Using Semantic Information from Neural Networks to Detect Context-Sensitive Spelling Errors

Julie Corder
Swarthmore College CS97
Spring 2003

Abstract

This paper proposes a means of using the internal representations of an artificial neural network to represent the semantic contexts in which a word can appear. Once the network has been trained, its hidden layer activations are recorded as a representation of the average context in which a word can appear. This context can then be compared to the contexts in which a word appears in novel text to detect context-sensitive spelling errors. While no significant results are found in the trials described here, several modifications of the system are proposed that might prove promising in future work.

Introduction

Context sensitive spelling correction is the process of identifying words in written text that are spelled correctly but are used in the wrong context. Kukich (1992) discusses various studies that show that between 25% and 40% of spelling errors in typed text result in legal words. This category of spelling errors includes word pairs that are easily mistyped (e.g. "form" and "from"), homophones (e.g. "they're", "their" and "there") and words with similar usages (e.g. "affect" and "effect"). Because all of these errors result in words that are valid, an approach that relies on just a dictionary look-up process will not detect them as spelling errors. Further, Atwell and Elliott [1987] found that 20% to 38% of errors in texts from a variety of sources resulted in valid words that did not result in local syntactic errors. Since dictionary- and syntax-based approaches are not able to detect most context-sensitive spelling errors, semantic clues must be taken into account to determine if the correct word is being used in a given context.

Previous Work

Instead of relying on a comparison to a dictionary of valid words, researchers interested in context sensitive spelling correction must find ways to represent the semantic context in which a word occurs to determine if it is spelled correctly. This approach may be as simple as calculating statistical probabilities of words appearing in certain n-grams, or they may involve greater syntactic and semantic analysis of a corpus. Jones and Martin [1997] report accuracy rates of 56% to 94% for various sets of confusable words using Latent Semantic Analysis. Granger [1983], Ramshaw [1989] and others have used expectation-based techniques. Their systems maintain a list of words that they expect to see next in

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 16–22
Computer Science Department, Swarthmore College

a corpus based on semantic, syntactic, and pragmatic information in the text. If the next word that appears is not on the list of words that were expected, it is marked as a spelling error. In this way, the systems can both detect spelling errors and learn the meaning of new words (by comparing to the meanings of the expected words when a novel word appears).

In all of these cases, though, the researcher must specify the level of information that is relevant to the task. Jones and Martin [1997], for example, specifically tell their system to look at a window of seven words before or after the word in question to build the initial matrices for their analysis. They rely on the researcher to determine how big the window should be. Further, since they look at words before and after the word in question, their method is only useful with complete texts.

These limitations can, perhaps, be avoided by a system that incorporates a neural network. Artificial neural networks (ANNs) are well-suited to a variety of NLP tasks; they can develop their own characterization of which features of a problem are most significant. In addition, simple recurrent networks can store a copy of their previous hidden layer activations. In this way, they are able to build up abstract representations of patterns that occur throughout time [Elman et al. 1996]. Thus, a simple recurrent network should be able to develop an abstract representation of the current context by looking at its internal representation of any number of the words that come before the current word. Given this context, an expectation-based system should be able to predict which words should be able to come next in the text. If the actual next word is not on this list, it should be marked as a spelling error. Further, this system can be combined with a shortest path algorithm to select a word from the list as the correct word, as Wang and Jean [1993] did to correct spelling errors resulting from character merging during OCR. Because this method does not look at future words, it would be useful in applications like word processing systems, where the most recently entered word can be examined for a potential context-sensitive spelling error before more text is entered.

Methods

One of the most limiting aspects of neural networks is the fact that the time needed to train them increases rapidly as the size of the network increases. To test my method, it was necessary to drastically limit the size of the input representation for the network. Consequently, a very small vocabulary represented in localist binary vectors was used to encode the corpus. Vocabulary words were represented by vectors whose length was equal to the number of words in the vocabulary. For a vocabulary of twenty-five words, then, only twenty-five bits were needed to represent any given word. Each vector consisted of exactly one bit that was “on,” and the rest of the bits were set to zero.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 16–22
Computer Science Department, Swarthmore College

Training and testing data came from a part of speech tagged Wall Street Journal corpus. Several categories of words were collapsed into a single "pseudoword" based on part of speech as a means of decreasing the vocabulary size. In particular, the part of speech categories of NN, NNP, NNS, JJ, VBD, VBN, VBZ, DT, and MD were only recorded in the training data by their part of speech class. Further, all punctuation marks were collapsed into the single pseudoword *PUNCT*. Finally, all numerals and number words were changed to the pseudo word *CD* since each number is relatively uncommon in the training text but numbers can usually appear in the same positions in texts. The remaining words, including most function words, were not collapsed into pseudowords at all.

Next, the 25 word vocabulary for this project was selected. The three words to be looked at as possible context-senstive spelling errors were automatically added to the vocabulary. In this trial, that meant that *to*, *too* and *CD* (which contains *two*) were added to the vocabulary. The trianing corpus was then examined, and the 22 most common words were added to the vocabulary. Sentences in the corpus that contained words that were not part of the vocabulary were deleted since they could not be encoded. To ensure that enough data was available about the three target words, sentences that did not include one of those words were also deleted; essentially, a new training and testing corpus was generated by accepting the first sentence that could be encoded that included *to*, then the next sentence that included *too*, and then the next sentence that included *cd* until fifty examples of each had been found. This corpus was encoded as described above and passed to a neural network for training.

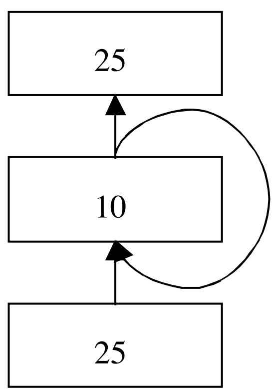

Figure 1: Architecture of recurrent neural network with 25 input and output nodes and a 10-unit hidden layer.

A simple recurrent network was trained on the first half of the encoded corpus. The network had 25 input nodes, corresponding to the 25-bit vector representations of words in the corpus. It also had a 10 node hidden layer and a 25 output nodes. The hidden nodes fed into the output as well as back into the hidden layer. The overall architecture of the network is shown in Figure 1. At each time step, the network's task was to predict the word that would come next.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 16–22
Computer Science Department, Swarthmore College

The network was trained on the entire sequence for 50 epochs using back-propagation.

Once training was completed, the network's representation of the context in which each word appeared was of more interest than the network's prediction of the next work in the corpus. This context representation is stored in the activations of the network's hidden nodes. One final pass through the entire corpus was completed with learning turned off, and the activations of the hidden nodes were recorded at each time step. The hidden layer is the place where the network can establish its own categorization of inputs before having to generate the correct output, so looking at the hidden layer activations gives an illustration of the way that the network is categorizing words internally. The average activation of the hidden layer nodes right before a word was presented was recorded for each word in the training corpus. Because the hidden layer represented the network's representation of the semantic context in which a word would be expected to appear, this vector will be referred to as the "expectation vector" for a given word.

The expectation vectors for all of the words in the vocabulary can be mapped in n-dimensional space; nodes that are closer together in this space can appear in similar semantic contexts, while nodes that are further apart in this space appear in more drastically different semantic contexts.

Context-sensitive spelling errors result, in general, when a word appears in the wrong semantic context. That is, "form" is a spelling error in the sentence "The letter arrived form Cleveland" because it does not appear in a valid location in the sentence (and not because it is an invalid word in English). To detect context-sensitive spelling errors, then, one need only compare the hidden layer activations representing the current context of a network to the average hidden layer activations when the next word is about to appear. If the two are substantially different, the word should be marked as a spelling error, even if it is a valid word in a dictionary.

For each word in the testing part of the corpus, the euclidean distance between the expected context (that is, the average context in which the word appeared in the training corpus) and the actual context (that is, the current hidden layer activations) is calculated. If the current word is one of the target words, then the current hidden layer activation is also compared to each of the expectation vectors of the other target vectors. A large (order of magnitude) difference between the distances for words found in the corpus and the alternative pairings of target words would indicate that the use of the wrong target word somewhere in a novel corpus could be identified by examining the euclidean distance between its expectation vector and the current hidden layer activation.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 16–22
Computer Science Department, Swarthmore College

Results

Unfortunately, there did not seem to be a clear distinction between expectation vectors and the actual hidden layer contexts for different target words. The average euclidean distance between the network's hidden layer activations and the expectation vector for the correct next word was 0.921. The average euclidean distance between the hidden layer activations and the expectation vectors of the other (wrong) target words' expectation vectors was 0.975 (Figure 2). The distance values varied greatly from one word to the next; the standard deviation for both sets of distances was over 0.19, so the difference between the two is clearly not significant.

	Mean distance	Standard Deviation
Correct Expectation Vector	0.921	0.191
Wrong Expectation Vector	0.975	0.196

Figure 2: Average Distance between actual hidden layer activations and the average context for the same (left bar) or different (right bar) target words. Standard deviation is .191 for same-target and .196 for different-target words.

This is a disappointing result. Ideally, the distance to the correct expecation vectors would be significantly smaller than the distance to the wrong expectation vectors. Then the distance between the current hidden layer activation, for example, and the next word typed in an application could be used to predict if there was a context-sensitive spelling error in the current word in the document. Latent semantic analysis could then be used to suggest words whose expecation vectors more closely match the current hidden layer activation. Without a clear distinction between correct and incorrect target words, though, no further analysis can be conducted in terms of application of this process to an actual instance of context sensitive spelling correction. These results do not, however, mean that there is definitely not a way to use an approach of this sort; the following section discusses some of the limitations inherent in this particular study and ways that they might be addressed in future work in this area.

Discussion

One of the most substantial limitations of this project was the small vocabulary size. By collapsing full part of speech categories into a single pseudoword, much of the semantic content that might originally have been available in the text was lost. While this simplified the search space, it also may have resulted in a loss of information that would have been particularly useful to the network in its task.

The solution to this problem is not as simple as just increasing the number of words in the vocabulary and the corresponding number of nodes in the

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 16–22
Computer Science Department, Swarthmore College

representation of each word. For very large networks, the cost of backpropogating error can be prohibitably expensive. Consequently, a localist representation quickly becomes unreasonable as vocabulary size increases.

One possible way to address this problem is through a more distributed representation. Words can be represented, for example, as a binary representation of their unigrams and bigrams. The first twenty-six nodes in the representation vector correspond to the unigrams that may be present in a word. If the current word contains a unigram, then the corresponding node in the input vector is activated. The rest of the nodes correspond to potential bigrams. Bigrams may contain any combination of the alphabetic letters, an end of word marker, and a beginning of word marker. In total, this results in an input vector whose length is 754. For example, in the representation of "two," the nodes representing "t", "w", "o", "_t", "tw", "wo" and "o_" would have values of one, while all other nodes would have values of zero. This representation is drastically larger than the one used for the trials discussed in this paper. It has the advantage, though, of scaling well for extremely large vocabularies. In fact, for a corpus with a vocabulary of more than 754 words, this representation will actually result in a smaller network than will a localist representation. Since 754 words is not a very large vocabulary size for a real-life corpus, a representation of this sort seems essential to further study in this area.

Another possibility is that error was inherent in the process of averaging the context vectors for each word. If the contexts for a given word are clustered in two or more drastically different locations, then averaging them together results in a single vector that is not really representative of any of them. Once the contexts for each step through the training corpus have been gathered, it may be beneficial to conduct some sort of clustering analysis on the vectors. This could avoid the creation of artificial and misrepresentative average vectors that are currently used as the basis for comparison during testing. Unfortunately, only the average contexts for each word were recorded in this experiment, so the presence of this sort of error cannot be confirmed, but it seems like a likely problem that is worth exploring before future work in this area is conducted.

The final possibility is that better results could be found by adjusting the learning parameters of the neural network itself. The epsilon, tolerance and momentum values can all be tweaked. In addition, changes to the number of hidden nodes or the number of training epochs might provide interesting enhancements in the overall system performance. Without any reliable means of predicting what sorts of adjustments would be likely to be beneficial, it was not feasible to test adjustments of these factors in this trial; varying these parameters on a smaller-scale problem would not give a useful indication of how they would affect the larger network or a longer training process, and training a large network takes long enough that running multiple trials of the entire experiment was not possible.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 16–22
Computer Science Department, Swarthmore College

References

Atwell, E. and S. Elliot. 1987. “Dealing with Ill-formed English Text (Chapter 10). In *The Computational Analysis of English: A Corpus-Based Approach*. R. Garside, G. Leach, G. Sampson, Ed. Longman, Inc. New York.

Elman, Jeffrey L., Elizabeth A. Bates, Mark H. Johnson, Annette Karmiloff-Smith, Domenico Parisi, and Kim Plunkett. *Rethinking Innateness: A Connectionist Perspective on Development*. 1996: Massachusetts Institute of Technology

Granger, R.H. 1983. “The NOMAD system: Expectation-based detection and correction of errors during understanding of syntactically and semantically ill-formed text.” *American Journal of Computational Linguistics 9*, 3-4 (July-Dec.), 188-196.

Jones, Michael P. and James H. Martin. “Contextual Spelling Correction using Latent Semantic Analysis.” 1997.

Kukich, Karen. "Techniques for Automatically Correcting Words in Text." ACM Computing Surveys Vol. 24, No. 4, December 1992.

Ramshaw, L. A. 1989. “Pragmatic knowledge for resolving ill-formedness.” Tech. Rep. No. 89-18, BBN, Cambridge, Mass.

Wang, Jin and Jack Jean. "Segmentation of Merged Characters by Neural Networks and Shortest-Path." ACM-SAC 1993.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 16–22
Computer Science Department, Swarthmore College