Word Discrimination Based on Bigram Co-occurrences

Adnan El-Nasan, Sriharsha Veeramachaneni, George Nagy
DocLab, Rensselaer Polytechnic Institute, Troy, NY 12180
elnasan@rpi.edu, veeras@rpi.edu, nagy@ecse.rpi.edu

Abstract

Very few pairs of English words share exactly the same letter bigrams. This linguistic property can be exploited to bring lexical context into the classification stage of a word recognition system. The lexical n-gram matches between every word in a lexicon and a subset of reference words can be precomputed. If a match function can detect matching segments of at least n-gram length from the feature representation of words, then an unknown word can be recognized by determining the subset of reference words having an n-gram match at the feature level with the unknown word. We show that with a reasonable number of reference words, bigrams represent the best compromise between the recall ability of single letters and the precision of trigrams. Our simulations indicate that using a longer reference list can compensate errors in feature extraction. The algorithm is fast enough, even with a slow processor, for human-computer interaction.

1. Introduction

We study letter n-gram statistics as a possible basis for a general method for large-vocabulary word recognition based on finding matching segments between unknown words from a lexicon and known words from a reference list. We assume that co-occurrence of at least one n-gram between the unknown word and a reference word can be determined by comparing their feature representations. Performing this comparison with every reference word results in a *match list* that contains only the reference words that share at least one n-gram with the unknown word. This list is converted into a *match vector* and compared lexically with a precomputed matrix (the *match matrix*) whose entries indicate the presence or absence of lexically identical n-grams between the reference words and the lexicon of admissible words.

Even with perfect n-gram matching, the selectivity depends on the proportion of n-grams in the lexicon that are included in the reference list. A longer reference list can compensate for imperfect features: if the features of a particular n-gram in an unknown word do not match those of the corresponding n-gram in a reference word, they still might match the features of the same n-gram in another reference word. With minimum-distance rather than exact matching, many words can be identified without detecting all of their constituent n-grams. The following questions are of interest:

- What is the relative effectiveness of singlets, bigrams, and trigrams as a function of the size of the lexicon and the length of the reference list?
- How does the discriminating ability of n-gram co-occurrence depend on the size of the lexicon and on the length and content of the reference list?
- What is the degradation resulting from imperfect feature-level comparisons, and how can it be compensated?

2. Prior Work

N-gram statistics have been used for OCR postprocessing since the sixties, when large lexicons could not yet be stored. Raviv introduced Markov models [10], and Shinghal and Toussaint applied the Viterbi algorithm [12,13]. Hull and Srihari quantized n-gram probabilities [8] and explored combining them with dictionary look-up [9]. Suen tabulated the growth in the number of distinct n-grams as a function of vocabulary size, their word-positional dependence, and the influence of the selected corpus [2]. The entropy of n-grams for $n \leq 5$ is computed in [3]. We have not, however, found any study of n-gram co-occurrences between pairs of words.

Feature-level n-gram detection combines some of the advantages of character-level [11] and word-level classification [7]. It is more stable than character-based methods because individual characters or phonemes are often indistinct, and the presence of ligatures or co-articulation facilitates matching longer segments. Like partial-word matching [15,16], feature-level n-gram based recognition is not limited by the size of the vocabulary in the training set, only by that of the lexicon. Full-word recognition, on the other hand, is limited to words with explicit samples in a training set and hence is seldom used

for large vocabularies. Two systems for large vocabularies use Time Delay Neural Networks to avoid segmentation [4,14]. Seni et al. [4], reduces the number of candidate words from the lexicon using a coarse feature description of the unknown and a letter-generating grammar. The NPEN++ system [14] is based on character models built using HMM and arranged in a trie structure to guide the output from a Time Delay Neural Networks.

In [1], we proposed full-word recognition, using bigrams, for words without training samples. Like the widely-used Hidden Markov methods [7], feature-level n-gram detection brings the context into the classification phase rather than relegating it to post-processing. In contrast to HMM, it requires no estimation of model parameters, but only storing a set of representative patterns with their labels.

3. Method

To illustrate the power of n-gram matching, we postulate a (large) lexicon that contains all admissible words. The reference words, for which some feature-level representation is available, are a (small) subset of the lexicon. The unknown (target) word is represented in the same feature space. There is a match routine that detects any common segment between the target word and each of the reference words. The output of the match routine is a binary match vector where "1" means a match, and "0" means no match.

An L x R binary matrix can represent the lexical n-gram matches between the R reference words and the L lexicon words. The entries of the matrix indicate whether a reference word and a lexicon word share at least one n-gram. The percentage of unique rows in the match matrix is called the *selectivity* of the reference set with respect to the given lexicon.

Table I is a small but complete example of a lexicon and a reference list. It also shows the bigram match list for an unknown word generated by the feature-matching process. The alphabet includes initial and trailing blanks (^) that form additional bigrams. This increases the length of the match lists and improves selectivity.

Table II shows the match matrix for the chosen lexicon and reference list.. With the specified reference list, every word in the lexicon corresponds to a unique match vector. For instance, *1011* must be ***people***. But if ***period*** is added to the lexicon, *1011* matches both ***people*** and ***period***. Adding ***tripod*** to the reference list resolves the ambiguity. If only a single row matches the Match Vector exactly, then the target word is identified uniquely (and therefore correctly), otherwise it is ambiguous.

Lexicon	Ref. List	Match List	Match Vector
^consequences^	^Erie^	^Erie^	1
^Erie^	^has^		0
^hair^	^lever^	^lever^	1
^has^	^position^	^position^	1
^lever^			
^nile^			
^pair^			
^people^			
^position^			
^they^			

Table I. This lexicon, reference list, and match list provide sufficient information to identify an unknown word with match vector *1011* as *people*.

	^Erie^	**^has^**	**^lever^**	**^position^**	^tripod^
^consequences^	**0**	**1**	**0**	**1**	0
^Erie^	**1**	**0**	**0**	**0**	1
^hair^	**0**	**1**	**1**	**0**	0
^has^	**0**	**1**	**0**	**0**	0
^lever^	**0**	**0**	**1**	**0**	0
^nile^	**1**	**0**	**1**	**0**	0
^pair^	**0**	**0**	**1**	**1**	0
^people^	**1**	**0**	**1**	**1**	0
^position^	**0**	**0**	**0**	**1**	1
^they^	**0**	**0**	**0**	**0**	1
^period^	1	0	1	1	1
^tripod^	1	0	0	1	1

Table II. Match matrix corresponding to Table I. The match vector "1011" uniquely identifies the eighth row (*people*) in the original matrix (bold). If *period* were added to the lexicon, then 1011 could either be *period* or *people*. But if *tripod* is now added to the reference list, then *period* (10111) can be distinguished from *people* (10110).

Clearly any instance of a lexicon word can be identified uniquely if the corresponding row is unique. Therefore $R = \lceil \log_2 L \rceil$ is a lower bound on the length of the reference list for unique identification of every word in the lexicon.

Identical columns are redundant and the corresponding words can be deleted from the reference list. In fact, any column that is a linear combination of other columns is redundant (over the binary field GF[2], addition is modulo 2). The converse, that all redundant columns are linearly dependent on the other columns, is false, therefore we cannot find by algebraic manipulation a minimal subset of the lexicon to serve as reference words. This problem is equivalent to the NP-complete *Minimum Test Collection problem* [6]. We can find short reference lists with a greedy algorithm that selects at each step the lexicon word that discriminates the most pairs of still-confused lexical words.

4. Experiments

All of our experiments are based on the 1,013,253 word Brown Corpus [7]. This corpus contains 43,300 unique words collected from thousands of published items. The Corpus contains only lower case letters, apostrophes (e.g. *i'm*) and a few quotation marks. Not all of the words are regular dictionary words: there are 454 isolated *s*'s, *mmm* occurs twice, and *mmmm* once (these may be initials or abbreviations stripped of periods). The most common word is *the* (69991). Examples of words that share the same letters are *rate/tear*. Among the rare examples of words that share the same bigrams are *asses/assess*, and *possess/possesses*. The percentages of words with a unique set of letters, bigrams, and trigrams are 48.68%, 99.92%, and 99.99%.

We first compare the selectivity (percentage of unique matches) of singlet, bigram, and trigram length segments. Then we examine the effect of the relative lengths of the reference list and the lexicon. We compare reference lists composed of stop words against lists of less common words). Finally we simulate the effects of feature-level errors on selectivity, and demonstrate the advantage of additional instances of each reference word. Only words from the lexicon are tested. We have not yet investigated methods to reject outliers (words that do not occur in the lexicon).

4.1 Length of Matching Segments

Figure 1 compares the selectivity of unigram (singlet), bigram, and trigram length segments on a lexicon of the 1000 most common words of the Brown Corpus, as a function of the length of the reference list. The reference list is selected from the lexicon according to decreasing word frequency, so most of the words are short.

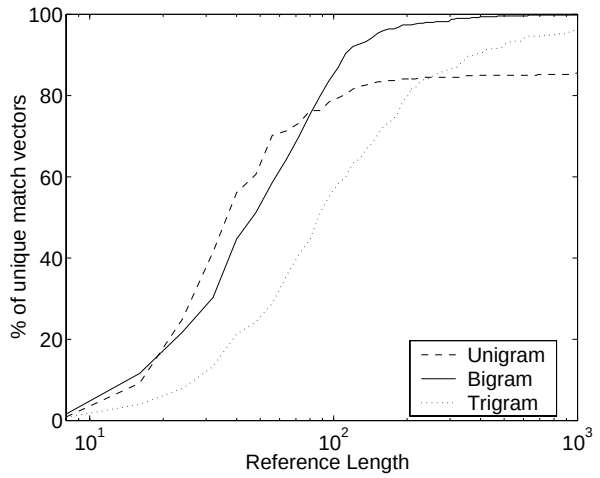

Figure 1. Selectivity as a function of the length of the reference list: unigrams, bigrams, and trigrams (L=1000).

Bigrams appear to be the best option, except for very short reference lists that contain too few bigrams. The unigram graph flattens out because anagrams (like rate and tear) are so common. The trigram selectivity keeps climbing as additional trigrams are included in the reference list, but because there are more trigrams, a longer reference list is required to cover them all. The remaining experiments are all based on bigrams.

4.2 Length of the reference list and the lexicon

Table III shows selectivity for reference lists and lexicons of various lengths. Here too the reference list contains the most frequent words of the lexicon, and the longer lexicons include the shorter lexicons.

L / R	1000	3000	10,000	30,000	43,300
30	27.2	20.9	12.97	9.04	8.01
100	84.0	81.1	75.12	68.15	66.94
300	98.4	98.7	97.58	95.21	95.25
1000	99.8	99.8	99.57	98.63	98.66

Table III. Selectivity for different sizes of Lexicon (L) and Reference List (R).

4.3 Choice of Reference Words

The most common words (stop words) tend to be short and contain only frequent bigrams. Figure 2 compares selecting the reference words from the head of the lexicon with selecting them from the tail. The longer words are more effective: with 50 stop-words, we recognize only 75% of the lexicon, but with the 50 least common words, the selectivity is over 95%. The performance of reference words selected by our greedy algorithm matches is best. The first 40 words selected by the greedy algorithm result in the same number (998) of unique matches as the reference list composed of the entire lexicon.

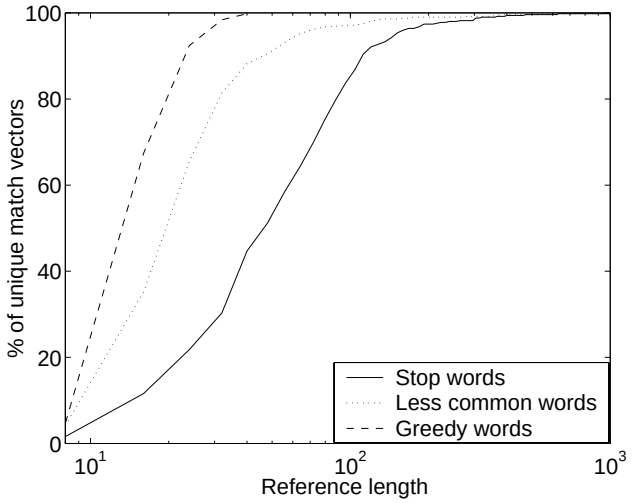

Figure 2. Selectivity with stop words, less common words, and an independent, random match matrix, as a function of the length of the reference list (L-1000).

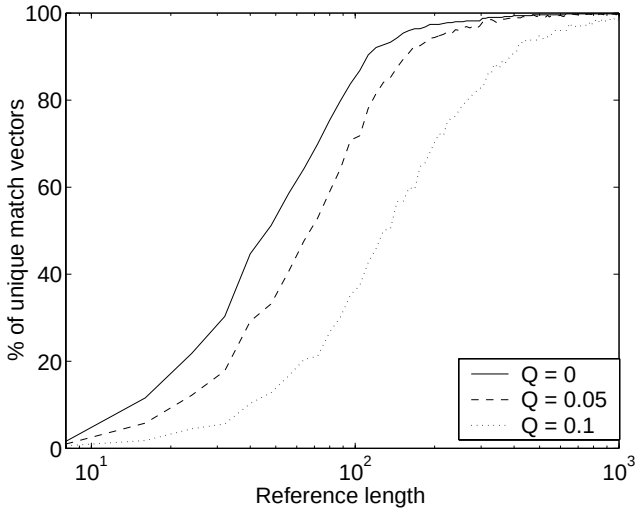

Figure 3. Recognition rate, with feature-level errors, as a function of the length of the reference list.

4.4 Effect of Feature-level Errors

Because we do not expect the feature matching between unknown target word and the words in the reference list to be reliable, we investigate the vulnerability of this scheme to false matches and missed matches. We explore the sensitivity of the method to noise by randomly altering ones in the match vector to zeros, and zeros to ones. This creates the possibility that an unknown word can be misidentified rather than rejected.

We relax the requirement for a perfect match. We will accept a word if the row nearest the target vector (in the Hamming distance sense) is unique. We include multiple instances of each word in the reference list, with the expectation that correct matches against the other instances of the same word may compensate an incorrect feature match against one exemplar.

We call Q the probability of feature-level error. We randomly change, in the Match Vector, 0's to 1's with probability p(e|0), and 1's to 0's with probability p(e|1). These probabilities are determined for each value of Q under the assumption that the match scores, given either a match or a non-match, are both Gaussian with equal variances, and that the prior probability of a match is equal to the proportion of 1's in the match matrix (0.20).

Q	p(e|0)	p(e|1)	σ	M	%R	%E
0	0	0		**1**	13.2	0
0.01	0.0056	0.0277	**0.2245**	**1**	14.6	0.4
				3	13.2	0
				5	13.2	0
0.05	0.0249	0.1502	**0.3337**	**1**	24.6	3.7
				3	14.6	0.5
				5	13.2	0.1
0.10	0.0425	0.3301	**0.4626**	**1**	44.8	18.0
				3	29.8	11.1
				5	23.0	8.2

Table IV. Error (E) and reject (R) rate as a function of the feature-level error and the number of instances (M) of each reference word (L=1000, R=104).

We vary Q from 0.0 to 0.10. Figure 3 shows the rate of correct recognition (100% - %errors - %rejects) when the nearest row in the match matrix is chosen instead of seeking an exact match. For every reference list, the recognition drops sharply between Q=0.05 and Q=0.10. Now we vary the number of instances M of every word in the reference list from 1 to 5. This increases the length of the Match Vector by M. As above, we perturb its elements randomly and select the nearest row of the match matrix. The resulting errors and rejects are shown in Table IV. The results indicate that multiple copies of the reference words alleviate the effect of feature errors.

5. Discussion

The lexical matching can be performed extremely fast by a variety of techniques. With perfect matching, the rows of the match matrix can be presorted for binary search. Comparing each row can be aborted as soon as a mismatch is found, and the search can also be ordered column by column, and halted as soon as a unique row is found. Another fast search method replaces the match matrix by a list of bigrams that co-occur between words in the lexicon and the reference list, and indexes into this list from the Match Vector. For inexact matching, we can use a standard nearest-neighbors preprocessing technique based on the triangle inequality or k-d trees. With the 43,300-word lexicon and a 100-word reference list, we can process 90 words per second without any code optimization on a 350MHz PII.

The only information from feature-level matching used by the current algorithm is whether two words share a segment of minimum length. Using additional information, such as the approximate location and number of matches, will increase the selectivity.

The greedy algorithm drastically reduces the length of the reference list required for unique matches with a given lexicon, but it must be remembered that it also reduces redundancy against feature errors. Furthermore, providing a fixed reference list for a writer or a speaker is likely to be less satisfactory than using the writer's or speaker's own words.

N-gram matching may find application in the recognition of unsegmented words in situations where partial matches between different words can be detected. It could be used with phoneme bigrams for spoken words, and with letter bigrams for word traces in electronic ink and bitmaps of printed or handwritten words. In personalized systems, the system would gradually become more accurate as new samples of words and their labels are added to the reference list. In other words, the system will learn with use.

6. Acknowledgment

We thank Yarmouk University, Jordan, for their financial support.

7. References

1. El-Nasan, G. Nagy, “Ink-Link”, Proc. ICPR, vol. 2, pp. 573-575, 2000.
2. Suen, "N-gram statistics for natural language understanding and text processing”, PAMI-1, 2, pp. 164-172, 1979.
3. E. Yannakoudakis, G. Angelidakis, "*An insight into the entropy and redundancy of the English dictionary*", PAMI-10, 6, pp. 960-970, 1988.
4. G. Seni, R. Srihari, N. Nasrabadi, “Large Vocabulary Recognition of On-Line Handwritten Cursive Words”, PAMI-18, 7, pp. 757-762, 1996.
5. H. Kucera, W. Francis, “Computational analysis of present-day American English”, Providence, RI: Brown University Press, 1967.
6. http://www.nada.kth.se/~viggo/wwwcompendium/node148.html
7. J. Hu, M. Brown, Turin W., “HMM Based Online Handwriting Recognition”, PAMI-18, 10, pp. 1039-1045, 1996.
8. J. Hull, S. Srihari, "Experiments in text recognition with binary n-gram and Viterbi algorithms", PAMI-4, 5, pp. 520-530, 1982.
9. Hull, S. Srihari, R. Choudhari, "An integrated algorithm for text recognition: comparison with a cascaded algorithm", PAMI-5, 4, pp. 384-395, 1983.
10. Raviv, “Decision making in Markov chains applied to the problem of pattern recognition”, IEEE Trans. Inform. Theory, IT-3, 4, pp. 536-551, 1967.
11. K-F. Chan, D-Y. Yeung, “Elastic Structural Matching for On-Line Handwritten Alphanumeric Character Recognition”, Proc. ICPR, vol. 2, pp. 1508-1511, 1998.
12. R. Shinghal, G.T. Toussaint, "Experiments in text recognition with the modified Viterbi algorithm", PAMI-1, 2, pp. 184-193, 1979.
13. R. Shinghal, G.T. Toussaint, "The sensitivity of the modified Viterbi algorithm to the source statistics", PAMI-2, 2, pp. 1181-1184, 1980.
14. S. Jager, S. Manke, A. Waibel, “NPEN++: An On-Line Handwriting Recognition System”, IWFHR, pp. 249-260, 2000.
15. T. Hong, J. Hull, “Algorithms for Post-Processing OCR Results with Visual Inter-Word Constraints”, Proc. ICIP, vol. 3, pp. 312-315, 1995.
16. T. Hong, J. Hull, “Visual Inter-Word Relations and Their Use in OCR Post-Processing”, Proc. ICDAR, vol. 1, pp. 442-445, 1995.