2	12.96
3	26.59
4	28.41
5	13.87
6	9.90
7	3.16
8	0.72
9	2.89
10	0.05

by Rambow et all in "Parsing Arabic Dialects" [5] using basic linguistic knowledge such as category-dependent probability mass redistribution from the MSA-only words to the LA-only words according to the unigram probability distribution of the part-of-speech, the distribution over the lengths of words for each POS tag, and the first and last few characters of each word.

The other methods that I tried performed as follows. As a baseline, I consider the performance of a bigram tagger whose parameter values are estimated from annotated MSA data. I divided the MSA training dataset (the vocalized version) into 10% test and 90% training. I then ran a 10-fold cross-validation test on the remaining 90% and chose the parameters (transition matrix, emission matrix, and π vector) with the best accuracy of the ten tests (92.95%) to run the algorithm on the 10% of the data reserved for testing purposes. This resulted in an accuracy of 91.54%. When tested on the development set of MSA sentences, the tagger

performs reasonably well, with a 93.55% accuracy. Note that the morphological approach resulted in a 1.7% improvement in accuracy when using the circumfixes found in the tokenized treebank, and a 0.13% improvement in accuracy when using the circumfixes induced from the corpus using the trie method described by Schone and Jurafsky[2].

When trained on the MSA training set and tested on the LA development set, the basic method produced an accuracy of 69.21%. However, with the addition of EM, this method produced slightly different accuracies depending on which parameters EM was allowed to modify. When EM was allowed to modify the emission matrix only, the resulting accuracy was 69.60%. For transition only, this accuracy was 69.66%. When EM was allowed to modify both, the resulting accuracy was 70.19%. Refer to Figure 3.3 for more details.

When using a trigram relative frequency estimation and add-alpha smoothing to represent the tags in the basic model, the accuracy plummeted to 51.01% when tested on the LA development set. This is most likely because of the sparseness of the resulting transition matrix.

The results for the modified 2-Layer MM version of Baum-Welch were promising, but not superior to some of the previous methods. When using this method with a simple bigram model for the letters in each word, I obtained an accuracy of 71.13%. This is better than the baseline accuracy for this model with no EM, which was 70.92% (please see Figure 3.4). However, it appears that using Kneser-Ney smoothing and higher order k-grams when representing letters in

words caused a more significant improvement than EM did, and as a result, the modified Baum-Welch method does not perform as well.

The 2-layer Markov Model in which Kneser-Ney smoothing was used for n-grams in the words and Baum-Welch was used to learn all the parameters, did not work as well as I had hoped because of the reasons explained in the Methodology section. After one iteration, the accuracy plummeted to 70.65%, after which it eventually converged to 70.38%.

Tables 3.2 and 3.3 are the confusion matrices corresponding to the best results obtained using both methods.

Close observation of the confusion matrices shows that major confusions occur in the following areas in descending order:

- Singular proper nouns (NNP) are being guessed as singular common nouns (NN), punctuations (PUNC), perfect verbs (VBD), coordinating conjunctions (CC), and adjectives (JJ).
- Plural common nouns (NNS) are being confused with singular common nouns, IN, and adjectives.
- Adjectives are primarily guessed as singular common nouns, singular proper nouns, VBD, IN, and CC.
- Cardinal numbers (CD) are being confused with NN, JJ, and NNS.
- Singular common nouns are guessed as JJ, IN, NNP, CC, and perfect verbs.

- Perfect verbs (VBD) are guessed mostly as NN, CC, VBP, and INs.
- INs are confused with NNs, adverbs (RB), and particles (RP).
- Personal pronouns (PRP) are being confused with CC, IN, NNP, possessive personal pronouns (PRP$), and NN.
- PRP$ tags are being tagged as PRP and IN.

One can notice that the major improvement in accuracy between the two methods is on present verbs, common nouns, and adjectives, while there was a decline for the particles. This is not surprising due to the morphological analysis of the words in the second method, since verbs of present tense are prefixed with a present-tense indicator in LA which is absent in MSA. Similarly, for the most part, nouns and adjectives have different vowels in MSA and LA while sharing the same root. Particles on the other hand are known to be fundamentally different in the two languages. Perhaps the sub-structure analysis of the particles has created some confusion with words of other types.

3.3 Conclusion

In this paper my goal was to develop a part-of-speech tagger for a dialect of Modern Standard Arabic (MSA) called Levantine, using MSA as the training data and testing on a Levantine corpus. The need for this stems from the lack of enough annotated data in Levantine. The task proved to be nontrivial due to

Table 3.2: Confusion Matrix (by count) for Basic Model with EM (1 iteration) (first column represents the gold tags)

	NNP	PUNC	NNS	JJ	CD	NN	VBD	IN	PRP	PRP$	VBP	CC	WP	RP	NOFUNC	VBN	RB	WRB	DT	UH	VB	VERB
NNP	259	15	0	6	0	35	8	1	0	0	1	4	0	0	3	0	0	0	0	0	0	0
PUNC	0	464	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NNS	0	2	71	2	1	14	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJ	50	14	4	556	0	138	37	22	0	0	3	9	3	1	0	0	0	0	0	7	0	0
CD	0	3	5	6	47	13	1	3	0	0	2	0	0	0	0	0	0	0	0	0	0	0
NN	70	24	3	127	2	2175	81	82	4	0	10	24	5	9	11	5	0	0	0	20	4	0
VBD	9	2	0	8	0	24	205	16	2	0	20	10	2	2	2	11	0	0	0	0	0	4
IN	8	6	0	4	0	111	1	1478	4	5	0	3	12	35	0	0	73	0	0	0	0	0
PRP	31	23	0	5	0	33	38	67	1043	42	2	48	0	2	0	0	0	0	0	0	0	0
PRP$	2	0	0	0	0	0	0	24	33	402	0	1	0	0	0	0	0	0	0	0	0	0
VBP	42	39	0	104	2	115	126	144	5	1	336	45	6	5	0	4	0	0	0	0	1	0
CC	1	0	0	0	0	0	0	0	0	0	0	424	0	0	0	0	0	0	0	0	0	0
WP	18	0	0	0	0	4	0	0	0	0	0	1	111	4	0	0	0	0	0	0	0	0
RP	120	5	0	5	0	153	16	16	8	0	0	12	114	98	119	0	0	0	0	1	0	0
NOFUNC	1	2	1	3	0	5	2	10	0	0	3	1	0	0	0	0	0	0	0	0	0	0
VBN	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
RB	38	10	0	1	0	107	22	35	0	0	5	43	3	1	0	0	45	0	0	0	0	0
WRB	0	0	0	0	0	4	0	5	0	0	0	1	0	9	0	0	0	2	0	0	0	0
DT	1	2	0	12	30	0	2	18	1	0	4	0	0	0	0	0	0	0	68	0	0	0
UH	0	0	0	2	0	6	6	0	0	2	0	1	0	0	0	0	0	0	0	39	0	0
VB	1	0	0	1	0	5	24	7	0	0	15	13	0	1	0	1	0	0	0	0	0	0
VERB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.3: Confusion Matrix (by count) for 2-Layer Markov/Basic Markov Mixture Model with EM (1 iteration, k = 4) (first column represents the gold tags)

	NNP	PUNC	NNS	JJ	CD	NN	VBD	IN	PRP	PRP$	VBP	CC	WP	RP	NOFUNC	VBN	RB	WRB	DT	UH	VB	VERB
NNP	254	0	3	6	0	50	9	0	0	0	3	0	0	0	7	0	0	0	0	0	0	0
PUNC	0	464	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NNS	3	0	80	2	0	6	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	0
JJ	54	0	14	597	0	133	18	1	0	0	9	0	0	0	7	0	0	0	0	9	0	0
CD	0	0	4	13	46	15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
NN	125	0	7	132	1	2285	22	4	2	0	8	2	1	7	37	3	0	0	0	20	0	0
VBD	13	0	3	11	0	45	209	1	0	0	11	0	0	2	6	11	1	0	0	0	0	4
IN	21	0	0	0	0	116	0	1467	4	5	0	1	30	13	6	0	77	0	0	0	0	0
PRP	9	1	0	0	0	152	2	57	1042	64	0	6	0	0	1	0	0	0	0	0	0	0
PRP$	2	0	0	0	0	0	0	29	25	405	0	1	0	0	0	0	0	0	0	0	0	0
VBP	116	0	1	23	0	183	64	17	3	0	530	0	0	0	31	6	1	0	0	0	0	0
CC	0	0	0	0	0	0	0	0	0	0	0	424	0	0	1	0	0	0	0	0	0	0
WP	20	0	0	1	0	2	0	0	0	0	0	0	114	0	0	0	0	0	0	1	0	0
RP	161	0	0	10	0	114	5	12	7	0	0	0	99	72	131	0	0	34	0	22	0	0
NOFUNC	4	0	2	5	0	11	3	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0
VBN	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	3	0	0	0	0	0	0
RB	39	0	1	10	0	136	5	20	0	0	3	0	0	21	15	0	58	0	0	2	0	0
WRB	6	0	0	0	0	3	0	0	0	0	0	0	0	3	1	0	0	8	0	0	0	0
DT	0	0	0	1	0	7	6	0	1	1	0	0	0	0	0	0	0	0	73	0	0	0
UH	1	0	0	1	0	7	6	0	1	1	0	0	0	0	0	0	0	0	0	39	0	0
VB	4	0	1	1	0	25	8	2	0	0	27	0	0	0	0	0	0	0	0	0	0	0
VERB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.4: Confusion Matrix (by percentage) for 2-Layer Markov/Basic Markov Mixture Model with EM (1 iteration, k = 4) (first column represents the gold tags)

	NNP	PUNC	NNS	JJ	CD	NN	VBD	IN	PRP	PRP$	VBP	CC	WP	RP	NOFUNC	VBN	RB	WRB	DT	UH	VB	VERB
NNP	78.0	4.5	0.0	1.8	0.0	10.5	2.4	0.3	0.0	0.0	0.3	1.2	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PUNC	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NNS	0.0	2.1	75.5	2.1	1.1	14.9	1.1	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
JJ	5.9	1.7	0.5	65.9	0.0	16.4	4.4	2.6	0.0	0.0	0.4	1.1	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0
CD	0.0	3.8	6.2	7.5	58.8	16.2	1.2	3.8	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NN	2.6	0.9	0.1	4.8	0.1	81.9	3.0	3.1	0.2	0.0	0.4	0.9	0.2	0.3	0.4	0.2	0.0	0.0	0.0	0.8	0.2	0.0
VBD	2.8	0.6	0.0	2.5	0.0	7.6	64.7	5.0	0.6	0.0	6.3	3.2	0.6	0.6	0.6	3.5	0.0	0.0	0.0	0.0	0.0	1.3
IN	0.5	0.3	0.0	0.2	0.0	6.4	0.1	84.9	0.2	0.3	0.0	0.2	0.7	2.0	0.0	0.0	4.2	0.0	0.0	0.0	0.0	0.0
PRP	2.3	1.7	0.0	0.4	0.0	2.5	2.8	5.0	78.2	3.1	0.1	3.6	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PRP$	0.4	0.0	0.0	0.0	0.0	0.0	0.0	5.2	7.1	87.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VBP	4.3	4.0	0.0	10.7	0.2	11.8	12.9	14.8	0.5	0.1	34.5	4.6	0.6	0.5	0.0	0.4	0.0	0.0	0.0	0.0	0.1	0.0
CC	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	99.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
WP	13.0	0.0	0.0	0.0	0.0	2.9	0.0	0.0	0.0	0.0	0.0	0.7	80.4	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RP	18.0	0.7	0.0	0.7	0.0	22.9	2.4	2.4	1.2	0.0	0.0	1.8	17.1	14.7	17.8	0.0	0.0	0.0	0.0	0.1	0.0	0.0
NOFUNC	3.6	7.1	3.6	10.7	0.0	17.9	7.1	35.7	0.0	0.0	10.7	3.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VBN	0.0	0.0	0.0	0.0	0.0	20.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.0	0.0	0.0	0.0	0.0	0.0	0.0
RB	12.3	3.2	0.0	0.3	0.0	34.5	7.1	11.3	0.0	0.0	1.6	13.9	1.0	0.3	0.0	0.0	14.5	0.0	0.0	0.0	0.0	0.0
WRB	0.0	0.0	0.0	0.0	0.0	19.0	0.0	23.8	0.0	0.0	0.0	4.8	0.0	42.9	0.0	0.0	0.0	9.5	0.0	0.0	0.0	0.0
DT	0.7	1.4	0.0	8.7	21.7	0.0	1.4	13.0	0.7	0.0	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	49.3	0.0	0.0	0.0
UH	0.0	0.0	0.0	3.6	0.0	10.7	10.7	0.0	0.0	3.6	0.0	1.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	69.6	0.0	0.0
VB	1.5	0.0	0.0	1.5	0.0	7.4	35.3	10.3	0.0	0.0	22.1	19.1	0.0	1.5	0.0	1.5	0.0	0.0	0.0	0.0	0.0	0.0
VERB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3.5: Confusion Matrix (by percentage) for 2-Layer Markov/Basic Markov Mixture Model with EM (1 iteration, k = 4) (first column represents the gold tags)

	NNP	PUNC	NNS	JJ	CD	NN	VBD	IN	PRP	PRP$	VBP	CC	WP	RP	NOFUNC	VBN	RB	WRB	DT	UH	VB	VERB
NNP	76.5	0.0	0.9	1.8	0.0	15.1	2.7	0.0	0.0	0.0	0.9	0.0	0.0	0.0	2.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PUNC	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NNS	3.2	0.0	85.1	2.1	0.0	6.4	0.0	0.0	0.0	0.0	2.1	0.0	0.0	0.0	1.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
JJ	6.4	0.0	1.7	70.9	0.0	15.8	2.1	0.1	0.0	0.0	1.1	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0	1.1	0.0	0.0
CD	0.0	0.0	5.1	16.5	58.2	19.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NN	4.7	0.0	0.3	5.0	0.0	86.0	0.8	0.2	0.1	0.0	0.3	0.1	0.0	0.3	1.4	0.1	0.0	0.0	0.0	0.8	0.0	0.0
VBD	4.1	0.0	0.9	3.5	0.0	14.2	65.9	0.3	0.0	0.0	3.5	0.0	0.0	0.6	1.9	3.5	0.3	0.0	0.0	0.0	0.0	1.3
IN	1.2	0.0	0.0	0.0	0.0	6.7	0.0	84.3	0.2	0.3	0.0	0.1	1.7	0.7	0.3	0.0	4.4	0.0	0.0	0.0	0.0	0.0
PRP	0.7	0.1	0.0	0.0	0.0	11.4	0.1	4.3	78.1	4.8	0.0	0.4	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PRP$	0.4	0.0	0.0	0.0	0.0	0.0	0.0	6.3	5.4	87.7	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VBP	11.9	0.0	0.1	2.4	0.0	18.8	6.6	1.7	0.3	0.0	54.4	0.0	0.0	0.0	3.2	0.6	0.1	0.0	0.0	0.0	0.0	0.0
CC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	99.8	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
WP	14.5	0.0	0.0	0.7	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	82.6	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0
RP	24.1	0.0	0.0	1.5	0.0	17.1	0.7	1.8	1.0	0.0	0.0	0.0	14.8	10.8	19.6	0.0	0.0	5.1	0.0	3.3	0.0	0.0
NOFUNC	14.3	0.0	7.1	17.9	0.0	39.3	10.7	0.0	0.0	0.0	3.6	0.0	0.0	0.0	7.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VBN	0.0	0.0	0.0	0.0	0.0	0.0	20.0	0.0	0.0	0.0	20.0	0.0	0.0	0.0	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0
RB	12.6	0.0	0.3	3.2	0.0	43.9	1.6	6.5	0.0	0.0	1.0	0.0	0.0	6.8	4.8	0.0	18.7	0.0	0.0	0.6	0.0	0.0
WRB	28.6	0.0	0.0	0.0	0.0	14.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	14.3	4.8	0.0	0.0	38.1	0.0	0.0	0.0	0.0
DT	0.0	0.0	0.0	1.1	0.0	7.9	6.7	0.0	1.1	1.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	82.0	0.0	0.0	0.0
UH	1.8	0.0	0.0	1.8	0.0	12.5	10.7	0.0	1.8	1.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	69.6	0.0	0.0
NNPS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NUMCOMMA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VB	5.9	0.0	1.5	1.5	0.0	36.8	11.8	2.9	0.0	0.0	39.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VERB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3.6: Description of Tagset Used in the Penn Arabic Treebank

Tag	**Description**
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PRP	Personal pronoun
PRP$	Possessive pronoun
PUNC	Punctuation
RB	Adverb
RP	Particle
UH	Interjection
VBD	Verb, past tense
VBN	Verb, past participle
VBP	Verb, present
WP	Wh-pronoun
WRB	Wh-adverb
NO_FUNC	
NUMERIC_COMMA	

Table 3.7: Best Accuracies Per Method

Method	**Accuracy**
Trigram with no EM	51.01%
Basic HMM with no EM	69.21%
Basic HMM with EM on Emission	69.60%
Basic HMM with EM on Transition	69.66%
Basic HMM with EM on Both	70.19%
HMM / 2-Layer Markov mixture with modified Baum-Welch and KN	70.65%
HMM / 2-Layer Markov mixture with modified Baum-Welch	70.92%
HMM / 2-Layer Markov mixture with no EM and KN	73.13%
HMM / 2-Layer Markov mixture with simple EM and KN	73.28%

lexical, grammatical, and inflectional differences as well as the difference in register between MSA and its dialect. The Arabic language is a very complicated template-based language with sophisticated and complex inflectional rules. The inflections often consist of adding interleaving vowels to the word roots and thus making the morphological analysis of the language a challenging task. I obtained my results by analyzing the morphological properties of Levantine via a novel idea of representing words as a sequence of characters emitted from a sequence of tags called the two-layer Markov Model shown in Figure 3.1. The best accuracy was achieved by using the 2-layer Markov / basic Markov mixture model with simple EM and Kneser-Ney smoothing described in the methodology section. Using this method I achieved an accuracy of 73.28% which is at least as good as that achieved by Rambow et al. in[5]. For future study, one could experiment with different methods of adapting Kneser-Ney smoothing to the 2-layer model. Because Kneser-Ney smoothing uses

discrete counts instead of probabilities, its adaptation for use with this model is non-trivial and there is a lot of room for possible fine-tuning. Additionally, if a model is devised that takes into account the Arabic root template morphology, it may be possibe to further improve accuracy.

Bibliography

[1] Viterbi, A.J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory,* 13(2), 260–269.

[2] Schone, P., & Jurafsky, D. (2000). Knowledge-Free Induction of Morphology Using Latent Semantic Analysis. In *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop* (pp. 67–72). Somerset, New Jersey: Association for Computational Linguistics.

[3] Dempster, A.P., Laird, N.M., & Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society,* 39, 1–38.

[4] Baum, L.E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occuring in the statistical analysis of probabilistic functions in Markov chains. *The Annals of Mathematical Statistics,* 41(1), 164–171.

[5] Rambow, O., Chiang, D., Diab, M., Habash, N., Hwa, R., Sim'an, K., Lacey, V., Levy, R., Nichols, C., & Shareef, S. (2005). Parsing Arabic Dialects. *Final Report, 2005 JHU Summer Workshop.*

[6] Habash, N., & Rambow, O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 573–580). Ann Arbor, MI: Association for Computational Linguistics.

[7] Habash, N., Rambow, O., & Kiraz, G. (2005). Morphological Analysis and Generation for Arabic Dialects. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages* (pp. 17–24). Ann Arbor, MI: Association for Computational Linguistics.

[8] Diab, M., Hacioglu, K., & Jurafsky, D. (2004). Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. In *Proceedings of HLT/NAACL.* Boston, MA

[9] Adler, M., & Elhadad, M. (2006). An unsupervised morpheme-based HMM for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL* (pp. 665–672). Sidney, Australia: Association for Computational Linguistics.

[10] T. Brants. (2000). TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference*. Seattle, WA.

[11] Welch, L.R. (2003). Hidden Markov Models and the Baum-Welch Algorithm In *IEEE Information Theory Society Newsletter* (pp. 1, 10–13).

[12] Chen, F. & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (pp. 310–318). Santa Cruz, CA.

[13] http://www.ldc.upenn.edu/myl/morph/buckwalter.html

[14] Mohamed Maamouri, Ann Bies, & Tim Buckwalter (2004). The penn arabic treebank: Building a large-scale annotated arabic corpus In *NEMLAR Conference on Arabic Language Resources and Tools*. Cairo, Egypt.

[15] Jakub Zavrel & Walter Daelemans (1999). Evaluatie van part-of-speech taggers voor het corpus gesproken nederlands. In *CGN technical report*. Katholieke Universiteit Brabant, Tillburg.