Stemming Hebrew words

Daniel Khodyrker
daniel.khodyrker@gmail.com

Abstract

A Hebrew word's root (or stem) can be thought of as a word's "pure" meaning, i.e it's entry in the dictionary. Identification of a word's root can aid in meaning extraction, automatic text understanding, automatic hebrew text vocalization[5]. In this project we attempt to automatically extract a Hebrew word's root, given the corpus from which the word is taken.

1 Introduction

The Hebrew language is highly morphologically ambiguous, as the average number of interpretations is 2.7 as opposed to 1.4 for English ([1]). This high ambiguity rate makes Hebrew text analysis very difficult. Hebrew morphology is non-concatenative : words are created in a process of interdigitation - the combination of two morphemes -"root" (or stem) and pattern. The root is a combination of usually three consonants which we shall refer to as "radicals". A word's stem is it's basic pardigam, hence, Hebrew word root extraction might greatly aid in Hebrew text disambiguation and as a result in it's analysis and comprehension. The process of creation of a Hebrew word out of a root and pattern is rather straightforward, but is accompanied by a set of morphological rules based on inflection tables which are too many to manually hard-code. The reverse process, however, is not trivial, as without vocalization (in form of diacritics or "niqqud"), a single word can be mapped to several root+pattern combinations, hence context is required. We wish to automate the process of a word's root extraction. It is natural to also demand an automatic pattern extraction, which is achieved in [5], with a root as **input** and not a target. The task of root and pattern extraction is much more cumbersome and therefore we shall assume a word's pattern as input. A case could be made for a static model, encoding the aforementioned inflection tables, matching a root to a pattern under the relevant inflection table given the word's context. This would be very difficult and error-prone. We shall attempt to solve the problem by means of Machine Learning. We shall address two types of roots : Regular roots which combine with any pattern by simple interdigitation. For instance : the combination of the root r˙$.m with the pattern Nifa'al yields the word Nir$m. Irregular roots which are not simply combined with any pattern, rather undergo changes in some patterns calledd conjugations. These are also called "weak roots" - roots which contain a "weak" letter (a,h,w,x,i,n,&,r) or two of the same consonant. Each pairing of a weak letter with a position within the root yields a different conjugation pattern. Of the weak positions we shall address irregular roots of the form : $R_1 \in[i, w, n], R_2 \in[i, w, h], R_3 \in[h, i], R_2 = R_3$.

2 Prior Work

We draw our baseline from the article [4] which tackles the same problem. We shall attempt a different ML framework, utilize a richer corpus, analyze the performance of the baseline model more finely on the different categories of roots and attempt to further focus the research required for tackling the task of root extraction.

3 Data and methodology

3.1 Machine-learning Framework

In this work we address the problem of root identification as a classification task, in which the features are categorical and the classification target is either a whole root (combination of three radicals) or a single radical. We have examined several frameworks for classification : LogLinear classifier, Support Vector Machine [3], Random Forests [2]. The best results with minimal hyper parame-

ter adjustments were attained by Random Forests with parameters set to the recommendations by the framework used.

3.2 Data and Evaluation

For training and testing, we used a manually tagged corpus taken from MILA - a publicly available collection of morphologically and syntactically tagged Hebrew articles from various sources (property of the Technion, under GNU-GPL). We used only articles from HAARETZ, although several other corpora are available. The data available are a rich, manually generated morphological analysis of sentences, and a full syntactic analysis in the form of constituent parse trees for each sentence. We shall from now on refer to the set of unique word, Part of Speech (POS), root, pattern combinations in the database as the corpus. There are 12122 distinct items in the corpus, of which 10309 distinct word-root combinations and 1186 unique roots - meaning an average ambiguity rate of 9 distinct words per root. Initially, there were 9858 distinct words in the corpus, of which 416 had roots of length other than 3, which constitute less than 1% of size of the corpus (number of unique , therefore we decided these would be disregarded and dropped.

The table 1 shows the number of distinct words per number of roots in the corpus for those words. We can see that the majority of words are not ambiguous, i.e have a single root in the corpus.

1 root	2 roots	3 roots	4 roots
8814	674	45	3
92%	7%	1%	0%

Table 1: Root ambiguity

We split the corpus by root strength : i.e to regular and irregular corpora by the weak root definition given in the Introduction. the table 2 shows the frequency of irregular roots by the criteria which makes them weak. We can see that irregular roots make up almost half of the entire corpus, which justifies the special attention given to this category.

We trained each model separately on each of the different corpora : regular, irregular and combined, each of which we shall split to a training set and a test set, with a test set ratio of 0.2, and evaluate the model's **accuracy** on each of the test sets.

Pardigm	Number	Frequency
$R_1 = i$	363	3%
$R_1 = w$	23	0%
$R_1 = n$	444	6%
$R_2 = i$	467	6%
$R_2 = w$	848	11%
$R_2 = h$	148	2%
$R_3 = h$	1060	14%
$R_3 = i$	20	0%
$R_2 = R_3$	231	3%
Total irregular	3254	44%

Table 2: Weak pardigm frequency

3.3 Feature Design

All of our experiments draw from the following feature pool :

- The word's letters, one-hot encoded : Assuming words no longer than 11 characters, each character position i is translated to a 22-character binary vector, in which entry j is "on" iff the letter i of the encoded word is the j'th letter of the Hebrew (transliterated) alphabet.
- The word's pattern.
- The word's Part of Speech.
- The word's prefix if such exists.

Note : The PoS referenced here is superficial, as in "verb"—"noun"—"Participle" , and not the PoS drawn from a full syntactic parse tree of the sentence from which the word is taken.

4 Classification Methods

4.1 Naive

Our straightforward approach was to train a model with the word's whole root (the whole three consonants) as a target. This obviously leads to a potentially very large domain of up to 22^3 10000 classes. Furthermore, this limits the classes to only those seen in training leading to overfitting. This shall be referred to as "Model A".

4.2 Single radical, naive combination

Another approach we examine is training a designated model for each radical, and combination of the three models by simple concatenation. This limits the domain for a single model to only 22 letters, and allows for generation of unseen roots. However, this approach might lead to "local" optimization for each radical and disregard strong connections between the radicals. This model shall be referred to as "Model B".

5 Results and analysis

5.1 Straightforward

We examine the accuracy of model A trained on each corpus type, evaluated on each test-corpus type:

Train \ Test	Regular	Irregular	Combined
Regular	0.747	0	0.416
Irregular	0	0.65	0.288
Combined	0.721	0.633	0.683

Table 3: Whole root prediction accuracy

As expected, these results indicate that it is particularly hard to model conjugation of irregular roots. It seems most errors in the combined model are from the inability to classify weak roots, however this should be explored further by clustering the misclassifications.

5.2 Individual radical prediction and naive concatenation

We examine the accuracy of model B trained on each corpus type, on each test-corpus type:

Train \ Test	Regular	Irregular	Combined
Regular	0.893	0.04	0.515
Irregular	0.186	0.711	0.419
Combined	0.802	0.722	0.866

Table 4: Naively combined radical prediction accuracy

We also examine the accuracy of each individual radical prediction, using 10 fold cross validation on the train set:

Train \ Radical	1	2	3
Regular	0.948	0.915	0.951
Irregular	0.877	0.820	0.829
Combined	0.905	0.847	0.88

Table 5: Individual radical prediction accuracy

Individual radical predictions perform well, all of which attaining between 82% and 95% accuracy, with a common drop in accuracy for the second radical, and significant ease in prediction of the first. Performance drops when naively combining the individual predictions. It is evident that the regular and irregular models perform well on their corresponding evaluation set, and very poorly on the other set, yielding an overall mediocre performance on the combined evaluation set. This strengthens the dichotomy between the categories and the classification of irregular roots as the challenging target.

6 Conclusion

In the few experiments we ran, it is evident that naively combining single radical prediction to form word's root fairs slightly better than simply predicting the root as whole. It is also clear that the problem of root extraction could be broken down to two optimization tasks : an easier regular root prediction, and a tougher problem of irregular root extraction. The task of predicting each individual root is evidently a very simple one, probably because the target space is very small and maybe uniform for each root category. However, the transition to a meaningful morpheme is not trivial and requires additional linguistic knowledge - this of course hinders the generality we wish to attain in such research, as in the extreme case we can achieve very high accuray by manually encoding as many inflection tables as possible.

7 Future Work

- Analyze the errors of each model, for instance by clustering wrongly tagged words.
- Encode data from the syntactic parse of a sentence a word belongs too, allowing for richer context. This is not at all trivial, as although such syntactic parse trees exist for the corpus we used, it is difficult to correlate between leaves in the parse tree and words as

they appear in the original sentence. For instance ”kshm” (when they) would appear as [ks] [hm] in a constituent parse tree, and as ”kshm” in the morphological analysis.

- Explore prediction effects of single radicals on each other. For instance, having predicted radical #1, attempt to encode it as a feature for prediction of radical 2. Or perhaps a Markovian n-gram approach as described in [4].
- Train two distinct models, one for regular roots and one for irregular, and attempt a combination by choosing the one which predicts with more certainty.
- Utilize suffixes.
- Combine individually predicted radicals using additional morphological knowledge.

8 Code and resources

Our research code is available in Daniel's github. The data we used is freely available in The Technion's MILA project under GPU-GPL.

References

[1] M. Adler. Hebrew morphological disambiguation: An unsupervised stochastic word-based approach. 2007.

[2] L. Breiman. Random forests. 2001.

[3] v. Cortes. Support-vector networks. 1995.

[4] S. W. Ezra Daya, Dan Roth. Identifying semitic roots: Machine learning with linguistic constraints. 2006.

[5] E. Tomer. Automatic hebrew text vocalization. 2012.