Introduction

“Word”, according to Semitic language word formation process, is described as the combination of two morphemes which are the root, and the pattern. The root of a word, is should only consist of three consonants, called the radicals, though they can be longer than this. The pattern on the other hand, consists of vowels and at times, consonants too. The Pattern of a word has a “slot” into which the root is inserted during word formation. So basically, roots are interdigitated into patterns, and then words are formed. The three consonants are respectively inserted into their slots, i.e., the first radical goes to the first consonant slot, the second radical goes to the second slot, and same to the third.

We focus on extraction of Hebrew and Arabic words. There have been programs designed to help in word extraction in the above two languages. However, these programs have largely relied on constructing large scale lexicons which are intensively laborious. Lexicons are morphological analyzers. We therefore provide a learning approach to word extraction that is machine oriented. This approach can be used by people with limited or no linguistic knowledge and helps in identification of roots of scientific words. The approach automated the identification process, which in turn reduces bottlenecks experienced by listing a lexeme’s roots and patterns.

Buckwatter’s Arabic morphological analyzer (2002 software Documentation) ignores the use of patterns and roots in word formation, and instead, uses word stems in identifying lexicons. Identification of a word root is a complex process that needs training and understanding. This is mainly because the Semitic derivation and morphological inflection are complex and difficult to understand, and the orthography is also peculiar and adverse. However, it is very important to first identify the word root and this process can not be ignored.

The Hebrew morphological analyzer provides a root and pattern combination for other languages unlike in Arabic where the root is only important in morphological analysis because dictionaries are written by roots. Roots are important as they explain the etymological processes of a word within a language and other subsequent languages. They also carry word meanings which are however difficult to understand. Therefore information on roots can be used for computational applications. A practical system that extracts roots in both Hebrew and Arabic is available online at: http//cl.haifa.ac.il/projects/roots/index.shtml. The above system helps in both scientific linguistic research and practical applications. It also includes resources on Semitic languages and can extract morphemes that are not contiguous, from their respective surface forms. It also addresses the Semitic languages’ non concatenate morphemes.

Experiments are carried out below to show that using this application is possible even with limited linguistic knowledge, and that it can improves classification results, which would however be very incorrect without the approach.

2. Linguistic background

To offer an example form in the Hebrew roots for word formation, consider g.d.l, k.t.b, and r.s.n as roots, to correspond to patterns haCCaCa, hitCaCCut and miCCaC. Note that “C” is the pattern slots. When the roots are respectively inserted into their patterns, the words formed respectively will be, *hagdala, hitgadlut, migdal, haktaba, hitkatbut, miktab, har$ama, hitra$mut, and mir$am*. Morph-phonological changes take place in each word once the root and pattern are combined. This change is a complex process, as in the above example, in the second pattern, hitCaCCut, if the root consonant is *t* or *d*; then d.r. $ + hitCaCCut brings, out hiddar$ut. This also happens when the first radical is s or $; s.d.r. + hitCaCCut brings histadrut. Semi vowels in a root are assimilated with vowels, e.g., q.w.m. + haCCaCa brings *haqamma*. The semi vowels / root consonants like w or y are always absent in the resulting word form. The Hebrew word formation id complicated for a number of reasons; the Hebrew orthography does not specify most vowels e.g., *a*, and *e* are not explicated, *o* and *u* are not distinguished and many *i* vowels are not specified. Some single letters are used as both vowels and consonants, like w, used as both o and u, and also as a consonant v, I is used as i (vowel), and y (consonant). The Hebrew orthography also dictates that certain language particles like prepositions, coordinating conjunctions, certain subordinating conjunctions and definite articles should all be attached to the words that immediately come after them. Therefore a form like *mhgr* will be read as “immigrant”, *m-hgr,* as “hagar”, and *m-h-gr* ,”foreigner”. The script does not determine if the first m is part of the pattern, roots or is a preposition.

The second reason why the Hebrew word formation is complex is because it has twenty two letters which are all consonants. This means that tri-consonantal roots are twenty two, though they are drawn down to a smaller number by phonological constraints. This is because roots whose first and third radicals are similar are very rare. A compilation of roots from the dictionary and the Zdeqa Paradigm table helps in the estimation of the total numbers of Hebrew roots, which are estimated to be two thousand, one hundred and fifty two(2,152). Most of these roots are regular, but are constrained in weak paradigm, i.e., their roots consonants change in certain patterns. As an example, i. or n will be chosen as the first consonant, w or I will follow, I will be third; and their second and third consonants are identical. Considering a pattern like hCCCh, regular roots like p.s.q forms hpsqh, while irregular root like n.p.l, i.c.g, q.w.m and g.n.n will respectively form; hplh,hcgh,hqmh, and hqnh. Note that in the first and second examples, n or I which is the radical is missing. In the third example, the w radical is missing, and in the last example, one radical is omitted. A form like hC1C2h can have different roots like n.C1.C2, C1.w.C2, C1.i.C2 or i.C1.C2. Most Hebrew and Arabic words do not have roots because they are not formed through root and pattern language morphology. Because of this, the word is either loan words, or short functional and frequent words. Loan word is usually longer than their original Semitic words.

The Hebrew script is very ambiguous, hence very difficult to understand.. With the machine approach, this ambiguity is reduced because lexemes of a word share the same root. Word context must be considered in order to identify its root. As an example, $mnh can be read as “fat” when its root is $.m.n, or “count” when the root is m.n.i. The above example ignore word context, therefore the results lack design.

3. Data methodology

3.1 Machine Learning framework.

We employ the use of SNoW as a learning environment, to identify root through machine learning techniques, together with Winnow as an update rule. SNoW is an online classifier, tailored to help learning in situation where information sources are very large and confusing. It is highly used, in addition to the current classifier, used in NLP. SNoW‘s extensions include the regularization, proper multi-class classification handling, and algorithms. SNoW’s successful use includes speech tagging, information extraction and shallow parsing. It also has three versions for linear algorithms namely, Perceptron, Winnow, and Naïve Bayes.

3.2 Data and Evaluation

To carry out evaluation, a corpus of fifteen thousand (15,000) Hebrew words was aged for the sake of training and testing. Out of this, only nine thousand seven hundred and fifty two were annotated. This is because, in Hebrew, frequent words like prepositions do not follow the root and pattern word formation paradigm, hence, are excluded from the experiment. One hundred and sixty eight (168) roots were further eliminated, to remain with five thousand, two hundred and forty two annotated words. The eliminated roots contained more than three consonants, hence their elimination. The below tables (table 1) shows root ambiguity or word types.

Number of roots	1	2	3	4
Number of word types	4,886	335	18	3

Table 2 shows root distribution of the 5,242 word types.

Paradigm	Number	Percentage
R1=i	414	7.90
R1=w	28	0.53
R1=n	419	7.99

R2=1	297	5.66
R2=w	517	9.86
R3=h	18	0/19
R3=i	677	12.92
R2=r3	445	8.49
Regular	3,061	58.41

R1 is the ith radical for statistical reliability.

Cross evaluation was done ten times for every classification task and the annotated corpus were divided into two sets; the training set, consisting of four thousand eight hundred (4800) words and a test set, consisting four hundred and forty two (442) words. The training set was used to tune parameter (d), and once this was done, the results were reported by training and testing.

An example is a word type, with all the possible roots. The system produces one or more candidates for each example. i.e., for each example, *tp* will be the number of all the correct candidates identified y the systems, fp, the number of candidates with no correct roots, and fn, as roots not produced by the system. Recall, precision, and F score will therefore be defined as; tptpfp , precision as tptpfn and F-score as 2×recall×precision; to get the overall F Score, all words obtained are macro averaged.

Six human subject, who were Computer science graduates, native Hebrew speakers and with no linguistic background were asked to perform the above task, to illustrate how difficult and complex it is. They were asked to identify all the possible roots for the words in a list of two hundred non context words. Their average precision was 83.52%, recall at 80.27% and the F score at 81.86%. This low performance was deduced to be as a result of lack of word context, and the ambiguity of weak paradigms.

3.3 Feature and design

SNoW's main advantage is that it utilizes feature vectors represented, rather than the Boolean vectors. The following features are used to characterize a word;

1. Letter position, i.e., the first, second, or third letter of the word could be a, the word length is however limited to twenty.

2. Letter biagrams, which is independent of its location.

3. Prefixes

4. Suffixes.

3.4 Linguistic resources

This experiment's main objective was to demonstrate limited linguistic knowledge towards Machine learning, to an NLP task. The following resources were used for both Arabic and Hebrew;

A list of roots

List of prefixes and suffixes

Corpora annotated with roots

Process knowledge, i.e., weak paradigms in word formation

The above sources do not however constitute methods for root identification.

4. Naïve Classification Methods

4.1 Direct Prediction

To experiment with simple baseline, classifiers were performed in order to establish a baseline. Two experiments were performed, dubbed A, and B. in experiment A, a classifier was trained to learn roots as single units. The disadvantage of this approach lies on the scarcity of training data, together with the target sets. In this experiment, the results were as follows; precision was 45.72%, recall

44.37% and F score 45.03%, after ten fold validation. The experiment was the repeated using a different organisation of data, and the accuracy was close to 0%.

4.2 Decoupling the problem

In experiment B, the problem was divided into three tasks, and three classifiers were trained privately to take on the task, and learn each root consonant after which, the results were later combined. Targets in all the tasks are twenty two, which is same as the number of Hebrew alphabetical letters. In this experiment, there was sufficient data, unlike in experiment A. All classifiers performed well, though the method ignores interdependence of targets. There was a difference in recognition of the first and third radicals as well as the second one, as expected. The most difficult case was when w, or i, was presented as the second radical.

With the above combination, an F score of 52.84% was achieved. To demonstrate the difficulty of the problem another experiment was carried out, and the results showed that both Naïve methods were unsuccessful.

5. Combining Interdependent classifiers.

Adding linguistic constraints

The above experiments lacked linguistic knowledge, hence the poor performance. It is important that all radicals are included in any inflected form of a word in this model, known as sequential model; there was a slight improvement in the performance of SNoW as it was combined with Linguistic knowledge. The F score was hence, 58.89%.

Sequential combination

A natural approach to improve the above results was to combine SNoW,s outcomes via a Markovian approach. The approach is used in part of speech tagging, shallow parsing, and entity recognition. When the experiments are done with this approach, with more training data, better results are yielded as compared to the naïve methods. Some models include the HMM models, and the PMM model of Punyakanok od Roth. The sequential model is however simplistic and causes poor performance, with an F score of 37.79%. this is probably because the model is biased, and causes the system to abandon SNoW's choices, and instead chooses worse candidates that perform better globally. E.g., SNoW correctly identifies *mqrn* as the best candidate for the root, *q.r.n*, but because of p(R3=r/R2=r) which is 0.066, and greater than p(R3=n/R2=r), which is 0.025, *q.r.r* is instead produces as the root.

Some letters in Arabic and Hebrew cannot appear in a sequence due to phonetic restrictions. For example, if the first radical is *"s",* then it can not be followed by *z,c,*or *e* as radicals.

To provide better results, HMM approach was extended , following the PMM model of Punyakanok and Roth. In an example with the word *w*, where a classifier R1, is already trained,the classifier's predictions are *a1, a2, a3…..,ak*. With confidence scores of *c1, c2 c3*..,ck respectively,(keeping in mind that the maximum value is 22), for each value a1(1<i<k), that are predicted by R1. Another classifier, R2, is run when R1 is a1. then the value of *i* is checked. i.e, *i* runs from 1 to k. this gives the best sum of the two classifiers with confidence measures for R1 and R2 as a1. using the results

of R2, the same evaluation is carried out on R3. the value that maximizes R3's confidence is then selected, by identifying the root that maximizes all the three clasifers, and yields better results.

Learning biagrams

In this method, root biagrams are learned, instead of learning the roots as single units, as in the previous methods. The potential root number is reduced, hence making it simpler and easy to learn and use.

Combining classifiers using linguistic knowledge

A function called "the scoring function" is introduced to estimate the probability of a candidate being the root of a word. This function classifies a candidate as good, bad, or average, depending on its likelihood of being the root. And therefore produces three different values.

Error analysis

With the above experiments, it was clear to understand the main sources of human errors. Humans have root identification problems when the root paradigm is weak, or when the word can be read in more than one way. The above methods for root identification are internationally used, in conjunction with the local classification tasks. Applying constraints in the scoring function improves the results in two ways; the importance of the global inference method is that it improves the global decisions of root identification, and at the same time, improves the local classification tasks. The most dominant constraint an observed in the experiments is that a candidate root can occur in a list of roots, making it difficult to identify three correct roots. The system used in the experiments also exhinited the same problems. Its performance on the regular paradigms was much superior than to the overall performance. Secondly, it was also unable to distinguish between different several roots. The above problems are illustrated by perfoming additional experiments. In one experiment, words with regular roots are tested versus those with irregular roots, and those tha are "mixed", i.e, at least one regular roots. In another experiment, 200 "hard" words were extracted from a corpus. Hard words are those whose root characters are missing, or because of metathesis, their characters are transposed. Some errors noted include; a case where the system produces many roots in which only one is correct., e.g, the word *hmtndbim*, "volunteer", with an irregular root *n.d.b,* the system produces as many as five roots for the same word, e.g, *n.d.b, i.t.d.,d.w.b, i.h.d,* and *i.d.d.*. however in this list, *i.t.d.* and i/h/d should not be produced at all. This is a problem which is very difficult to correct.

The table below shows error analysis for weak paradigms.

Paradigm	F score
R1=i	70.57

R1=n	71.97
R2=i/w	76.33
R3=i	58.00
R2=R3	47.42

Extension to Arabic

As much as Arabic and Hebrew as Semitic Languages have the same morphological systems, roots in Arabic are more difficult to learn compared to Hebrew because of the following reasons;

Arabic has 28 letters with approximately 40 characters, compared to Hebrew's 22 alphabetical letters.

Related to the above point, Hebrew has a higher pattern number, which are almost twice as those in Hebrew.

i and w are the only letters in Hebrew which can intervene between radicals, whereas in Arabic, the letters are so many (y,w, A, t and wA) can all intervene between radicals r1 and R2, while y, w, A, and A can intervene between r2 and r3. in an experiment for learning Arabic roots, a corpus of 31,991 word types was produced. A stand alone classifier was then trained to identify eacg root radical. This was done through; location of letters, suffixes, prefixes, and letter biagrams. The results are presented in the table below. R1, R2. and R3 collums show the results of each of the three classifiers, while "root" colum is the combination of the three classifiers.

Accuracy of Radical identification in Arabic

	R1	R2	R3	Root
Precision	86.02	70.71	82.95	54.08
Recall	80.84	80.29	88.99	68.10
F Score	87.89	75.20	85.86	60.29

The classifiers were later combined using linguistic knowledge on Arabic word formation process. This function check two issues;

If a root candidate is the correct root for a word, then it should occur in the word, or with any of y, w, A, t, or wA.

A candidate cannot be a root of any word in the corpus, if it does not occur in the pre complied root list.

The Arabic results were poorer than Hebrew because of the reasons discussed earlier, raising the probability of wrong identification.

Conclusion

The word formation process is basically a combination of a root, and a pattern, which are the morphemes use in this process. In Hebrew and Arabic languages, these are mostly ignored, and in place, a word stem is used, making them complex and difficult to understand. This difficulty is however not addressed by the available analyzers, facilitating the invention of Machine learning, which facilitates root identification, by combining different methods like the part of speech tagging, entity recognition, and shallow parsing. Unlike the naïve methods previously depended on, this new classifier, called SNoW, makes the whole process easy even for those with limited linguistic knowledge. It reduces the number of radicals to be identified, and through the biagram learning, a classifier is trained to learn the root radicals instead of the roots themselves. This reduces the labour and tiresome process, making the new method most efficient. As it employs the use of a scoring function, applying the scoring function's constraints generally improve the results, and at the same time, improve the local classification tasks.