Hebrew Vowel Restoration With Neural Networks

M. Spiegel and **J. Volk**
Swarthmore College
Computer Science Dept.
{spiegel,volk}@cs.swarthmore.edu

Abstract

Modern Hebrew is written without vowels, presenting a problem for those wishing to carry out lexical analysis on Hebrew texts. Although fluent speakers can easily replace vowels when reading or speaking from a text, there are no simple rules that would allow for this task to be easily automated. Previous work in this field has involved using statistical methods to try to solve this problem. Instead we use neural networks, in which letter and morphology information are fed into a network as input and the output is the proposed vowel placement. Using a publicly available Hebrew corpus containing vowel and morphological tagging, we were able to restore 85% of the correct vowels to our test set. We achieved an 87% success rate for restoring the correct phonetic value for each letter. While our results do not compare favorably to previous results, we believe that, with further experimentation, our connectionist approach could be made viable.

1 Introduction

As would be expected from the lack of vowels, Hebrew text contains a large amount of ambiguous words. Levinger et al. (1995) calculated, using a Modern Hebrew newspaper corpus, that 55% of Hebrew words are ambiguous[1]. This ambiguity can take several forms: different vowel patterns can be used to distinguish between multiple verb or noun forms, as well as between multiple forms of other parts of speech, such as certain prepositions. Vowel patterns also distinguish between two unrelated words in certain cases. As an example of the following, the consonant string SPR[2] can be vowel tagged in one way such that it means "book" and another way such that it means "to count". This consonant string also has four other possible vowel patterns, each with a different translation.

The problem is further complicated by the fact that Hebrew has twelve vowels, designated in our corpus (discussed below) as {A, F, E, ", I, O, U, W., :, :A, :E, :F}. However, the number of vowels can sometimes be simplified by using phonetic groupings. These are groups of vowels for which all the vowels in the group produce equivalent (or near-equivalent) sounds. As will be discussed later, in certain situations it is enough to produce a vowel from the phonetic group of the target vowel, rather than having to produce the exact vowel. We have identified the following phonetic groups: {A, F, :A, :F}, each of which produce, roughly, the sound "ah" and {E, :E}, each of which produce, roughly, the sound "eh".

We believe that there are two main areas to which this work could be applied, each of which demands somewhat different goals. The first is automated speech generation, which, of course, requires vowel restoration. For this task, we would only need phonetic group accuracy because the vowels within phonetic groups all make the same sounds in spoken Hebrew. The second task is the restoration of vowels to Hebrew texts for the purpose of aiding people who are learning the language, either children learning it as their first language or adults. For this task, we would not be able to combine phonetic groups.

2 Previous Work

Previous relevant work falls into two main categories: work done in the field of Hebrew vowel restoration and work done using neural networks to solve lexical problems which could be similar to vowel restoration. Note that these categories are mutually exclusive; to the best of our knowledge, no previous work has been done which has attempted to combine these fields as we are.

This work makes use of a number of performance metrics which are defined as follows: Word accuracy is the percentage of words which have their complete vowel pattern restored exactly. Letter accuracy is the percentage

[1] It is unclear whether this refers to types or tokens.

[2] Throughout this paper we have used an ASCII representation in place of actual Hebrew letters and vowels.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

of letters which are tagged with the correct vowel. W-phonetic group accuracy is word accuracy, allowing for vowels to be substituted for others within a given phonetic group. L-phonetic group accuracy is letter accuracy, allowing for vowels to be substituted within phonetic groups.

In the field of Hebrew vowel restoration, the most recent work can be found in Gal (2002). This paper attempts to perform vowel restoration on both Hebrew and Arabic using Hidden Markov models. Using a bigram HMM, Gal achieved 81% word accuracy for Hebrew and 87% W-phonetic group accuracy. He did not calculate letter accuracy, but we have to assume that it would have been higher than 81%. Gal also used the Westminster Database as a corpus and calculated that approximately 30

Similar work was also done in Yarowsky (1994), where he addressed accent restoration in Spanish and French. He, like Gal, uses statistical methods - decision lists in this case. His descision lists rely on both local and document-wide collocational information. While this task is quite similar to ours, French and Spanish accent patterns are much less ambiguous than Hebrew vowel patterns; Yarowsky cites baseline values in the 97-98% range. Given this baseline, he is able to achieve 99% accuracy.

Gal also cites a Hebrew morphological analyzer called Nakdan Text (Choueka and Neeman 1995) which uses context-dependent syntactic rules and other probabilistic rules. Nakdan Text uses the morphological information that it creates as a factor in vowel placement, meaning that its results are most comparable to ours obtained using morphology tags. Nakdan Text achieved 95% accuracy, but the authors do not specify if this is word accuracy or letter accuracy.

In the field of neural networks, networks have, in a number of past experiments, been applied to part-of-speech tagging problems, as well as used to solve other lexical classification problems. For example, Hasan and Lua (1996) applied neural nets to the problems of part-of-speech tagging and semantic category disambiguation in Chinese. In Schmid (1994), the author describes his Net-Tagger software which uses a connectionist approach to solve part-of-speech tagging. Schmid's Net-Tagger software performs as well as a trigram-based tagger and outperforms a tagger based on Hidden Markov Models. Given these results, it seems likely that a connectionist approach would produce satisfactory results when applied to vowel restoration in Hebrew, given that vowel restoration is closely linked with lexical categorization.

3 Corpus

We used the Westminster Hebrew Morphological Database (2001), a publicly available corpus containing the complete Tanakh (Hebrew Bible), tagged with both vowels and morphological information. We would have ideally used a corpus of Modern Hebrew, but to our knowledge, there are no Modern Hebrew corpora which are vowel tagged. Throughout this paper we have used the codes from the Westminster Database in which ASCII characters are substituted for Hebrew letters and vowels. Normally, Hebrew is written with the vowels positioned underneath the consonant, with each consonant taking either one vowel or no vowels. In the Westminster notation, vowels are positioned to the right of vowels. For example, the string 'RA' represents what would be written in Hebrew as the consonant 'resh' with the vowel 'qamets' underneath it.

3.1 Morphology

As mentioned above, the Westminster Database includes morphological tags. We chose to run our main experiment using these tags as input for our neural network, given that vowel placement is often tied to morphology. If this system were to be applied to a corpus which had no morphology tags, we would assume that the corpus would be first run through a morphology tagger. In addition, we ran our network once without morphological information as a baseline of sorts.

The morphology tagging in the Westminster Corpus is very broad and varied, although it has certain gaps, as mentioned in our data analysis section. There are several layers of tags, each of which provides more specific information. The most basic level includes tags to distinguish between particles, nouns, adjectives, and verbs. The particle tag is further broken down into tags for articles, conjunctions, prepositions, etc. The tags for pronouns, nouns, and adjectives are subdivided by tags for gender, plurality distinctions, 1st/2nd/3nd person distinctions, etc. The verb tags have a wide range of sub-tags, including the standard ones mentioned above, but also including a variety of tags designed to differentiate between different Hebrew and Aramaic[3] verb forms.

4 Implementation

4.1 Parsing the Corpus

The raw data from our corpus is first run through a series of parsing scripts before becoming input for our neural network. First we remove the initial part of the tag which corresponds to the location in Tanakh (book, chapter, verse, etc.). We then run the following series of parsing tools as necessary:

[3] Aramaic is an ancient Semitic language closely related to Hebrew. Portions of the Hebrew Bible are written in Aramaic, although Genesis, the only section we examined, contains only Hebrew.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

• We recombine words which are written as one word in standard Hebrew but which our corpus splits into multiple words for morphological purposes. These are quite often particles which can be attached to nouns as prefixes. These include the conjunction W, the prepositions B,K,L,M, the relative particle $, the definite article H, etc. The combined word has the morphological tags of both of its components. To illustrate the way this works, consider the Hebrew word "R“)$IYT" meaning "beginning." The prefix "B.:" meaning "in" can be added to form the compound "B.:R“)$IYT" meaning "in the beginning"[4]. In the Westminster Database, the two parts of this word are included as two separate words, the first of which is tagged as a preposition and the second of which is tagged as a noun. For our purposes, we recombine these parts into one word which is tagged as both a preposition and a noun.

• The corpus contains certain word pairs which are slightly modified versions of each other and which occupy a single morphological position within the sentence. One of these words is tagged as the Ketiv version (*) and one is tagged as the Qere version (**). For our purposes we have eliminated the archaic and unvoweled Ketiv version[5].

• In some cases, two words are joined together with either a Maqqef (-) or a compound joint (˜). In these cases we have split the word into two words, given that they can both function as independent words.

• While the dagesh (.) could be considered a vowel, we chose not to attempt to restore it. The dagesh is a character which can be used to distinguish between certain consonant pairs. For example, the string 'B' is pronounced like the English consonant 'v', while the string 'B.' is pronounced like the English consonant 'b'. The dagesh is placed inside the consonant, rather than under it, as with vowels, and a consonant can have both a vowel and a dagesh. Thus, using it would force us to modify our system, which is currently designed to handle one vowel per consonant. Furthermore, our system does not need to deal with the dagesh, as its placement always follows a set of simple rules.

• All other extraneous characters are thrown out: accents (ˆ), affix/suffix separators(/), etc.

4.2 Implementing the Neural Network

The neural network consists of an input matrix, an output matrix, and, in some cases, a hidden layer (see Figure 2). We used a standard feed-forward, fully connected network. The input matrix consists of a single word with morphological tags. Before being used as input, the word is stripped of vowels, so only the consonants are used. Each word is represented as a two-dimensional array of 24 by 15 nodes, where 24 is the number of letters in the Hebrew alphabet plus one null character and 15 is the length of the longest word in the corpus. In addition, each has 104 nodes which represent morphological tags, as discussed above. The output matrix consists of a 14 by 15 node array. Each node in the output matrix represents a possible vowel with one additional node corresponding to the consonant having no associated vowel and one additional node corresponding to the null letter. The activated node in each row signifies the proposed vowel for the corresponding consonant.

[4] Technically, a more precise translation would be "in a beginning" or "at first," but "in the beginning" is the generally accepted translation.

[5] During the period when vowels were added to the Hebrew Bible (1st century CE), occasionally grammatical corrections were added to "fix" portions of the text. The original unvoweled text has been preserved as the "ketiv," and the modified voweled text is called the "qere."

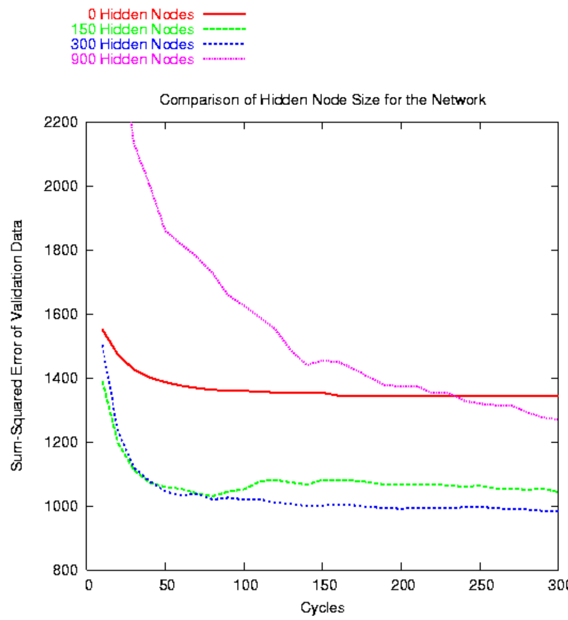

Figure 2: Network Error Based on Hidden Layer Size

Rather than using the entire Bible as input, we chose to use half of the Book of Genesis to cut down on computation time. This consisted of 10306 words, of which 90% were used for training and 10% for testing.

Although previous literature suggested that hidden nodes were not desirable for similar tasks, we ran a number of tests with hidden layers of different sizes, one containing 150 nodes, one with 300, and one with 900.

5 Results

As shown in Figure 2, the neural network with 300 hidden nodes outperformed all the other network configurations. Therefore, all of our results are based on that

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

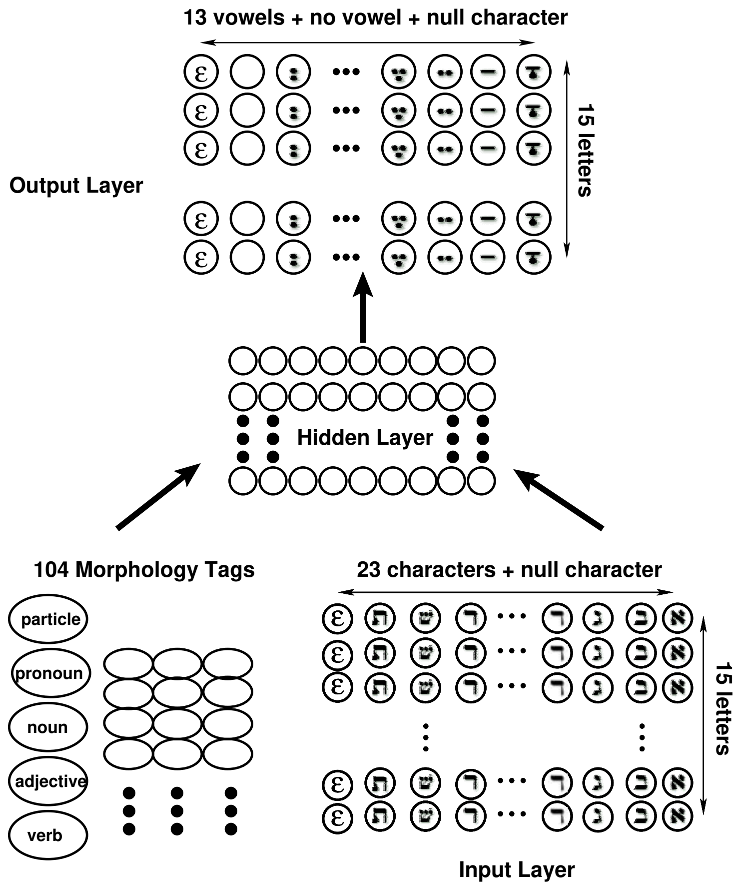

Figure 1: Neural Network Configuration

	A	F	E	“	I	O	U	W.	:	:A	:E	:F	_
Correct Vowels	356	352	216	155	277	224	2	65	381	70	12	0	1193

Table 1: Neural Network correct vowel recognition, with morphology information.

	A	F	E	“	I	O	U	W.	:	:A	:E	:F	_	ϵ
A		16	4	11	16	10	0	0	23	2	0	0	5	2
F	22		17	4	10	14	0	1	33	3	0	0	11	2
E	11	11		10	11	13	0	0	7	3	0	0	4	0
“	7	17	9		8	6	0	0	3	0	1	1	3	0
I	7	7	4	5		1	0	0	10	0	0	0	2	1
O	7	12	7	1	4		0	0	6	0	0	0	13	0
U	1	2	1	0	1	2		0	1	0	0	0	0	0
W.	1	2	0	0	0	0	0		5	0	0	0	19	0
:	12	19	8	4	6	8	1	1		0	0	0	9	1
:A	6	3	0	3	2	5	0	0	8		3	0	0	0
:E	1	0	0	2	0	0	0	0	0	0		0	0	0
:F	0	0	0	0	0	0	0	0	1	0	0		0	0
_	5	3	0	2	3	6	0	1	2	0	0	0		2

Table 2: Neural Network error distribution, with morphology information.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

	A	F	E	“	I	O	U	W.	:	:A	:E	:F	_
Correct Vowels	355	334	230	141	242	201	4	69	363	72	14	0	1190

Table 3: Neural Network correct vowel recognition, without morphology information.

	A	F	E	“	I	O	U	W.	:	:A	:E	:F	_	ϵ
A		22	10	13	15	1	3	1	16	3	1	0	3	2
F	17		19	20	9	12	3	2	34	3	1	2	12	1
E	12	12		7	6	9	0	0	6	4	0	0	0	0
“	11	20	12		10	7	0	0	6	2	0	0	1	0
I	18	10	8	8		6	0	0	14	0	1	1	6	0
O	10	18	11	5	3		1	2	8	2	0	0	13	0
U	1	3	2	0	0	0		0	0	0	0	0	0	0
W.	1	2	0	0	1	0	0		2	0	0	0	17	0
:	26	24	6	4	9	7	1	1		0	1	0	8	0
:A	7	8	3	1	1	2	2	0	2		0	0	1	1
:E	1	0	0	0	0	0	0	0	0	0		0	0	0
:F	0	0	0	0	0	0	0	0	1	0	0		0	0
_	3	3	2	0	1	7	0	8	2	0	0	0		1

Table 4: Neural Network error distribution, without morphology information.

configuration. We achieved 85% letter accuracy, getting 3306 vowels correct out of 3883 total vowels. By merging phonetic groups, we achieved 87% L-phonetic group accuracy, getting 3363 vowels correct. Table 2 shows the error distribution for this experiment and Table 1 shows the distribution of correct results. In these tables, _ represents a consonant with no vowel attached to it, and ϵ represents a null consonant. Further results are forthcoming, including word accuracy, the average percentage of correct vowels per word, and results which look at the type level, rather than at the token level.

We also trained a network on the same data set, but without including morphological information. The hope was that this would let us achieve a baseline, to which our later results could be compared. This experiment achieved a letter accuracy of 74%. Thus, using morphology allowed us to increase our results by 11%. Table 4 shows the error distribution for this no-morphology experiment and Table 3 is the distribution of correct results.

6 Data Analysis

Table 2 demonstrates the errors produced by our network on a vowel by vowel basis. Each entry in the table contains the number of times that the vowel on the row was expected but the vowel on the column was placed instead. Based on this information, several trends emerge. The first is related to the high level of confusion between F and : (33 errors) and vice versa (19 errors). By examining our data, we were able to determine that the vast majority of these errors were due to a morphological feature of Hebrew: prepositions and articles are attached to nouns as prefixes and these preposition and article tags can often be combined to form a single prefix. For example, the prefix “L:-” means “to” or “to a[6],” as in “I went to a party.” It can then be combined with the definite article “HF-” to form the prefix “LF-” meaning “to the,” as in “I went to the party.” Several other examples exist that follow the same pattern; a complete list of prefixal particles is included in the section on parsing the corpus. However, there is no morphological information in the Westminster corpus that would distinguish between these two prefix forms. Given that these prefixes occur in very similar situations, the network was not able to determine the correct vowel to be used in such situations, leading to errors of the type that we found.

We also observed that the vowels “patah” and “qamets,” designated in the Westminster corpus as A and F, respectively, were frequently confused. We believe that this is due to the similarity of these vowels. While more detailed analysis of the Hebrew language would be necessary to determine this for sure, we believe that this similarity exists given that, in spoken Hebrew, these vowels make the same sound (equivalent to the English vowel ’a’). These errors are removed when phonetic group accuracy is calculated, and we believe that using a larger test set might help to minimize these errors. The large amount of errors where A was confused with : presumably result from the combination of these two problems,

[6] Both of these meanings are valid because Hebrew has no indefinite article.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

and would be solved if the other problems were solved.

Note the far-right column in Table 2. These were instances in which the network suggested that ϵ was the vowel to be placed under the current consonant. This, of course, makes no sense, given that ϵ represents the null-consonant vowel, in other words, the vowel that is to be placed with a null consonant. We have no idea why these cases exist, other then that it is simply a quirk of the neural network. Luckily, there are very few of these cases - it is not a major problem.

7 Future Work

There are several areas in which we believe our approach could be improved to hopefully produce results comparable to those obtained through other means. First, we could use more input text. For our results presented here, we chose not to use a larger data set due to a desire to minimize time spent running the network rather than due to a lack of data. This would potentially minimize errors produced due to the network never having seen certain words in the testing corpus.

Second, we could experiment more with hidden layer sizes. We only tried running the three variations mentioned above; perhaps if we had tried others we would have come across one that outperformed the network with 300 hidden nodes.

Third, we could expand our network input to include a broader context of text, perhaps one word on either side of the target word. Given that a bigram model has been shown to work, there is clearly a correlation between word context and vowel placement. However, it is possible that the morphological information that we provide performs a similar role to the role that would be played by context information. In addition, if we wanted to include such information, we would have to find some way of representing it within the context of the network. A problem arises because the network is letter-based, so we would have to figure out how to use words as input. One solution would be to include only the morphological information for the preceding and following words, possibly only the part-of-speech tag. It seems possible that this morphology would be the crucial element in determining the vowel placement for the word in question.

Finally, we could modify the corpus to include tags that distinguish between definite and indefinite article-prefix compounds. The Westminster Database does not include such tags presumably because this distinction does not arise outside of these compounds, given the lack of an indefinite article in Hebrew. This would hopefully solve the problem mentioned earlier that was accounting for a large amount of our vowel placement errors.

In addition to applying these measures to improve our results, a further test of our system would be to apply it to Modern Hebrew text, such as that from an Israeli newspaper archive. The results obtained from such a test would certainly be poor, given the fairly major differences between Biblical and Modern Hebrew. In a Modern Hebrew corpus, we would certainly encounter words that do not appear in the Bible, as well as words which were in the Bible but have since been modified in some way. Our morphological information would also potentially be faulty due to changes in the language over the years. If we wanted to optimize our system for Modern Hebrew we would definitely have to obtain a Modern Hebrew voweled corpus, either by finding one or by creating one by having native speakers tag an unvoweled corpus.

8 Conclusions

From our results, we have to confront the apparent fact that neural networks are not ideal for solving Hebrew vowel restoration. Given that we were including morphological information, we would hope that our results would be comparable to those achieved by Nakdan Text, the only other Hebrew vowel restorer which takes morphology into account. However, our results are a full 10% behind Nakdan Text, assuming that Nakdan is citing a letter accuracy (if they are citing a word accuracy, the difference would be even greater). This data, combined with the Gal results, certainly seems to suggest that a statistical approach to the problem may be superior. However, given the possible improvements to our methods as well as the fact that our results were at least promising, we believe that it might be possible to develop a connectionist system that could perform equally well as, or even outperform, a statistical system.

9 References

References

Y. Choeka and Y. Neeman. 1995. Nakdan Text, (an In-context Text-Vocalizer for Modern Hebrew) *BISFAI-95, The Fifth Bar Ilan Symposium for Artificial Intelligence*

Ya'akov Gal. 2002. An HMM Approach to Vowel Restoration in Arabic and Hebrew *Semitic Language Workshop*

M. Levinger, U. Ornan, A. Itai. 1995. Learning Morpho-Lexical Probabilities from an Untagged Corpus with an Application to Hebrew *Computational Linguistics*, 21(3): 383-404.

Md Maruf Hasan and Kim-Teng Lua. 1996. Neural Networks in Chinese Lexical Classification

Westminster Hebrew Institute. 2001. The Groves-Wheeler Westminster Hebrew Morphology Database, Release 3.5

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College

David Yarowsky. 1994. Decision Lists for Lexical Ambiguity Resolution: Applications to Accent Restoration in Spanish and French *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 1–7
Computer Science Department, Swarthmore College