A Multilinear Approach to the Unsupervised Learning of Morphology

Anthony Meyer
Indiana University
antmeyer@indiana.edu

Markus Dickinson
Indiana University
md7@indiana.edu

Abstract

We present a novel approach to the unsupervised learning of morphology. In particular, we use a Multiple Cause Mixture Model (MCMM), a type of autoencoder network consisting of two node layers—hidden and surface—and a matrix of weights connecting hidden nodes to surface nodes. We show that an MCMM shares crucial graphical properties with autosegmental morphology. We argue on the basis of this graphical similarity that our approach is theoretically sound. Experiment results on Hebrew data show that this theoretical soundness bears out in practice.

1 Introduction

It is well-known that Semitic languages pose problems for the unsupervised learning of morphology (ULM). For example, Hebrew morphology exhibits both agglutinative and fusional processes, in addition to non-concatenative root-and-pattern morphology. This diversity in types of morphological processes presents unique challenges not only for unsupervised morphological learning, but for morphological theory in general. Many previous ULM approaches either handle the concatenative parts of the morpholgy (e.g., Goldsmith, 2001; Creutz and Lagus, 2007; Moon et al., 2009; Poon et al., 2009) or, less often, the non-concatenative parts (e.g., Botha and Blunsom, 2013; Elghamry, 2005). We present an approach to clustering morphologically related words that addresses both concatenative and non-concatenative morphology via the same learning mechanism, namely the Multiple Cause Mixture Model (MCMM) (Saund, 1993, 1994). This type of learning has direct connections to autosegmental theories of morphology (McCarthy, 1981), and at the same time raises questions about the meaning of morphological units (cf. Aronoff, 1994).

Consider the Hebrew verbs *zwkr*[1] ('he remembers') and *mzkir* ('he reminds'), which share the root *z.k.r*. In neither form does this root appear as a continuous string. Moreover, each form interrupts the root in a different way. Many ULM algorithms ignore non-concatenative processes, assuming word formation to be a linear process, or handle the non-concatenative processes separately from the concatenative ones (see survey in Hammarstrom and Borin, 2011). By separating the units of morphological structure from the surface string of phonemes (or characters), however, the distinction between non-concatenative and concatenative morphological processes vanishes.

We apply the Multiple Cause Mixture Model (MCMM) (Saund, 1993, 1994), a type of autoencoder that serves as a disjunctive clustering algorithm, to the problem of morphological learning. An MCMM is composed of a layer of hidden nodes and a layer of surface nodes. Like other generative models, it assumes that some subset of hidden nodes is responsible for generating each instance of observed data. Here, the surface nodes are features that represent the "surface" properties of words, and the hidden nodes represent units of morphological structure.

An MCMM is well-suited to learn non-concatenative morphology for the same reason that the autosegmental formalism is well-suited to representing it on paper (section 2): the layer of morphological structure is separate from the surface layer of features, and there are no dependencies between nodes within the same layer. This intra-layer independence allows each hidden node to associate with any subset of features, con-

[1] We follow the transliteration scheme of the Hebrew Treebank (Sima'an et al., 2001).

tiguous or discontiguous. We present details of the MCMM and its application to morphology in section 3. Our ultimate goal is to find a ULM framework that is theoretically plausible, with the present work being somewhat exploratory.

1.1 Targets of learning

Driven by an MCMM (section 3), our system clusters words according to similarities in form, thereby finding form-based atomic building blocks; these building blocks, however, are not necessarily morphemes in the conventional sense. A morpheme is traditionally defined as the coupling of a form and a meaning, with the meaning often being a set of one or more morphosyntactic features. Our system, by contrast, discovers building blocks that reside on a level between phonological form and morphosyntactic meaning, i.e., on the *morphomic* level (Aronoff, 1994).

Stump (2001) captures this distinction in his classification of morphological theories, distinguishing *incremental* and *realizational* theories. Incremental theories view morphosyntactic properties as intrinsic to morphological markers. Accordingly, a word's morphosyntactic content grows monotonically with the number of markers it acquires. By contrast, in realizational theories, certain sets of morphosyntactic properties *license* certain morphological markers; thus, the morphosyntactic properties cannot be inherently present in the markers. Stump (2001) presents considerable evidence for realizational morphology, e.g., the fact that "a given property may be expressed by more than one morphological marking in the same word" (p. 4).

Similarly, Aronoff (1994) observes that the mapping between phonological and morphosyntactic units is not always one-to-one. Often, one morphosyntactic unit maps to more than one phonological form, or vice versa. There are even many-to-many mappings. Aronoff cites the English past participle: depending on the verb, the past participle can by realized by the suffixes *-ed* or *-en*, by ablaut, and so on. And yet for any given verb lexeme, the *same* marker is used for the both the perfect tense and the passive voice, despite the lack of a relationship between these disparate syntactic categories. Aronoff argues that the complexity of these mappings between (morpho-)syntax and phonology necessitates an intermediate level, namely the morphomic level.

	MASC	FEM
SG	mqwmi	mqwmi-**t**
PL	mqwmi-im	mqwmi-w**t**

(a) *mqwmi* 'local'

	MASC	FEM
SG	gdwl	gdwl-h
PL	gdwl-im	gdwl-w**t**

(b) *gdwl* 'big'

Figure 1: The *t* quasi-morpheme

Our system's clusters correspond roughly to Aronoff's morphomes. Hence, the system does not require building blocks to have particular meanings. Instead, it looks for *pre-morphosyntactic* units, i.e., ones assembled from phonemes, but not yet assigned a syntactic or semantic meaning. In a larger pipeline, such building blocks could serve as an interface between morphosyntax and phonology. For instance, while our system can find Hebrew's default masculine suffix *-im*, it does not specify whether it is in fact masculine in a given word or whether it is feminine, as this suffix also occurs in idiosyncratic feminine plurals.

Our system also encounters building blocks like the *t* in fig. 1, which might be called "quasi-morphemes" since they recur in a wide range of related forms, but fall just short of being entirely systematic.[2] The *t* in fig. 1 seems to be frequently associated with the feminine morphosyntactic category, as in the feminine nationality suffix *-it* (*sini**t*** 'Chinese (F)'), the suffix *-wt* for deriving abstract mass nouns (*bhirw**t*** 'clarity (F)'), as well as in feminine singular and plural present-tense verb endings (e.g., *kwtb-**t*** 'she writes' and *kwtb-w**t*** 'they (F.PL) write', respectively).

In fig. 1(a), note that this *t* is present in both the F.SG and F.PL forms. However, it cannot be assigned a distinct meaning such as "feminine," since it cannot be separated from the *w* in the F.PL suffix *-wt*.[3] Moreover, this *t* is not always the F.SG marker; the ending *-h* in fig. 1(b) is also common. Nevertheless, the frequency with which *t* occurs in feminine words does not seem to be accidental. It seems instead to be some kind of building block, and our system treats it as such.

[2]Though, see Faust (2013) for an analysis positing /-t/ as Hebrew's one (underlying) feminine-gender marker.

[3]If the *w* in *-wt* meant "plural," we would expect the default M.PL suffix to be *-wm* instead of *-im*.

Because our system is not intended to identify morphosyntactic categories, its evaluation poses a challenge, as morphological analyzers tend to pair form with meaning. Nevertheless, we tentatively evaluate our system's clusters against the *modified* output of a finite-state morphological analyzer. That is, we map this analyzer's abstract morphosyntactic categories onto categories that, while still essentially morphosyntactic, correspond more closely to distinctions in form (see section 4).

2 Morphology and MCMMs

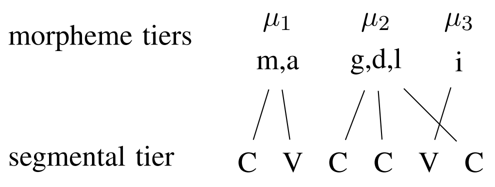
In this section, we will examine autosegemental (or *multi-linear*) morphology (McCarthy, 1981), to isolate the property that allows it to handle non-concatenative morphology. We will then show that because an MCMM has this same property, it is an appropriate computational model for learning non-concatenative morphology.

First, we note some previous work connecting autosegmental morphology to computation. For example, Kiraz (1996) provides a framework for autosegmental morphology within two-level morphology, using hand-written grammars. By contrast, Fullwood and O'Donnell (2013) provide a learning algorithm in the spirit of autosegmental morphology. They sample templates, roots, and residues from Pitmor-Yor processes, where a *residue* consists of a word's non-root phonemes, and a *template* specifies word length and the word-internal positions of root phonemes. Botha and Blunsom (2013) use mildly context-free grammars with crossing branches to generate words with discontiguous morphemes. The present work, in contrast, assumes nothing about structure beforehand.

Other works implement certain components of autosegmental theory (e.g., Goldsmith and Xanthos, 2009) or relegate it to a certain phase in their overall system (e.g., Rodrigues and Ćavar, 2005). The present work seeks to simulate autosegmental morphology in a more general and holistic way.

2.1 Multilinear morphology

The central aspect of autosegmental theory (McCarthy, 1981) is its multi-linear architecture, i.e., its use of a *segmental tier* along with many *autosegmental tiers* to account for morphological structure. The segmental tier is a series of placeholders for consonants and vowels, often called the *CV skeleton*. The other tiers each represent a particular morpheme. Fig. 2(a) shows four tiers. One is the CV skeleton. The other three, labeled μ_1, μ_2, and μ_3, are morphemes.[4]



(a) Multi-linear approach

single tier m a | **g d** | i | **l**

(b) Linear approach

Figure 2: Multiple tiers vs. a single tier

Notice that μ_2, the consonantal root, is discontinuous; it is interrupted by μ_3. If a model has only one tier, as in fig. 2(b), there would be no way of representing the unity of μ_2, i.e., that *g*, *d*, and *l* all belong to the same morpheme. With this multi-tier aspect of autosegmental morphology in mind, we can now state two criteria for a model of non-concatenative morphology:

(1) a. Morphemes are represented as being separate from the segmental tier.
    b. Each morpheme tier (or node) is orthogonal to all other morpheme tiers.

Criterion (1b) implies that the morpheme tiers are unordered. Without sequential dependencies between morpheme tiers, crossing edges such as those in fig. 2(a) are made possible. We should note that autosegmental morphology has other properties to constrain morphological structure, e.g., the well-formedness principle; at present, we are not concerned with capturing all aspects of autosegmental morphology, but instead in building a generic system to which one can later add linguistically motivated constraints.

2.2 A graph-theoretic interpretation

In graph-theoretic terms, the multi-linear formalism of McCarthy (1981) is a type of *multipartite* graph. This is a graph whose nodes can be partitioned into N sets of *mutually nonadjacent* nodes, i.e., N sets such that no two nodes within the *same* set are connected by an edge. Fig. 3, for example, shows a *bipartite* graph, i.e., a graph with two partitions, in this case the sets M and R. Within each

[4]Although McCarthy uses the term *morpheme* rather than *morphome*, the same principles apply.

set, all nodes are independent; the only connections are between nodes of different sets.

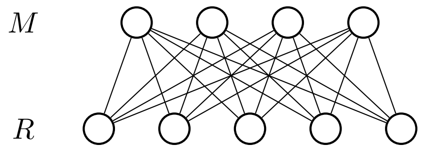


Figure 3: Bipartite graph

As it turns out, a bipartite graph suffices to capture the essential properties of McCarthy's autosegmental framework, for a bipartite graph meets the two criteria stated in (1). We can reformulate the morpheme tiers and the segmental tier in fig. 2(a) as the sets M and R, respectively, in fig. 3—disjoint by the definition of *bipartite*. This satisfies the first criterion. For the second, each node in M represents a morpheme (or morpheme tier), and, by the definition of *bipartite*, the nodes within M are independent and thus orthogonal.

An MCMM (section 3) is well-suited for the learning of non-concatenative morphology because it is bipartite graph. It has two *layers* (equivalently, sets) of nodes, a hidden layer and a surface layer—corresponding, respectively, to M and R in fig. 3. There are no intra-layer connections in an MCMM, only connections between layers.

We will henceforth refer to an MCMM's two partitions of nodes as *vectors* of nodes and will use matrix and vector notation to describe the components of an MCMM: uppercase boldface letters refer to matrices, lowercase boldface letters refer to vectors, and italicized lowercase letters refer to the individual elements of vectors/matrices. For example, $m_{i,k}$ is the k^{th} element in the vector $\mathbf{m}_i$, which is the i^{th} row in the $I \times K$ matrix $\mathbf{M}$. Thus, we will henceforth write the M and R in fig. 3 as $\mathbf{m}$ and $\mathbf{r}$, respectively (or $\mathbf{m}_i$ and $\mathbf{r}_i$, where i is the index of the i^{th} word).

3 The Multiple Cause Mixture Model

A Multiple Cause Mixture Model (MCMM) (Saund, 1993, 1994) is a graphical model consisting of a layer of surface nodes and a hidden layer of causal nodes. The hidden nodes (or units) are connected to surface nodes by weights. Each surface node is either ON (active) or OFF (inactive) depending on the hidden-node activities and the weights connecting hidden nodes to surface nodes.

3.1 Architecture

An MCMM can be viewed as a variation of the classical autoencoder network (Dayan and Zemel, 1995), a type of neural network used for unsupervised learning. In autoencoders, a hidden layer is forced to learn a compression scheme, i.e., a lower-dimensional encoding, for a dataset.

MCMMs are called *Multiple Cause* Mixture Models because more than one hidden unit can take part in the activation of a surface unit. This is illustrated in figure 4, where the nodes $\mathbf{m}$ are the hidden units, and $\mathbf{r}$ is the (reconstructed) surface vector. Each arc $c_{j,k}$ represents the weight on the connection between m_k and r_j. The activity of r_j is determined by a mixing function (section 3.2).

The MCMM learns by comparing the reconstructed vector $\mathbf{r}_i$ to its corresponding original datapoint $\mathbf{d}_i$. The discrepancy between the two is quantified by an *objective function*. If there is a discrepancy, the values of the nodes in $\mathbf{m}_i$ as well as the weights $\mathbf{C}$ are adjusted in order to reduce the discrepancy as much as possible. See section 3.3 for more on the learning process.

Suppose data points $\mathbf{d}_u$ and $\mathbf{d}_v$ have some features in common. Then, as the MCMM tries to reconstruct them in $\mathbf{r}_u$ and $\mathbf{r}_v$, respectively, similarities will emerge between their respective hidden-layer vectors $\mathbf{m}_u$ and $\mathbf{m}_v$. In particular, the vectors $\mathbf{m}_u$ and $\mathbf{m}_v$ should come to share at least one active node, i.e., at least one $k \in K$ such that $m_{u,k} = 1$ and $m_{v,k} = 1$. This can serve as a basis for clustering; i.e., $m_{i,k}$ indicates whether $\mathbf{d}_i$ is a member of cluster k.

3.2 Mixing Function

The mapping between the layer of hidden nodes $\mathbf{m}$ and the layer of surface nodes $\mathbf{r}$ is governed by a *mixing function*, which is essentially a voting rule (Saund, 1994); it maps from a set of input "votes" to a single output decision. The output decision is the activity (or inactivity) of a node r_{ij} in the surface layer. Following Saund (1994), we use the Noisy-Or function:

$$r_{i,j} = 1 - \prod_k (1 - m_{i,k} c_{j,k}) \qquad (1)$$

Note that the input to this function includes not only the hidden nodes $\mathbf{m}$, but also the *weights* $\mathbf{c}_j$ on the hidden nodes. That is, the activity of the hidden node m_k is weighted by the value c_{jk}. A classical autoencoder also has a mixing function,

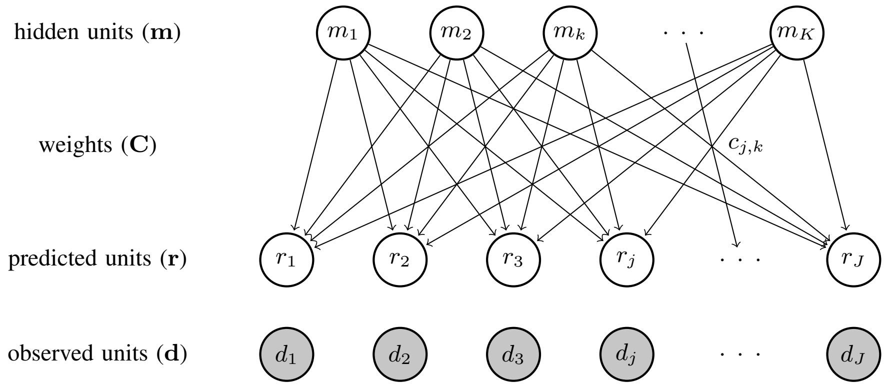


Figure 4: Architecture of a Multiple Cause Mixture Model (MCMM).

though it is more commonly called an *activation function* in autoencoders. The most common activation function involves a simple weighted sum of the hidden layer $\mathbf{m}$'s activations. The entirely linear weighted sum is then passed to the logistic sigmoid function σ, which squashes the sum to a number between 0 and 1:

$$r_{i,j} = \sigma\Big(\sum_k m_{i,k} c_{j,k}\Big) \tag{2}$$

Notice that both (1) and (2), have the same three primary components: the output (or surface node) $r_{i,j}$, the hidden layer of nodes $\mathbf{m}$, and a matrix of weights $\mathbf{C}$. Both are possible mixing functions.

3.3 Learning

In both the classical autoencoder and the MCMM, learning occurs as a result of the algorithm's search for an optimal valuation of key variables (e.g., weights), i.e., a valuation that minimizes the discrepancy between reconstructed and original data points. The search is conducted via numerical optimization; we use the nonlinear conjugate gradient method. Our objective function is a simple error function, namely the normalized sum of squares error:

$$E = \frac{1}{I \times J} \sum_i \sum_j (r_{i,j} - d_{i,j})^2 \tag{3}$$

where $I \times J$ is the total number of features in the dataset. The MCMM's task is to minimize this function by adjusting the values in $\mathbf{M}$ and $\mathbf{C}$, where $\mathbf{M}$ is the $I \times K$ matrix that encodes each data point's cluster-activity vector, and $\mathbf{C}$ is the $J \times K$ matrix that encodes the weights between $\mathbf{m}_i$ and $\mathbf{r}_i$ for every $i \in I$ (see fig. 5).

The MCMM's learning process is similar to Expectation Maximization (EM) in that at any given time it is holding one set of variables fixed while optimizing the other set. We thus have two functions, OPTIMIZE-M and OPTIMIZE-C, which take turns optimizing their respective matrices.

The function OPTIMIZE-M visits each of the I cluster-activity vectors $\mathbf{m}_i$ in $\mathbf{M}$, optimizing each one separately. For each $\mathbf{m}_i$, OPTIMIZE-M enters an optimization loop over its K components, adjusting each $m_{i,k}$ by a quantity proportional to the negative gradient of E at $m_{i,k}$. This loop repeats until E ceases to decrease significantly, whereupon OPTIMIZE-M proceeds to the next $\mathbf{m}_i$.

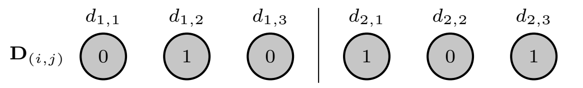
The function OPTIMIZE-C consists of a single optimization loop over the entire matrix $\mathbf{C}$. Each $c_{j,k}$ is adjusted by a quantity proportional to the negative gradient of E at $c_{j,k}$. Unlike OPTIMIZE-M, which comprises I separate optimization loops, OPTIMIZE-C consists of just one, When each of its $J \times K$ components has been adjusted, one round of updates to $\mathbf{C}$ is complete. E is reassessed only between completed rounds of updates. If the change in E remains significant, another round begins.

Both OPTIMIZE-M and OPTIMIZE-C are enclosed within an "alternation loop" that alternates between the two functions, holding $\mathbf{C}$ fixed during OPTIMIZE-M, and vice versa. This alternation continues until E cannot be decreased further. At this point, an "outer loop" splits the cluster which contributes the most to the error, adds one to the cluster count K, and restarts the alternation loop. The outer loop repeats until it reaches an overall stopping criterion, e.g., $E = 0$.

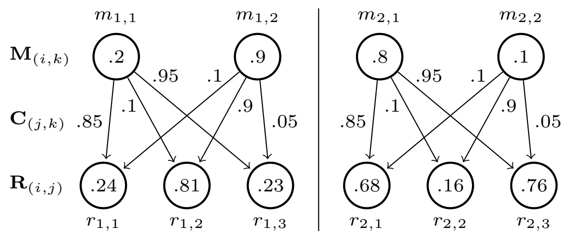
The optimization task is subject to the constraint that no value in $\mathbf{M}$ or $\mathbf{C}$ may exceed 1 or fall below 0. In other words, it is a task of bound constrained optimization. Thus, whenever a value in either $\mathbf{M}$ or $\mathbf{C}$ is about to fall below 0, it is set to 0. Likewise, whenever a value is about to exceed 1, it is set to 1 (Ni and Yuan, 1997).

3.4 A Simple MCMM Example

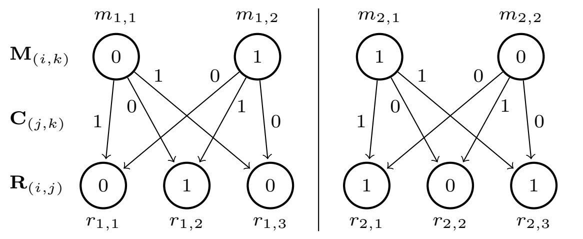
Fig. 5 shows an example of an MCMM for two data points (i.e., $I = 2$). The hidden cluster activities $\mathbf{M}$, the weights $\mathbf{C}$, and the mixing function r constitute a model that reproduces the observed data points $\mathbf{D}$. The nodes $m_{i,k}$ represent cluster activities; if $m_{1,2} = 1$, for instance, the second cluster is active for $\mathbf{d}_1$ (i.e., $\mathbf{d}_1$ is a member of cluster 2). Note that the $J \times K$ weight matrix $\mathbf{C}$ is the same for all data points, and the k^{th} row in $\mathbf{C}$ can be seen as the k^{th} cluster's "average" vector: the j^{th} component in $\mathbf{c}_k$ is 1 only if all data points in cluster k have 1 at feature j.



(a) Observed Data



(b) Learning in Progress



(c) Convergence

where $r_{i,j} = 1 - \Pi_{k=1}^{K}(1 - m_{i,k}c_{j,k})$
[NOISY-OR function]

Figure 5: A simple MCMM example

We can see that while learning is in progress, the cluster activities ($m_{i,k}$) and the cluster centers ($c_{j,k}$) are in flux, as the error rate is being reduced, but that they converge to values of 0 and 1. At convergence, a reconstruction node ($r_{i,j}$) is 1 if at least one $m_{i,k}c_{j,k} = 1$ (and 0 otherwise).

3.5 MCMMs for Morphology

To apply MCMMs to morphological learning, we view the components as follows. For each word i, the observed (d_j) and reconstructed (r_j) units refer to binary surface features extracted from the word (e.g., "the third character is *s*"). The hidden units (m_k) correspond to clusters of words which, in the ideal case, contain the same morpheme (or morphome). The weights ($c_{j,k}$) then link specific morphemes to specific features.

For an example, consider the English word *ads*. Ideally, there would be two clusters derived from the MCMM algorithm, one for the stem *ad* and one clustering words with the plural ending *-s*. Fig. 6 shows a properly learned MCMM, based upon *positional* features: one feature for each letter in each position. Note that *ads* does not have partial membership of the *ad* and *-s* hidden units, but is a full member of both.

4 Experiments

4.1 Gold Standard

Our dataset is the Hebrew word list (6888 unique words) used by Daya et al. (2008) in their study of automatic root identification. This list specifies the root for the two-thirds of the words that have roots. Only the roots are specified, however, and not other (non-root) properties. To obtain other morphological properties, we use the MILA Morphological Analysis tool (MILA-MA) (Itai and Wintner, 2008). Because its morphological knowledge is manually coded and its output deterministic, MILA-MA provides a good approximation to human annotation. The system is designed to analyze morphemes, not morphomes, an issue we partially account for in our category mappings and take up further in section 4.4.

As an example of the way we use MILA-MA's output, consider the word *bycim* ('in trees'), which MILA-MA analyzes as a `M%Pl` noun bearing the prefixal preposition *b-* ('in'). Given this analysis, we examine each MCMM-generated cluster that contains *bycim*. In particular, we want to see if *bycim* has been grouped with other words that MILA-MA has labeled as `M%Pl` or as having *b-*.

Category mappings MILA-MA outputs 22 possible feature labels. Four of these (`id`, `undotted`, `transliterated`, `register`)

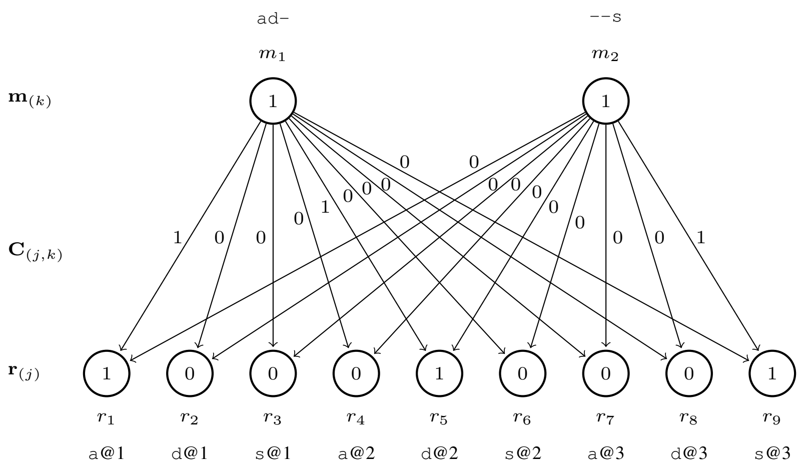

ad-
--s
m_1
m_2
$\mathbf{m}_{(k)}$
$\mathbf{C}_{(j,k)}$
$\mathbf{r}_{(j)}$
r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9
a@1 d@1 s@1 a@2 d@2 s@2 a@3 d@3 s@3

Figure 6: An MCMM example for the word *ads*, with nine features (three letters, each at three positions), and two clusters "causing" the word

are irrelevant and are discarded. Each of the 18 remaining features has at least two values, and some have many more, resulting in a great many feature-value pairs, i.e., categories.

Most part-of-speech (POS) categories are rather abstract; they often cannot be linked to particular elements of form. For example, a noun can be masculine or feminine, can be singular or plural, and can bear one of a variety of derivational affixes. But there is no *single* marker that unifies *all* nouns. The situation with verbs is similar: every Hebrew verb belongs to one of seven classes (*binyanim*), each of which is characterized by a distinctive vowel pattern.

We thus replace "super-categories" like NOUN and VERB with finer-grained categories that point to actual distinctions in form. In fact, the only POS categories we keep are those for adverbs, adjectives, and numerals (ordinal and cardinal). The rest are replaced by composite (sub-)categories (see below) or discarded entirely, as are negative categories (e.g., `construct:false`) and unmarked forms (e.g., `M%Sg` in nominals).

Sometimes two or more morphosyntactic categories share an element of form; e.g., the future-tense prefix *t-* can indicate the 2nd person in the MASC gender or, in the FEM gender, either the 2nd or 3rd person:

`temwr`	'you (M.SG) will keep'
`temwr`	'she will keep'
`temwrw`	'you (F.PL) will keep'

Verb inflections are thus mapped to composite categories, e.g., `future%(2%M)|(2|3%F)`, where the symbol | means 'or'. We also map MILA-MA's `binyan` and `tense` feature-value pairs to `stem_type`, since the shape of a verb's stem follows from its binyan, and, in some binyanim, past and future tenses have different stems. Both the composite-inflectional and `stem_type` categories represent complex mappings between morphosyntax and phonology.

However, since it is not yet entirely clear what would constitute a fair evaluation of the MCMM's clusters (see section 5), we generally try to retain the traditional morphosyntatctic labels in some form, even if these traditional labels exist only in combination with other labels. Most of our mappings involve fusing "atomic" morphosyntactic categories. For example, to capture Hebrew's fusional plural suffixes for nominals, we combine the atomic categories FEM or MASC with PL; i.e., MASC + PL $\mapsto$ `M%Pl` and FEM + PL $\mapsto$ `F%Pl`.

Whenever ambiguity is systematic and thus predictable, we choose the most general analysis. For instance, participle analyses are always accompanied by adjective and noun analyses (cf. English *-ing* forms). Since a participle is always both a noun and an adjective, we keep the participle analysis and discard the other two. Finally, we use `rootless:Nominal` to capture orthographic regularities in loan words. In sum, we employ reasonably motivated and informative categories, but the choice of mappings is nontrivial and worthy of investigation in its own right.

4.2 Thresholds and Features

The MCMM begins the learning process with a single cluster, and whenever its error stops decreasing significantly, it adds a cluster. It is supposed to continue to add clusters until it converges, i.e., until the error is (close to) 0, but so far our MCMM has never converged. As the number of clusters increases, the MCMM becomes increasingly encumbered by the sheer number of computations it must perform. We thus stop it when the number of clusters K reaches a pre-set limit: for this paper, the limit was $K = 100$. Such a cut-off leaves most of the cluster activities in $\mathbf{M}$ between 0 and 1. We set a threshold for cluster membership at 0.5: if $m_{i,k}c_{j,k} \geq 0.5$ for at least one index j in J, then $\mathbf{d}_i$ is a member of the k^{th} cluster.

If $m_{i,k}c_{j,k} \leq 0.5$ for all j in J, we say that the k^{th} cluster is *inactive* in the i^{th} word. If a cluster is inactive for *every* word, we say that it is currently only a *potential* cluster rather than an *actual* one.

Each word is encoded as a vector of features. This vector is the same length for all words. For any given word, certain features will be ON (with values = 1), and the rest—a much greater portion—will be OFF (with values = 0). Each feature is a statement about a word's form, e.g., "the first letter is *b*" or "*i* occurs before *t*". In our features, we attempt to capture some of the information implicit in a word's visual representation.

A **positional feature** indicates the presence of a particular character at a certain position, e.g., `m@[0]`, for '*m* at the first position' or `l@[-2]` for '*l* at the second-to-last position'. Each data point $\mathbf{d}_i$ contains positional features corresponding to the first s and the final s positions in word i, where s is a system parameter (section 4.4). With 22 letters in the Hebrew alphabet, this amounts to $22 \times s \times 2$ positional features.

A **precedence feature** indicates, for two characters a and b, whether a precedes b within a certain distance (or number of characters). This distance is the system parameter δ. We define δ as the difference between the indices of the characters a and b. For example, if $\delta = 1$, then characters a and b are adjacent. The number of precedence features is the length of the alphabet squared ($22^2 = 484$).

4.3 Evaluation Metrics

We evaluate our clustering results according to three metrics. Let U denote the set of M returned clusters and V the set of N gold-standard categories. The idea behind *purity* is to compute the proportion of examples assigned to the correct cluster, using the most frequent category within a given cluster as gold. Standard purity assumes each example belongs to only one gold category. For a dataset like ours consisting of multi-category examples, this can yield purities greater than 1. We thus modify the calculations slightly to compute **average cluster-wise purity**, as in (4), where we divide by M. While this equation yields purities within $[0, 1]$, even when clusters overlap, it retains the metric's bias toward small clusters.

$$\text{pur}_{\text{avg}}(U, V) = \frac{1}{M} \sum_{m \in M} \frac{\max_n |u_m \cap v_n|}{M} \quad (4)$$

Given this bias, we incorporate other metrics: **BCubed precision** and **BCubed recall** (Bagga and Baldwin, 1998) compare the cluster mappings of x with those of y, for every pair of data points x and y. These metrics are well-suited to cases of overlapping clusters (Artiles and Verdejo, 2009). Suppose x and y share m clusters and n categories. BCubed precision measures the extent to which $m \leq n$. It is 1 as long there are not more clusters than gold-standard categories. BCubed Recall measures the extent to which $m \geq n$. See Artiles and Verdejo (2009) for calculation details.

4.4 Results

With a cut-off point at $K = 100$ clusters, we ran the MCMM at different valuations of s and δ. The results are given in table 1, where "$\delta = *$" means that δ is the entire length of the word in question, and "n/a" means that the feature type in question was left out; e.g., in the s column, "n/a" means that no positional features were used. Depending upon the threshold (section 4.2), a cluster may be empty: K' is the number of *actual* clusters (see section 4.2). *Cov(erage)*, on the other hand, is the number of words that belong to least one cluster. The valuations $s = 1$ and $\delta = 1$ or 2 seem to produce the best overall results.[5]

Some of the clusters appear to be capturing key properties of Hebrew morphology, as evidenced by the MILA-MA categories. For example, in one cluster, 677 out of 942 words turn to be of the composite MILA-MA category `M%Pl`, a purity of 0.72.[6] In another cluster, this one containing 584 words, 483 are of the MILA-MA category

[5]While $s = 1$ indicates an preference for learning short prefixes and suffixes, it is important to note that more than one-letter affixes may be learned through the use of the precedence features, which can occur anywhere in a word.

[6]Recall that `M%Pl` is merger of the originally separate MILA-MA categories `M` and `Pl` .

`preposition:l` (the prefixal preposition *l*-), a purity of 0.83.

Thus, in many cases, the evaluation recognizes the efficacy of the method and helps sort the different parameters. However, it has distinct limitations. Our gold standard categories are modified categories from MILA-MA, which are not entirely form-based. For example, in one 1016-word cluster, the three most common gold-standard categories are `F%Pl` (441 words), `F%Sg` (333 words), and `pos:adjective` (282 words). Taking the most frequent category as the correct label, the purity of this cluster is $\frac{441}{1016} = 0.434$. However, a simple examination of this cluster's words reveals it to be more coherent than this suggests. Of the 1016 words, 92% end in *t*; in 96%, *t* is one of the final two characters; and in 98%, one of the final three. When *t* is not word-final, it is generally followed by a morpheme and thus is stem-final. Indeed, this cluster seems to have captured almost exactly the "quasi-morpheme" *t* discussed in section 1. Thus, an evaluation with more form-based categories might measure this cluster's purity to be around 98%—a point for future work.

None of the experiments reported here produced *actual* (section 4.2) clusters representing consonantal roots. However, past experiments did produce some consonantal-root clusters. In these clusters, the roots were often discontinuous, e.g., *z.k.r* in the words *lizkwr*, *lhzkir*, and *zikrwn*. It is not yet clear to us why these past experiments produced actual root clusters and the present ones did not, but, in any case, we expect to see more root clusters as K (and especially K') increases.

s	δ	Purity	BP	BR	Cov.	K'
n/a	1	0.394	0.456	0.223	3279	12
n/a	2	0.330	0.385	0.218	4002	14
n/a	3	0.396	0.423	0.261	4214	19
n/a	*	0.379	0.422	0.319	4495	20
1	n/a	0.576	0.599	0.458	3577	4
2	n/a	0.428	0.488	0.396	5942	12
3	n/a	0.429	0.508	0.370	6384	18
1	1	0.463	0.580	0.325	5760	16
1	2	0.443	0.540	0.358	5401	14
1	3	0.458	0.500	0.369	5144	12
1	*	0.456	0.518	0.383	5096	14
2	1	0.371	0.460	0.298	6316	26
2	2	0.401	0.481	0.291	5728	20
2	3	0.392	0.465	0.366	5509	17
2	*	0.412	0.474	0.347	5366	18
3	1	0.399	0.461	0.334	6102	19
3	2	0.403	0.474	0.326	5756	19
3	3	0.364	0.438	0.345	5164	17
3	*	0.391	0.463	0.390	5496	17

Table 1: Results at $K = 100$

5 Summary and Outlook

We have presented a model for the unsupervised learning of morphology, the Multiple Cause Mixture Model, which relies on hidden units to generate surface forms and maps to autosegmental models of morphology. Our experiments on Hebrew, using different types of features, have demonstrated the potential utility of this method for discovering morphological patterns.

So far, we have been stopping the MCMM at a set number (K) of clusters because computational complexity increases with K: the complexity is proportional to $I \times J \times K$, with $I \times J$ already large. But if the model is to find consonantal roots along with affixes, K is going to have to be much larger. We can attack this problem by taking advantage of the nature of bipartite graphs (section 2): with intra-layer independence, every $r_{i,j}$ in the vector $\mathbf{r}_i$—and thus each element in the entire matrix $\mathbf{R}$—can be computed *in parallel*. We are currently parallelizing key portions of our code, rewriting costly loops as kernels to be processed on the GPU.

In a different vein, we intend to adopt a better method of evaluating the MCMM's clusters, one more appropriate for the *morphome*-like nature of the clusters. Such a method will require gold-standard categories that are morphomic rather than morphosyntactic, and we anticipate this to be a nontrivial undertaking. From the theoretical side, an exact inventory of (Hebrew) morphomes has not been specified in any work we know of, and annotation criteria thus need to be established. From the practical side, MILA-MA provides neither segmentation nor derivational morphology for anything other than verbs, and so much of the annotation will have to built from scratch.

Finally, our data for this work consisted of Modern Hebrew words that originally appeared in print. They are spelled according to the orthographic conventions of Modern Hebrew, i.e., without representing many vowels. As vowel absences may obscure patterns, we intend to try out the MCMM on phonetically transcribed Hebrew.

References

Mark Aronoff. 1994. *Morphology by Itself: Stems and Inflectional Classes*, volume 22 of *Linguistic Inquiry Monograph*. MIT Press, Cambridge, MA.

Enrique Amigó Julio Gonzalo Javier Artiles and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval* 12(4):353–371.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics: Volume 1*. Association for Computational Linguistics, pages 79–85.

Jan A. Botha and Phil Blunsom. 2013. Adaptor grammars for learning non-concatenative morphology. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*. Seattle, pages 345–356.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing* 4(1):3.

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

Peter Dayan and Richard S Zemel. 1995. Competition and multiple cause models. *Neural Computation* 7:565–579.

Khaled Elghamry. 2005. A constraint-based algorithm for the identification of arabic roots. In *Proceedings of the Midwest Computational Linguistics Colloquium*. Indiana University.

Noam Faust. 2013. Decomposing the feminine suffixes of modern hebrew: a morpho-syntactic analysis. *Morphology* 23(4):409–440.

Michelle Fullwood and Tim O'Donnell. 2013. Learning non-concatenative morphology. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*. Sofia, Bulgaria, pages 21–27.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics* 27:153–198.

John Goldsmith and Aris Xanthos. 2009. Learning phonological categories. *Language* 85(1):4–38.

Harald Hammarstrom and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics* 37(2):309 – 350.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation* 42(1):75–98.

George Anton Kiraz. 1996. Computing prosodic morphology. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996)*. Copenhagen, pages 664–669.

Andreas Klöckner, Nicolas Pinto, Yunsup Lee, B. Catanzaro, Paul Ivanov, and Ahmed Fasih. 2012. PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation. *Parallel Computing* 38(3):157–174.

John J McCarthy. 1981. A prosodic theory of nonconcatenative morphology. *Linguistic inquiry* 12:373–418.

Taesun Moon, Katrin Erk, and Jason Baldridge. 2009. Unsupervised morphological segmentation and clustering with document boundaries. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*. Association for Computational Linguistics, pages 668–677.

Q Ni and Y Yuan. 1997. A subspace limited memory quasi-newton algorithm for large-scale nonlinear bound constrained optimization. *Mathematics of Computation of the American Mathematical Society* 66(220):1509–1520.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *In ACL. Ariya Rastrow, Abhinav Sethy, and Bhuvana Ramabhadran*.

Paul Rodrigues and Damir Ćavar. 2005. Learning arabic morphology using information theory. In *Proceedings from the Annual Meeting of the Chicago Linguistic Society*. Chicago Linguistic Society, pages 49–58.

Eric Saund. 1993. Unsupervised learning of mixtures of multiple causes in binary data. In *NIPS*. pages 27–34.

Eric Saund. 1994. A multiple cause mixture model for unsupervised learning. *Neural Computation* 7(1):51–71.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and N. Nativ. 2001. Building a tree-bank of Modern Hebrew text. *Traitment Automatique des Langues* 42(2).

Gregory T Stump. 2001. *Inflectional morphology: A theory of paradigm structure*, volume 93. Cambridge University Press.