# Learning Hebrew Roots: Machine Learning with Linguistic Constraints

**Ezra Daya**
Dept. of Computer Science
University of Haifa
31905 Haifa
Israel
edaya@cs.haifa.ac.il

**Dan Roth**
Dept. of Computer Science
University of Illinois
Urbana, IL 61801
USA
danr@cs.uiuc.edu

**Shuly Wintner**
Dept. of Computer Science
University of Haifa
31905 Haifa
Israel
shuly@cs.haifa.ac.il

## Abstract

The morphology of Semitic languages is unique in the sense that the major word-formation mechanism is an inherently non-concatenative process of *interdigitation*, whereby two morphemes, a *root* and a *pattern*, are interwoven. Identifying the root of a given word in a Semitic language is an important task, in some cases a crucial part of morphological analysis. It is also a non-trivial task, which many humans find challenging. We present a machine learning approach to the problem of extracting roots of Hebrew words. Given the large number of potential roots (thousands), we address the problem as one of combining several classifiers, each predicting the value of one of the root's consonants. We show that when these predictors are combined by enforcing some fairly simple linguistics constraints, high accuracy, which compares favorably with human performance on this task, can be achieved.

## 1 Introduction

The standard account of word-formation processes in Semitic languages describes words as combinations of two morphemes: a *root* and a *pattern*.[1] The root consists of consonants only, by default three (although longer roots are known), called *radicals*. The pattern is a combination of vowels and, possibly, consonants too, with 'slots' into which the root consonants can be inserted. Words are created by *interdigitating* roots into patterns: the first radical is inserted into the first consonantal slot of the pattern, the second radical fills the second slot and the third fills the last slot. See Shimron (2003) for a survey.

Identifying the root of a given word is an important task. Although existing morphological analyzers for Hebrew only provide a lexeme (which is a combination of a root and a pattern), for other Semitic languages, notably Arabic, the root is an essential part of any morphological analysis simply because traditional dictionaries are organized by root, rather than by lexeme. Furthermore, roots are known to carry some meaning, albeit vague. We believe that this information can be useful for computational applications and are currently experimenting with the benefits of using root and pattern information for automating the construction of a WordNet for Hebrew.

We present a machine learning approach, augmented by limited linguistic knowledge, to the problem of identifying the roots of Hebrew words. To the best of our knowledge, this is the first application of machine learning to this problem. While there exist programs which can extract the root of words in Arabic (Beesley, 1998a; Beesley, 1998b) and Hebrew (Choueka, 1990), they are all dependent on labor intensive construction of large-scale lexicons which are components of full-scale morphological analyzers. Note that Tim Bockwalter's Arabic morphological analyzer[2] only uses "word stems – rather than root and pattern morphemes – to identify lexical items. (The information on root and pattern morphemes could be added to each stem entry if this were desired.)" The challenge of our work is to automate this process, avoiding the bottleneck of having to laboriously list the root and pattern of each lexeme in the language, and thereby gain insights that can be used for more detailed morphological analysis of Semitic languages.

As we show in section 2, identifying roots is a non-trivial problem even for humans, due to the complex nature of Hebrew derivational and inflectional morphology and the peculiarities of the Hebrew orthography. From a machine learning perspective, this is an interesting test case of interactions among different yet interdependent classifiers. After presenting the data in section 3, we discuss a simple, baseline, learning approach (section 4) and then propose two methods for combining the results of interdependent classifiers (section 5), one which is purely statistical and one which incorporates lin-

[1] An additional morpheme, *vocalization*, is used to abstract the pattern further; for the present purposes, this distinction is irrelevant.

[2] http://www.qamus.org/morphology.htm

guistic constraints, demonstrating the improvement of the hybrid approach. We conclude with suggestions for future research.

## 2 Linguistic background

In this section we refer to Hebrew only, although much of the description is valid for other Semitic languages as well. As an example of root-and-pattern morphology, consider the Hebrew roots *g.d.l, k.t.b* and *r.$.m* and the patterns *haCCaCa, hitCaCCut* and *miCCaC*, where the 'C's indicate the slots. When the roots combine with these patterns the resulting lexemes are *hagdala, hitgadlut, migdal, haktaba, hitkatbut, miktab, har$ama, hitra$mut, mir$am*, respectively. After the root combines with the pattern, some morpho-phonological alternations take place, which may be non-trivial: for example, the *hitCaCCut* pattern triggers assimilation when the first consonant of the root is *t* or *d*: thus, *d.r.$+hitCaCCut* yields *hiddar$ut*. The same pattern triggers metathesis when the first radical is *s* or *$*: *s.d.r+hitCaCCut* yields *histadrut* rather than the expected **hitsadrut*. Semi-vowels such as *w* or *y* in the root are frequently combined with the vowels of the pattern, so that *q.w.m+haCCaCa* yields *haqama*, etc. Frequently, root consonants such as *w* or *y* are altogether missing from the resulting form.

These matters are complicated further due to two sources: first, the standard Hebrew orthography leaves most of the vowels unspecified. It does not explicate *a* and *e* vowels, does not distinguish between *o* and *u* vowels and leaves many of the *i* vowels unspecified. Furthermore, the single letter *w* is used both for the vowels *o* and *u* and for the consonant *v*, whereas *i* is similarly used both for the vowels *i* and for the consonant *y*. On top of that, the script dictates that many particles, including four of the most frequent prepositions, the definite article, the coordinating conjunction and some subordinating conjunctions all attach to the words which immediately follow them. Thus, a form such as *mhgr* can be read as a lexeme ("immigrant"), as *m-hgr* "from Hagar"or even as *m-h-gr* "from the foreigner". Note that there is no deterministic way to tell whether the first *m* of the form is part of the pattern, the root or a prefixing particle (the preposition *m* "from").

The Hebrew script has 22 letters, all of which can be considered consonants. The number of tri-consonantal roots is thus theoretically bounded by $22^3$, although several phonological constraints limit this number to a much smaller value. For example, while roots whose second and third radicals are identical abound in Semitic languages, roots whose first and second radicals are identical are extremely rare (see McCarthy (1981) for a theoretical explanation). To estimate the number of roots in Hebrew we compiled a list of roots from two sources: a dictionary (Even-Shoshan, 1993) and the verb paradigm tables of Zdaqa (1974). The union of these yields a list of 2152 roots.[3]

While most Hebrew roots are regular, many belong to *weak paradigms*, which means that root consonants undergo changes in some patterns. Examples include *i* or *n* as the first root consonant, *w* or *i* as the second, *i* as the third and roots whose second and third consonants are identical. For example, consider the pattern *hCCCh*. Regular roots such as *p.s.q* yield forms such as *hpsqh*. However, the irregular roots *n.p.l, i.c.g, q.w.m* and *g.n.n* in this pattern yield the seemingly similar forms *hplh, hcgh, hqmh* and *hgnh*, respectively. Note that in the first and second examples, the first radical (*n* or *i*) is missing, in the third the second radical (*w*) is omitted and in the last example one of the two identical radicals is omitted. Consequently, a form such as $hC_1C_2h$ can have any of the roots $n.C_1.C_2$, $C_1.w.C_2$, $C_1.i.C_2$, $C_1.C_2.C_2$ and even, in some cases, $i.C_1.C_2$.

While the Hebrew script is highly ambiguous, ambiguity is somewhat reduced for the task we consider here, as many of the possible lexemes of a given form share the same root. Still, in order to correctly identify the root of a given word, context must be taken into consideration. For example, the form *$mnh* has more than a dozen readings, including the adjective "fat" (feminine singular), which has the root *$.m.n*, and the verb "count", whose root is *m.n.i*, preceded by a subordinating conjunction. In the experiments we describe below we ignore context completely, so our results are handicapped by design.

## 3 Data and methodology

We take a machine learning approach to the problem of determining the root of a given word. For training and testing, a Hebrew linguist manually tagged a corpus of 15,000 words (a set of newspaper articles). Of these, only 9752 were annotated; the reason for the gap is that some Hebrew words, mainly borrowed but also some frequent words such as prepositions, do not have roots; we further eliminated 168 roots with more than three consonants and were left with 5242 annotated word types, exhibiting 1043 different roots. Table 1 shows the distribution of word types according to root ambiguity.

[3] Only tri-consonantal roots are counted. Ornan (2003) mentions 3407 roots, whereas the number of roots in Arabic is estimated to be 10,000 (Darwish, 2002).

| Number of roots | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Number of words | 4886 | 335 | 18 | 3 |

Table 1: Root ambiguity in the corpus

Table 2 provides the distribution of the roots of the 5242 word types in our corpus according to root type, where $C_i$ is the $i$-th radical (note that some roots may belong to more than one group).

| Paradigm | Number | Percentage |
|---|---|---|
| $C_1 = i$ | 414 | 7.90% |
| $C_1 = w$ | 28 | 0.53% |
| $C_1 = n$ | 419 | 7.99% |
| $C_2 = i$ | 297 | 5.66% |
| $C_2 = w$ | 517 | 9.86% |
| $C_3 = h$ | 18 | 0.19% |
| $C_3 = i$ | 677 | 12.92% |
| $C_2 = C_3$ | 445 | 8.49% |
| Regular | 3061 | 58.41% |

Table 2: Distribution of root paradigms

As assurance for statistical reliability, in all the experiments discussed in the sequel (unless otherwise mentioned) we performed 10-fold cross validation runs a for every classification task during evaluation. We also divided the test corpus into two sets: a *development set* of 4800 words and a *held-out set* of 442 words. Only the development set was used for parameter tuning. A given *example* is a word type with all its (manually tagged) possible roots. In the experiments we describe below, our system produces one or more root *candidates* for each example. For each example, we define *tp* as the number of candidates correctly produced by the system; *fp* as the number of candidates which are not correct roots; and *fn* as the number of correct roots the system did not produce. As usual, we define *recall* as $\frac{tp}{tp+fp}$ and *precision* as $\frac{tp}{tp+fn}$; we then compute $f$-measure for each example (with $\alpha = 0.5$) and (macro-) average to obtain the system's overall $f$-measure.

To estimate the difficulty of this task, we asked six human subjects to perform it. Subjects were asked to identify all the possible roots of all the words in a list of 200 words (without context), randomly chosen from the test corpus. All subjects were computer science graduates, native Hebrew speakers with no linguistic background. The average precision of humans on this task is 83.52%, and with recall at 80.27%, $f$-measure is 81.86%. Two main reasons for the low performance of humans are the lack of context and the ambiguity of some of the weak paradigms.

## 4 A machine learning approach

To establish a baseline, we first performed two experiments with simple, baseline classifiers. In all the experiments described in this paper we use SNoW (Roth, 1998) as the learning environment, with *winnow* as the update rule (using *perceptron* yielded comparable results). SNoW is a multi-class classifier that is specifically tailored for learning in domains in which the potential number of information sources (features) taking part in decisions is very large, of which NLP is a principal example. It works by learning a sparse network of linear functions over a pre-defined or incrementally learned feature space. SNoW has already been used successfully as the learning vehicle in a large collection of natural language related tasks, including POS tagging, shallow parsing, information extraction tasks, etc., and compared favorably with other classifiers (Roth, 1998; Punyakanok and Roth, 2001; Florian, 2002). Typically, SNoW is used as a classifier, and predicts using a winner-take-all mechanism over the activation values of the target classes. However, in addition to the prediction, it provides a reliable confidence level in the prediction, which enables its use in an inference algorithm that combines predictors to produce a coherent inference.

### 4.1 Feature types

All the experiments we describe in this work share the same features and differ only in the target classifiers. The features that are used to characterize a word are both grammatical and statistical:

- Location of letters (e.g., the third letter of the word is *b* ). We limit word length to 20, thus obtaining 440 features of this type (recall the the size of the alphabet is 22).
- Bigrams of letters, independently of their location (e.g., the substring *gd* occurs in the word). This yields 484 features.
- Prefixes (e.g., the word is prefixed by *k$h* "when the"). We have 292 features of this type, corresponding to 17 prefixes and sequences thereof.
- Suffixes (e.g., the word ends with *im*, a plural suffix). There are 26 such features.

### 4.2 Direct prediction

In the first of the two experiments, referred to as Experiment A, we trained a classifier to learn roots

as a single unit. The two obvious drawbacks of this approach are the large set of targets and the sparseness of the training data. Of course, defining a multi-class classification task with 2152 targets, when only half of them are manifested in the training corpus, does not leave much hope for ever learning to identify the missing targets.

In Experiment A, the macro-average precision of ten-fold cross validation runs of this classification problem is 45.72%; recall is 44.37%, yielding an $f$-score of 45.03%. In order to demonstrate the inadequacy of this method, we repeated the same experiment with a different organization of the training data. We chose 30 roots and collected all their occurrences in the corpus into a test file. We then trained the classifier on the remainder of the corpus and tested on the test file. As expected, the accuracy was close to 0%,

### 4.3 Decoupling the problem

In the second experiment, referred to as Experiment B, we separated the problem into three different tasks. We trained three classifiers to learn each of the root consonants in isolation and then combined the results in the straight-forward way (a conjunction of the decisions of the three classifiers). This is still a multi-class classification but the number of targets in every classification task is only 22 (the number of letters in the Hebrew alphabet) and data sparseness is no longer a problem. As we show below, each classifier achieves much better generalization, but the clear limitation of this method is that it completely ignores interdependencies between different targets: the decision on the first radical is completely independent of the decision on the second and the third.

We observed a difference between recognizing the first and third radicals and recognizing the second one, as can be seen in table 3. These results correspond well to our linguistic intuitions: the most difficult cases for humans are those in which the second radical is *w* or *i*, and those where the second and the third consonants are identical. Combining the three classifiers using logical conjunction yields an $f$-measure of 52.84%. Here, repeating the same experiment with the organization of the corpus such that testing is done on unseen roots yielded 18.1% accuracy.

To demonstrate the difficulty of the problem, we conducted yet another experiment. Here, we trained the system as above but we tested it on different words whose roots were known to be in the training set. The results of experiment A here were 46.35%, whereas experiment B was accurate in 57.66% of the cases. Evidently, even when testing only on previously seen roots, both naïve methods are unsuccessful (although method A here outperforms method B).

| | $C_1$ | $C_2$ | $C_3$ | root |
|---|---|---|---|---|
| Precision: | 82.25 | 72.29 | 81.85 | 53.60 |
| Recall: | 80.13 | 70.00 | 80.51 | 52.09 |
| $f$-measure: | 81.17 | 71.13 | 81.18 | 52.84 |

Table 3: Accuracy of SNoW's identifying the correct radical

## 5 Combining interdependent classifiers

Evidently, simple combination of the results of the three classifiers leaves much room for improvement. Therefore we explore other ways for combining these results. We can rely on the fact that SNoW provides insight into the decisions of the classifiers – it lists not only the selected target, but rather all candidates, with an associated confidence measure. Apparently, the correct radical is chosen among SNoW's top-$n$ candidates with high accuracy, as the data in table 3 reveal.

This observation calls for a different way of combining the results of the classifiers which takes into account not only the first candidate but also others, along with their confidence scores.

### 5.1 HMM combination

We considered several ways, e.g., via HMMs, of appealing to the sequential nature of the task ($C_1$ followed by $C_2$, followed by $C_3$). Not surprisingly, direct applications of HMMs are too weak to provide satisfactory results, as suggested by the following discussion. The approach we eventually opted for combines the predictive power of a classifier to estimate more accurate state probabilities.

Given the sequential nature of the data and the fact that our classifier returns a distribution over the possible outcomes for each radical, a natural approach is to combine SNoW's outcomes via a Markovian approach. Variations of this approach are used in the context of several NLP problems, including POS tagging (Schütze and Singer, 1994), shallow parsing (Punyakanok and Roth, 2001) and named entity recognition (Tjong Kim Sang and De Meulder, 2003).

Formally, we assume that the confidence supplied by the classifier is the probability of a state (radical, $c$) given the observation $o$ (the word), $P(c|o)$. This information can be used in the HMM framework by

applying Bayes rule to compute

$$P(o|c) = \frac{P(c|o)P(o)}{P(c)},$$

where $P(o)$ and $P(c)$ are the probabilities of observing $o$ and being at $c$, respectively. That is, instead of estimating the observation probability $P(o|c)$ directly from training data, we compute it from the classifiers' output. Omitting details (see Punyakanok and Roth (2001)), we can now combine the predictions of the classifiers by finding the most likely root for a given observation, as

$$r = argmax P(c_1 c_2 c_3 | o, \theta)$$

where $\theta$ is a Markov model that, in this case, can be easily learned from the supervised data. Clearly, given the short root and the relatively small number of values of $c_i$ that are supported by the outcomes of SNoW, there is no need to use dynamic programming here and a direct computation is possible.

However, perhaps not surprisingly given the difficulty of the problem, this model turns out to be too simplistic. In fact, performance deteriorated. We conjecture that the static probabilities (the model) are too biased and cause the system to abandon good choices obtained from SNoW in favor of worse candidates whose global behavior is better.

For example, the root *&.b.d* was correctly generated by SNoW as the best candidate for the word *&obdim*, but since $P(C_3 = b|C_2 = b)$, which is 0.1, is higher than $P(C_3 = d|C_2 = b)$, which is 0.04, the root *&.b.b* was produced instead. Note that in the above example the root *&.b.b* cannot possibly be the correct root of *&obdim* since no pattern in Hebrew contains the letter *d*, which must therefore be part of the root. It is this kind of observations that motivate the addition of linguistic knowledge as a vehicle for combining the results of the classifiers. An alternative approach, which we intend to investigate in the future, is the introduction of higher-level classifiers which take into account interactions between the radicals (Punyakanok and Roth, 2001).

### 5.2 Adding linguistic constraints

The experiments discussed in section 4 are completely devoid of linguistic knowledge. In particular, experiment B inherently assumes that any sequence of three consonants can be the root of a given word. This is obviously not the case: with very few exceptions, all radicals must be present in any inflected form (in fact, only *w, i, n* and in an exceptional case *l* can be deleted when roots combine with patterns). We therefore trained the classifiers to consider as targets only letters that occurred in the observed word, plus *w, i, n* and *l*, rather than any of the alphabet letters. The average number of targets is now 7.2 for the first radical, 5.7 for the second and 5.2 for the third (compared to 22 each in the previous setup).

In this model, known as the *sequential model* (Even-Zohar and Roth, 2001), SNoW's performance improved slightly, as can be seen in table 4 (compare to table 3). Combining the results in the straight-forward way yields an $f$-measure of 58.89%, a small improvement over the 52.84% performance of the basic method. This new result should be considered baseline. In what follows we always employ the sequential model for training and testing the classifiers, using the same constraints. However, we employ more linguistic knowledge for a more sophisticated combination of the classifiers.

| | $C_1$ | $C_2$ | $C_3$ | root |
|---|---|---|---|---|
| Precision: | 83.06 | 72.52 | 83.88 | 59.83 |
| Recall: | 80.88 | 70.20 | 82.50 | 57.98 |
| $f$-measure: | 81.96 | 71.34 | 83.18 | 58.89 |

Table 4: Accuracy of SNoW's identifying the correct radical, sequential model

### 5.3 Combining classifiers using linguistic knowledge

SNoW provides a ranking on all possible roots. We now describe the use of linguistic constraints to rerank this list. We implemented a function which uses knowledge pertaining to word-formation processes in Hebrew in order to estimate the likelihood of a given candidate being the root of a given word. The function practically classifies the candidate roots into one of three classes: good candidates, which are likely to be the root of the word; bad candidates, which are highly unlikely; and average cases.

The decision of the function is based on the observation that when a root is regular it either occurs in a word consecutively or with a single *w* or *i* between any two of its radicals. The scoring function checks, given a root and a word, whether this is the case. Furthermore, the suffix of the word, after matching the root, must be a valid Hebrew suffix (there is only a small number of such suffixes in Hebrew). If both conditions hold, the scoring function returns a high value. Then, the function checks if the root is an unlikely candidate for the given word. For example, if the root is regular its consonants

must occur in the word in the same order they occur in the root. If this is not the case, the function returns a low value. We also make use in this function of our pre-compiled list of roots. A root candidate which does not occur in the list is assigned the low score. In all other cases, a middle value is returned.

The actual values that the function returns were chosen empirically by counting the number of occurrences of each class in the training data. For example, "good" candidates make up 74.26% of the data, hence the value the function returns for "good" roots is set to 0.7426. Similarly, the middle value is set to 0.2416 and the low – to 0.0155.

As an example, consider *hipltm*, whose root is *n.p.l* (note that the first *n* is missing in this form). Here, the correct candidate will be assigned the middle score while *p.l.t* and *l.t.m* will score high.

In addition to the scoring function we implemented a simple edit distance function which returns, for a given root and a given word, the inverse of the edit distance between the two. For example, for *hipltm*, the (correct) root *n.p.l* scores $1/4$ whereas *p.l.t* scores $1/3$.

We then run SNoW on the test data and rank the results of the three classifiers *globally*, where the order is determined by the product of the three different classifiers. This induces an order on *roots*, which are combinations of the decisions of three independent classifiers. Each candidate root is assigned three scores: the product of the confidence measures of the three classifiers; the result of the scoring function; and the inverse edit distance between the candidate and the observed word. We rank the candidates according to the product of the three scores (i.e., we give each score an equal weight in the final ranking).

In order to determine which of the candidates to produce for each example, we experimented with two methods. First, the system produced the top-$i$ candidates for a fixed value of $i$. The results on the development set are given in table 5.

| $i =$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Precision | **82.02** | 46.17 | 32.81 | 25.19 |
| Recall | **79.10** | 87.83 | 92.93 | 94.91 |
| $f$-measure | **80.53** | 60.52 | 48.50 | 39.81 |

Table 5: Performance of the system when producing top-$i$ candidates.

Obviously, since most words have only one root, precision drops dramatically when the system produces more than one candidate. This calls for a better threshold, facilitating a non-fixed number of outputs for each example. We observed that in the "difficult" examples, the top ranking candidates are assigned close scores, whereas in the easier cases, the top candidate is usually scored much higher than the next one. We therefore decided to produce all those candidates whose scores are not much lower than the score of the top ranking candidate. The drop in the score, $\delta$, was determined empirically on the development set. The results are listed in table 6, where $\delta$ varies from 0.1 to 1 ($\delta$ is actually computed on the log of the actual score, to avoid underflow).

These results show that choosing $\delta = 0.4$ produces the highest $f$-measure. With this value for $\delta$, results for the held-out data are presented in table 7. The results clearly demonstrate the added benefit of the linguistic knowledge. In fact, our results are slightly better than average human performance, which we recall as well. Interestingly, even when testing the system on a set of roots which do *not* occur in the training corpus (see section 4), we obtain an $f$-score of 65.60%. This result demonstrates the robustness of our method.

| | Held-out data | Humans |
|---|---|---|
| Precision: | **80.90** | 83.52 |
| Recall: | **88.16** | 80.27 |
| $f$-measure: | **84.38** | 81.86 |

Table 7: Results: performance of the system on held-out data.

It must be noted that the scoring function alone is *not* a function for extracting roots from Hebrew words. First, it only scores a given root candidate against a given word, rather than yield a root given a word. While we could have used it exhaustively on all possible roots in this case, in a general setting of a number of classifiers the number of classes might be too high for this solution to be practical. Second, the function only produces three different values; when given a number of candidate roots it may return more than one root with the highest score. In the extreme case, when called with all $22^3$ potential roots, it returns on the average more than 11 candidates which score highest (and hence are ranked equally).

Similarly, the additional linguistic knowledge is not merely *eliminating* illegitimate roots from the ranking produced by SNoW. Using the linguistic constraints encoded in the scoring function only to eliminate roots, while maintaining the ranking proposed by SNoW, yields much lower accuracy. Clearly, our linguistically motivated scoring does more than elimination, and actually *re-ranks* the

| $\delta =$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Precision | 81.81 | 80.97 | 79.93 | **78.86** | 77.31 | 75.48 | 73.71 | 71.80 | 69.98 | 67.90 |
| Recall | 81.06 | 82.74 | 84.03 | **85.52** | 86.49 | 87.61 | 88.72 | 89.70 | 90.59 | 91.45 |
| $f$-measure | 81.43 | 81.85 | 81.93 | **82.06** | 81.64 | 81.10 | 80.52 | 79.76 | 78.96 | 77.93 |

Table 6: Performance of the system, producing candidates scoring no more than $\delta$ below the top score.

roots. It is only the *combination* of the classifiers with the linguistically motivated scoring function which boosts the performance on this task.

### 5.4 Error analysis

Looking at the questionnaires filled in by our subjects (section 3), it is obvious that humans have problems identifying the correct roots in two general cases: when the root paradigm is weak (i.e., when the root is irregular) and when the word can be read in more than way and the subject chooses only one (presumably, the most prominent one). Our system suffers from similar problems: first, its performance on the regular paradigms is far superior to its overall performance; second, it sometimes cannot distinguish between several roots which are in principle possible, but only one of which happens to be the correct one.

To demonstrate the first point, we evaluated the performance of the system on a different organization of the data. We tested separately words whose roots are all regular, vs. words all of whose roots are irregular. We also tested words which have at least one regular root (mixed). The results are presented in table 8, and clearly demonstrate the difficulty of the system on the weak paradigms, compared to almost 95% on the easier, regular roots.

| | Regular | Irregular | Mixed |
|---|---|---|---|
| Number of words | 2598 | 2019 | 2781 |
| Precision: | 92.79 | 60.02 | 92.54 |
| Recall: | 96.92 | 73.45 | 94.28 |
| $f$-measure: | 94.81 | 66.06 | 93.40 |

Table 8: Error analysis: performance of the system on different cases.

A more refined analysis reveals differences between the various weak paradigms. Table 9 lists $f$-measure for words whose roots are irregular, classified by paradigm. As can be seen, the system has great difficulty in the cases of $C_2 = C_3$ and $C_3 = i$.

Finally, we took a closer look at some of the errors, and in particular at cases where the system produces several roots where fewer (usually only one) are correct. Such cases include, for example, the word *hkwtrt* ("the title"), whose root is the regular *k.t.r*; but the system produces, in addition, also *w.t.r*, mistaking the *k* to be a prefix. This is the kind of errors which are most difficult to cope with.

| Paradigm | $f$-measure |
|---|---|
| $C_1 = i$ | 70.57 |
| $C_1 = n$ | 71.97 |
| $C_2 = i/w$ | 76.33 |
| $C_3 = i$ | 58.00 |
| $C_2 = C_3$ | 47.42 |

Table 9: Error analysis: the weak paradigms

However, in many cases the system's errors are relatively easy to overcome. Consider, for example, the word *hmtndbim* ("the volunteers") whose root is the irregular *n.d.b.* Our system produces as many as five possible roots for this word: *n.d.b, i.t.d, d.w.b, i.h.d, i.d.d.* Clearly some of these could be eliminated. For example, *i.t.d* should not be produced, because if this were the root, nothing could explain the presence of the *b* in the word; *i.h.d* should be excluded because of the location of the *h.* Similar phenomena abound in the errors the system makes; they indicate that a more careful design of the scoring function can yield still better results, and this is the direction we intend to pursue in the future.

## 6 Conclusions

We have shown that combining machine learning with limited linguistic knowledge can produce state-of-the-art results on a difficult morphological task, the identification of roots of Hebrew words. Our best result, over 80% precision, was obtained using simple classifiers for each of the root's consonants, and then combining the outputs of the classifiers using a linguistically motivated, yet extremely coarse and simplistic, scoring function. This result is comparable to average human performance on this task.

This work can be improved in a variety of ways. We intend to spend more effort on feature engineering. As is well-known from other learning tasks, fine-tuning of the feature set can produce additional accuracy; we expect this to be the case in this task, too. In particular, introducing features that capture

contextual information is likely to improve the results. Similarly, our scoring function is simplistic and we believe that it can be improved. We also intend to improve the edit-distance function such that the cost of replacing characters reflect phonological and orthographic constraints (Kruskal, 1999).

In another track, there are various other ways in which different inter-related classifiers can be combined. Here we only used a simple multiplication of the three classifiers' confidence measures, which is then combined with the linguistically motivated functions. We intend to investigate more sophisticated methods for this combination, including higher-order machine learning techniques.

Finally, we plan to extend these results to more complex cases of learning tasks with a large number of targets, in particular such tasks in which the targets are structured. We are currently working on similar experiments for Arabic root extraction. Another example is the case of morphological disambiguation in languages with non-trivial morphology, which can be viewed as a POS tagging problem with a large number of tags on which structure can be imposed using the various morphological and morpho-syntactic features that morphological analyzers produce. We intend to investigate this problem for Hebrew in the future.

## Acknowledgments

This work was supported by The Caesarea Edmond Benjamin de Rothschild Foundation Institute for Interdisciplinary Applications of Computer Science. Dan Roth is supported by NSF grants CAREER IIS-9984168, ITR IIS-0085836, and ITR-IIS 00-85980. We thank Meira Hess and Liron Ashkenazi for annotating the corpus and Alon Lavie and Ido Dagan for useful comments.

## References

Ken Beesley. 1998a. Arabic morphological analysis on the internet. In *Proceedings of the 6th International Conference and Exhibition on Multilingual Computing*, Cambridge, April.

Kenneth R. Beesley. 1998b. Arabic morphology using only finite-state operations. In Michael Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic languages*, pages 50–57, Montreal, Quebec, August. COLING-ACL'98.

Yaacov Choueka. 1990. MLIM - a system for full, exact, on-line grammatical analysis of Modern Hebrew. In Yehuda Eizenberg, editor, *Proceedings of the Annual Conference on Computers in Education*, page 63, Tel Aviv, April. In Hebrew.

Kareem Darwish. 2002. Building a shallow Arabic morphological analyzer in one day. In Mike Rosner and Shuly Wintner, editors, *Computational Approaches to Semitic Languages, an ACL'02 Workshop*, pages 47–54, Philadelphia, PA, July.

Abraham Even-Shoshan. 1993. *HaMillon HaXadash (The New Dictionary)*. Kiryat Sefer, Jerusalem. In Hebrew.

Y. Even-Zohar and Dan Roth. 2001. A sequential model for multi class classification. In *EMNLP-2001, the SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 10–19.

Radu Florian. 2002. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178. Taiwan.

Joseph Kruskal. 1999. An overview of sequence comparison. In David Sankoff and Joseph Kruskal, editors, *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 1–44. CSLI Publications, Stanford, CA. Reprint, with a foreword by John Nerbonne.

John J. McCarthy. 1981. A prosodic theory of non-concatenative morphology. *Linguistic Inquiry*, 12(3):373–418.

Uzzi Ornan. 2003. *The Final Word*. University of Haifa Press, Haifa, Israel. In Hebrew.

Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems 13*, pages 995–1001. MIT Press.

Dan Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of AAAI-98 and IAAI-98*, pages 806–813, Madison, Wisconsin.

H. Schütze and Y. Singer. 1994. Part-of-speech tagging using a variable memory markov model. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*.

Joseph Shimron, editor. 2003. *Language Processing and Acquisition in Languages of Semitic, Root-Based, Morphology*. Number 28 in Language Acquisition and Language Disorders. John Benjamins.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Yizxaq Zdaqa. 1974. *Luxot HaPoal (The Verb Tables)*. Kiryath Sepher, Jerusalem. In Hebrew.