Personalized Spell Checking using Neural Networks

Tyler Garaas, Mei Xiao, and Marc Pomplun

Visual Attention Laboratory, Computer Science Department,
University of Massachusetts Boston, 100 Morrissey Blvd., Boston, MA 02125-3393, USA
tgaraas@cs.umb.edu, meixiao@cs.umb.edu, marc@cs.umb.edu

Abstract. Spell checkers are one of the most widely recognized and heavily employed features of word processing applications in existence today. This remains true despite the many problems inherent in the spell checking methods employed by all modern spell checkers. In this paper we present a proof-of-concept spell checking system that is able to intrinsically avoid many of these problems. In particular, it is the actual corrections performed by the typist that provides the basis for error detection. These corrections are used to train a feed-forward neural network so that if the same error is remade, the network can flag the offending word as a possible error. Since these corrections are the observations of a single typist's behavior, a spell checker employing this system is essentially specific to the typist that made the corrections. A discussion of the benefits and deficits of the system is presented with the conclusion that the system is most effective as a supplement to current spell checking methods.

Keywords: spell check, neural network, personalized error correction

1 Introduction

Since its introduction into the PC in the 1980's, the spell checker has pervaded just about every level of text-editing provided by modern PCs. As it is, nearly all email and word processing programs in existence today, including web-based programs such as those championed by Google, provide some form of spell checking. Additionally, many blogging and social networking sites have included native support for spell checking via online interfaces, and since version 2.0, spell checking is a standard feature in the web browser Firefox. In fact, the spell checker is one of the most widely recognized program features in existence today. Despite its pervasiveness though, the spell checker's mechanisms are poorly understood and appreciated even less.

In its nascent period, the spell checker's primary functionality was to verify that no misspelled words were present in a block of text. This was usually accomplished by comparing words in the text against an internal list of acceptable words, commonly referred to as the dictionary of the spell checker; if a present word did not match any of the acceptable words, it was declared misspelled. One major deficiency with this method is that it allowed for little or no ability to suggest possible correct-spellings for the misspelled word. In an attempt to circumvent this, algorithms such as the Damerau-Levenshtein distance, which is defined as the number of operations that must be performed to transform one word into another, combined with hashing techniques have been employed. Today's spell checkers are much more sophisticated than their earlier cousins and are now able to overcome problems such as identifying simple grammatical errors and most context-sensitive spelling errors.

Even with the heaps of research performed into developing modern spell checkers, e.g. [1], [2], [3], [4], many problems still exist with current technologies, and while nobody expects a spell checker to be flawless, it is apparent that much work remains. A list of common limitations associated with current spell checkers includes, but is not limited to: inter-language problems, context-sensitive problems, dictionary size problems, and incorrectly flagging acronyms and names as misspelled. In order to better identify the root-causes of these spell checker problems, a couple of these problems are detailed in the following sections.

Context-sensitive problems in the spell checker are most obvious when a homonym of an intended word is used (e.g. there instead of their), but not flagged as an error by the spell checker. While this does not technically constitute a spelling error, since both words are present in the dictionary, if the context in which the word is used is taken into account, a homonym does indeed constitute a spelling error.

Dictionary size problems stem from the striking difference in behavior that a spell checker exhibits simply by the selection of one dictionary size over another. Although it may seem logical that a dictionary should be as complete as possible, most modern spell checkers limit the size of their dictionary to increase their efficiency. For example, while the word holt, which describes the den of a fox or other animal in tree roots or a river bank, is a valid word, it may be found through a frequency evaluation that its presence in a block of text is more likely a misspelling of the words hole or hold. Therefore, its inclusion into the dictionary of a spell checker effectively decreases its efficiency to detect likely misspellings.

In addition to the problems listed above, other problems facing spell checkers also exist, and by now two things should be apparent. One; spell checkers are far from perfect, and two; creating a spell checker that can account for all problems is non-trivial. Possible solutions to each of the problems listed above have been proposed, and when merited, implemented. Context-sensitive problems in particular have been studied extensively, e.g. [1] and [3], and recently the Winnow algorithm has been used to achieve a high degree of accuracy at detecting context-sensitive errors [1]. However, it is unlikely that an amalgamation of algorithms that solve individual problems will ever be capable of fixing all the problems associated with current spell checkers. Instead, it is likely that a new approach to spell checking will need to be taken to solve the host of problems that exist.

In this paper we suggest a new system for detecting misspelled words, which we refer to as the PENN system (Personalized Error correction using Neural Networks). In particular, we use a neural network that is trained using observations of the specific corrections that a typist makes. Corrections that are made enough times are characterized as possible errors, and the word used to replace the errant word is then defined as the corrected-word associated with the specific error. If the typist makes the same error again, the program subtly suggests the corrected-word that is associated with the error. Since the actual behavior of the typist is used to determine potential errors, the problems associated with traditional spell checking methods do not apply. In the following section an overview of the procedure for this system is given, which is then followed by a theoretical discussion of the specific benefits and deficits that are associated with this approach.

2 PENN System Overview

Since the behavior of the typist is used to determine potential errors, a means must be in place to determine the intention of the typist at any given moment. In the present approach, we use the simplistic state machine shown in Fig. 1 to establish the intent of the typist.

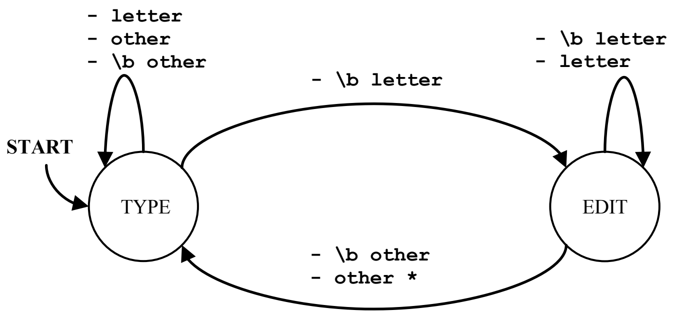

Fig. 1. The finite state machine used to determine when a correction is made. *Letter* and *other* indicate that a letter or something other than a letter was typed, and *\b letter* and *\b other* indicate a backspace was performed and that the character deleted was or was not a letter respectively.

As indicated, the typist starts in the TYPE mode, and will remain in this mode until an actual letter is deleted, which will transition the typist from TYPE mode to EDIT mode. EDIT mode indicates that a correction is being performed by the typist. If, while in EDIT mode, the typist deletes something other than a letter, no correction is made and the typist transitions back to TYPE mode; this indicates that the typist has deleted a full word, which does not constitute a correction in which the system is interested. If the typist types a letter or deletes a letter while in EDIT mode, no transition will be made and the user will remain in EDIT mode with the updated word. However, if the typist types something other than a letter, an indication is made to the system that the typist has performed a correction, the initial word is stored as the erroneous word, the new word is stored as the corrected-word, and finally, the typist is transitioned back to TYPE mode. This simple state machine is sufficient to control the basic spell checking system that we demonstrate in this paper, but a more sophisticated system would need a correspondingly more sophisticated state machine.

The state machine in Fig. 1 functionally defines the corrections that are made while a typist is typing, and once the erroneous word and corrected-word have been defined, both can fed both into the neural network of the spell checking system. Specifically, we use a simple feed-forward neural network that is progressively trained using the erroneous word as input for the network and the corrected-word as the target of the network. To facilitate the input and output of the network, the erroneous and corrected words are transformed into a 20 x 27 binary array as shown in Fig. 2.

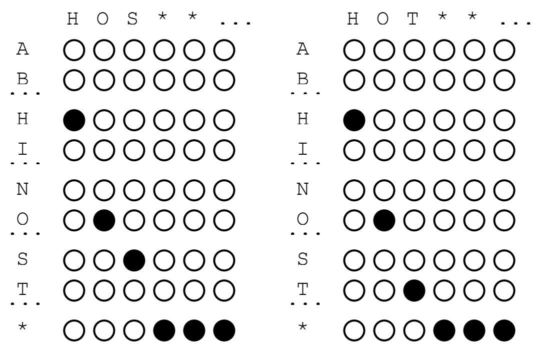

Fig. 2. The transformed erroneous word (*left*), hos, and corrected-word (*right*), hot, for use in the neural network. *Filled circles* and *unfilled circles* represent the values 1 and 0 respectively for the network, and asterisks represent null letters.

The actual choice for the neural network that is used is rather arbitrary as long as it fulfills the following criteria.

1. The network is able to associate an erroneous-word with a corrected-word (referred to as a correction-pair) with a consistent, minimal amount of training.
2. Previously learned correction-pairs are unaltered by the subsequent training of new correction-pairs that do not share the same erroneous word.
3. Correction-pairs with the same erroneous word can compete for activation based upon the number of times each correction has been made.

3 Evaluation of the PENN System

The use of a neural network trained from the observed behavior of the typist marks a large deviation in methodology to spell checking from traditional methods and its characteristics may not be readily apparent. Therefore, some of the more stark changes are listed below.

1. Initially, no words will be flagged as misspelled.
2. Words that are misspelled and corrected enough times are characterized as possible errors in the system.
3. Different words that are misspelled in the same way (e.g. if hit and sit are both commonly misspelled as bit) will compete for the suggested position.

4. Misspelled words need not even be words. Acronyms for example can be flagged as possible errors if they are corrected enough times (e.g. USX to USA).

5. Misspellings can be words themselves (e.g. there to their).

6. Misspellings are specific to the person that made them. This is a unique benefit of the system since many of thee errors people make are not made by most others (e.g. adding an extra ‘e’ to “the”). It is therefore important that programs employing this system have user profiles to ensure that trained neural networks are used only by those that they were trained from. Fortunately, this functionality already exists in most web-based software that employs spell checkers.

By using the typists’ actual behavior, the current method is able to circumvent many of the shortcomings associated with the traditional methods of spell checking. Specifically, those detailed in section 1, context-sensitive problems and dictionary size problems, can both be overcome for the most part by using this system. Still, the example system is far from perfect and even faces new problems that are not present with traditional methods. Some of the problems associated with the example system as well as possible solutions are given below.

1. Initially, the personalized method misses *all* misspelled words. This problem can be addressed by using the personalized method as a supplement to traditional spell checking methods.

2. Corrections made to misspellings that are also words (e.g. there to their) may introduce more false-negatives to flagging results. This could be handled by devising a system to include contextual information about the correction as additional input to the neural network, which would provide true context-sensitive flagging.

Acknowledgements. This work is funded in part by a GAANN (Graduate Assistance in Areas of National Need) Ph.D. scholarship awarded to Tyler W. Garaas by the U.S. Department of Education.

References

1. Golding, A. R. & Roth, D. (1999). A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3), 107-130.
2. Damerau, F. J. (1964). A technique for Computer Detection and Correction of Spelling Errors. *Communications of the Association for Computing Machinery*, 7(3), 171-176.
3. Mays, E., Damerau, F. J., & Mercer, R. L. (1991). Context based spelling correction. *Information Processing and Management*, 27(5), 517-522.
4. Mangu, L. & Brill, E. (1997). Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning.* Morgan Kaufmann.