

TreePredict

Improving text entry on PDA's

Fredrik Kronlid

Undergraduate Student

Computational linguistics, Göteborg University
cl5fkron@cling.gu.se

Victoria Nilsson

Mobile Informatics Group, Viktoria Institute
Box 620, SE-405 30 Gothenburg, Sweden
victoria@viktoria.informatics.gu.se

ABSTRACT

In this paper we describe how an improved word prediction implemented on a PDA can make it easier for users to enter text. The resulting predictions are a result of trigrams using POS-tags (Part Of Speech). The first two parts of the trigrams are POS-tagged, and the last part is extended into a ternary tree, using information from the trigrams to narrow the search. With an improved predictions system, the users are more likely to trust the system, find it improves their ability to enter text with less keystrokes. It is also likely that they will use the prediction feature more actively when they perceive that it is useful to them.

Keywords

Word prediction, ternary trees, hand held device, keystrokes.

INTRODUCTION

To enter more extensive notes on a handheld computer, is to most people a tiresome task [5]. In cases when a user is confronted with such a word prediction system is valuable since it lessens the burden entering characters. Most handheld computers have this service added, with the intent of helping the user enter larger texts. Unfortunately most of these prediction systems, like the one found in Microsoft® Windows CE, are based on statistics, not considering linguistic rules, hence sometimes suggesting poor predictions to the user. If the prediction system were implemented so that it also took linguistic consideration, the resulting predictions would most likely prove to be more useful.

TreePredict

Since words occur in certain combinations in texts, some of the combinations are more frequent than others [3]. One method to acquire information about which combinations are more frequent, is to gather statistics from collections of tagged text (corpora). This statistical information can then be used by a prediction system. We implemented a method, **TreePredict**, that was oriented on linguistics rather than

hardware. There are several approaches to analyzing text with the purpose of building a prediction system. One of these is to use n -grams (an ordered string with n elements). We choose to implement tri-grams, giving us the two previous words, and the word predicted. When searching for a trigram, we conducted the search based on the input data (two previous words) POS-tags. When a string such as “concerned(*adj.*) with(*prep.*)” is entered, the database is

searched for possible trigram completions of the string with the most likely next word. All trigrams starting with the POS-tag string “adj-prep” are searched for and, all words corresponding to the tags possible for the third position in the trigram is searched for highest rank. When the user starts refining the resulting predictions, if he is not



to enter characters in the word, some words do not have the required string of characters. We not only allowed the search to be for words but also for possible next characters. So that the first character of the predicted word could be correct, but not the rest of the word, giving the user the opportunity to choose that character as a prediction, giving new search information by entering following letters. The search is conducted in ternary trees [1], making the search speedier even when large amounts of data are used.

PRESENTING THE PREDICTIONS

The resulting predictions are presented in four boxes, placed in a row above and integrated into the keyboard. Placing the predictions here makes them a natural part of the “typing” interface, not disturbing the user with “pop-ups” nor placing the predictions too far from either the keyboard or the document where the text is being written [2,4]. The predictions are presented to the user as additional keys, although the keys are dynamical and change as the

text the user changes. The first rectangle presents the next character, i.e. the most likely character computed from the present input. The three other rectangles presents possible completions of the word (all of the string including present data). Those predictions represent three different methods of searching for a completion, and so, some method can show no predictions, while some do.

Experiences from Initial Testing

The predictions were first implemented without an ending blankspace, but statistics later on revealed that while the prediction were the correct word stem, the ending either needed to be manipulated or have a different ending added to it. Users add a blankspace after a predicted word was less efficient than forcing them to add them when the predictions were not in its entirety the desired. Also we found it useful to implement the system so that when the word stem was correct, i.e. “imp-lying” but a different ending had to be added i.e. “imp-roving”, the correct part of the word (imp) was still remembered by the system, not causing the system to predict new words from the wrong input (implying *that*). We proposed that this would be more efficient to give the user the opportunity to change the ending of a word this way, rather than assuming that it is always a perfect prediction. While the system at times does not predict the correct word, the prediction can be used if the stem is correct. Since the system remembers the stem while the user manipulates the end, the correct word will be stored by the system and get higher prediction ratings.

Single Character Prediction

This system presents the user with three word or “part of word” predictions, resulting from different searches. It is possible that the user will perceive this as a rating of the predictions, and possibly it would be better that only one of the currently implemented methods was implemented, presenting rated predictions. However we believe that the single character prediction could be kept since that will help keep the users attention close to the prediction area, not forcing her to shift the attention to the keyboard. Presenting single characters this way, could also improve the users impression of the system making them more accustomed to using it, thus improving their willingness to using it.

USER BENEFITS

This system presents to the user predictions that take linguistics into considerations. The predictions that are presented to the user are nearly always likely in the current context, and even if the predictions are not the correct ones, the user still gets to manipulate them without the system losing track of what the input for the prediction is. This gives the user the opportunity to choose to keep part of a prediction, using few keystrokes to manually enter the correct part of an otherwise incorrect prediction, lessening the strain on the user when entering large texts on a PDA. Allowing the user to use part of a prediction, while not making the system assume that a new word is begun, gives

the system the benefit of adapting to the users choice of vocabulary more rapidly than other systems. To have a system that gives linguistically logical predictions could give the user better confidence in using the predictions suggested by the system. When the user has greater confidence with the system it is also more likely that the user will also use the prediction system more frequently and not perceive it as something that steals the attention of the user when popping up within the text area, that only occasionally proves useful. When the predictions presented, while not correct, are close to the desired outcome, this could ease the burden on the user of finding usable words.

CONCLUSION

We proposed that implementing a predictions system that considered linguistics when “calculating” predictions was better than present systems that only considers statistics. While the system did not predict the correct word at all times, good alternatives are presented to the user.

FUTURE WORK

The work described in this paper needs to be tested to confirm our initial findings. We would like to run our system on different platforms and make a few improvements on the presentation of predictions as well as the performance of the predictions. Some work has to be done to make the predictions more efficient in timing and processing.

ACKNOWLEDGMENTS

We thank Staffan Björk at the PLAY studio, Interactive Institute, who gave us the opportunity to work in the *PowerView* project, and Åsa Wengelin, Department of linguistics, University of Göteborg, for their many helping and insightful comments during our work.

REFERENCES

1. Bentley, J.L., and Sedgewick, R. Fast algorithms for sorting and searching. *Proceedings to 8th ACM-SIAM Symposium on Discrete Algorithms*, pp. 360–369, 1997.
2. Björk, S., Redström, J., Ljungstrand, P., and Holmquist, L.E. POWERVIEW: Using information links and information views to navigate and visualize information on small displays. *Handheld and Ubiquitous Computing*, Springer Verlag, pp. 46-62, 2000.
3. Masui, T. POBox: An efficient Text Input method for Handheld and Ubiquitous Computers. *Lecture Notes in Computer Science (1707), Handheld and Ubiquitous Computing*, Springer Verlag, pp. 289-300, 1999.
4. McCoy, K.F., Demasco, P., Pennington, C.A., Luberoff Badman, A. Some interface issues in developing intelligent communication aids for people with disabilities. *Proceedings of the 1997 international conference on Intelligent user interfaces*, pp. 163–170, 1997.
5. Kristoffersen, S. and Ljungberg, F. Designing Interaction Styles for a Mobile Use Context. *Proceedings of International Symposium on Handheld and Ubiquitous Computing (HUC 99)*, pp. 281-288, 1999.