Improving SMS Usability Using Bayesian Networks

Manolis Maragoudakis[1], Nikolaos K. Tselios[2], Nikolaos Fakotakis[1], and Nikolaos M. Avouris[2]

[1]Wire Communications Laboratory
{mmarag,fakotaki}@wcl.ee.upatras.gr

[2]Human-Computer Interaction Group
Dept. of Electrical & Computer Engineering
University of Patras, 26500 Patras, Greece
{nitse,n.avouris}@ee.upatras.gr

Abstract. During the last years, the significant increase of mobile communications has resulted in the wide acceptance of a plethora of new services, like communication via written short messages (SMS). The limitations of the dimensions and the number of keys of the mobile phone keypad are probably the main obstacles of this service. Numerous intelligent techniques have been developed aiming at supporting users of SMS services. Special emphasis has been provided to the efficient and effective editing of words. In the presented research, we introduce a predictive algorithm that forecasts Greek letters occurrence during the process of compiling an SMS. The algorithm is based on Bayesian networks that have been trained with sufficient Greek corpus. The extracted network infers the probability of a specific letter in a word given one, two or three previous letter that have been keyed by the user with precision that reaches 95%. An important advantage, compared to other predictive algorithms is that the use of a vocabulary is not required, so the limited memory resources of mobile phones can easily accommodate the presented algorithm. The proposed method[1] achieves improvement in the word editing time compared to the traditional editing method by a factor of 34.72%, as this has been proven by using Keystroke Level Modeling technique described in the paper.

1 Introduction

Throughout the past decade, mobile telephony has boosted wireless communication to a high, most respectable level of public acceptance. Although we usually consider mobile phones as speech input and output devices, novel technologies such as SMS messaging, mobile chat and WAP, have been presented in an obvious attempt to transform the mobile phone into a hyper-node of incoming and outgoing information. As an example, Sonera, Finland's largest teleoperator reports a six-fold increase in text message [13]. Moreover, GSM association revealed that more than 10 billion of messages per month were sent by the end of 2000. However, the basic problem in

[1] This corresponding method is protected under copyright law.

I.P. Vlahavas and C.D. Spyropoulos (Eds.): SETN 2002, LNAI 2308, pp. 179–190, 2002.

mobile phones and pervasive devices in general is input, where the physical dimensions of the devices obstruct the user. User input is a crucial issue concerning mobile devices since there are numerous applications that take it for granted. SMS communication is now one of the most popular features of cellular phones. Myriads of messages are exchanged throughout the world in a growth rate that approximately doubles every year[2]. Apart from messages, input is also of great importance in mobile chat services, a new technology that aims to create mobile chat rooms as well as in WAP pages where the user forms a query for search, fill out forms.

Buchanan et al., [1] report a study concerning mobile phone users complaining about the difficulty of accessing the phone's functions using the key pad. Buchanan et al. [1] carried out extensive analysis to detect reasons for poor subjective users' satisfaction and found that the number of key presses to access all the menu options was 110, while the average number of key presses to access a function was 8.2. These excess numbers, provide a clear idea that mobile input design is of great importance, not only for quick accessing of the phone's menus but for editing text messages as well. Clearly, the problem exists because first, mobile handsets were anticipated as devices to make and receive calls, but actually transformed to complex information appliances delivering a variety of services to the user [7]. Moreover, physical limitations of devices such as tiny keypads and small screens with low resolution further intensifies the problem reducing the possibility for the users to build solid conceptual models of interaction with them.

For the present work, we focus on this issue of mobile input usability, study the existing methodologies and propose a novel, Bayesian approach that appears to improve typing speed without having to incorporate any linguistic information or dictionary at all. In order to determine the efficiency of the new interaction dialogue proposed, we have evaluated its efficiency using Keystroke Level Model as well as by building a software prototype for simulating preliminary real world experiments.

This paper is organized as follows: First, we briefly present current status in mobile usability research area. Subsequently, we present Bayesian networks upon which our proposed alternative text input method is based, followed by predicting text sequence algorithm description. Subsequently, keystroke level model of interaction is presented and alternative text input dialogues are compared. Finally a software prototype emulator is presented used to preliminary evaluate efficiency of the proposed method.

2 Text Entry Usability in Mobile Phones

As mentioned previously, usability in mobile devices still remains a subject under extensive debate [12]. Due to the fact that our work concerns mobile phones input, we shall discuss usability under this perspective. In a mobile phone, the letters of an alphabet have to be mapped onto a nine-key pad. As a consequence, this means that three or more letters have to be grouped in one single key. Due to that reason, usually

[2] Source: GSM World Association (www.gsmworld.com)

more than one keystroke required in order for a user to access and enter a letter. Nowadays, two alternative dialogues have been established in order to assist the user in editing a message.

The simpler, yet widely acceptable, which from now will be referred as STEM (Standard Text Entry Method), approach requires tapping the corresponding key as many times as needed to appear on screen for a letter to be entered. The basic disadvantage of multiple keystrokes is the lack of speed. However, as previously described, this lack of speed influences positively the need for user confirmation. So, the user does not have to pay any attention to the mobile phone screen. Another problem appears when typing two letters that lie in the same key. The most common solution is the introduction of a time delay between two taps of the same key, in order to verify that the user wants to type two letters from the same group or one letter by multiple taps. This obviously further deteriorates the message editing speed. Additionally to poor task execution time provided by this method, extensive effort in terms of keystrokes is required from the user to complete typing a message.

Another, similar approach is the two-key input method, in which a user specifies a character by pressing two keys. The first key represents the group of letters (e.g key 2 for A, B or C) and the second disambiguates the letter by selecting its place in the group (e.g key 1 would select A). Studies by Silfverberg et al., [13] have depicted that although two-key is very simple, it is not efficient for Roman characters, since there is great loss of speed by moving between the two keys. That is probably the main reason why this method is not popular among users. Note however that it is very common for typing Katakana characters.

Among the lexicon-based methods, the most popular is called T9©, developed by Tegic©, and uses a dictionary in order to deal with letter disambiguation. More specifically, the user presses the key in which the desired letter lies, only once. By the time a word is completed, which means that a space was entered, the system is trying to output the most probable word that corresponds to the key sequence that the user provided. If the guessed word is incorrect, then using a special key the system outputs a pool of other words that also correspond to the specific key sequence. This method significantly reduces editing speed but requires user attention and since it is based on a lexicon, it cannot efficiently handle unknown or shortened words, slang, names etc., heavily used in mobile text messaging [8]. Another important drawback of T9 is the poor feedback during the process of typing a word. There are times that letter disambiguation occurs at the latter characters of a word, so until then, the user may see a totally different set of characters, a phenomenon that obviously results in user confusion due to reduced sense of progress towards user's text entry goal.

In the following section, we shall provide some fundamental background concerning Bayesian networks theory, which we used in order to obtain knowledge about the probabilistic relations of letter sequences. This information contributes to the effective disambiguation of grouped letters according to the approach presenting in the following.

3 Bayesian Belief Networks

A Bayesian Belief Network (BBN) is a significant knowledge representation and reasoning tool, under conditions of uncertainty [9]. Given a set of variables D = $<X_1, X_2 \ldots X_N>$, where each variable X_i could take values from a set $Val(X_i)$, a BBN describes the probability distribution over this set of variables. We use capital letters as *X*,*Y* to denote variables and lower case letters as *x*,*y* to denote values taken by these variables. Formally, a BBN is an annotated directed acyclic graph (DAG) that encodes a joint probability distribution. We denote a network *B* as a pair $B=<G,\Theta>$, [11] where *G* is a DAG whose nodes symbolize the variables of *D*, and Θ refers to the set of parameters that quantifies the network. *G* embeds the following conditional independence assumption:

Each variable X_i is independent of its non-descendants given its parents.

Θ includes information about the probability distribution of a value x_i of a variable X_i, given the values of its immediate predecessors. The unique joint probability distribution over $<X_1, X_2 \ldots X_N>$ that a network *B* describes can be computed using:

$$P_B(X_1 \ldots X_N) = \prod_{i=1}^{N} P(x_i \mid parents(X_i)) \quad (1)$$

3.1 Learning BBN from Data

In the process of efficiently detecting the impact that the neighbouring characters apply to the target character, prior knowledge is not always straighforward. Thus, a BBN should be learned from the training data provided. Learning a BBN unifies two processes: learning the graphical structure and learning the parameters Θ for that structure. In order to seek out the optimal parameters for a given corpus of complete data, we directly use the empirical conditional frequencies extracted from the data [3]. The selection of the variables that will constitute the data set is of great significance, since the number of possible networks that could describe these variables equals to:

$$2^{\frac{N(N-1)}{2}} \quad (2)$$

where N is the number of variables [6]. We use the following equation along with Bayes theorem to determine the relation r (or Bayes factor) of two candidate networks B1 and B2 respectively:

$$r = \frac{P(B_1 \mid D)}{P(B_2 \mid D)} \quad (3)$$

$$P(B \mid D) = \frac{P(D \mid B)P(B)}{P(D)} \quad (4)$$

where:

P(B|D) is the probability of a network *B* given data *D*.
P(D|B) is the probability the network gives to data D.
P(D) is the 'general' probability of data.
P(B) is the probability of the network before seen the data.

Applying equation (3) to (4), we get:

$$r = \frac{P(D \mid B_1)P(B_1)}{P(D \mid B_2)P(B_2)} \tag{5}$$

Having not seen the data, no prior knowledge is obtainable and thus no straightforward method of computing P(B1) and P(B2) is feasible. A common way to deal with this is, is to follow the standard BBN approach and assume that every network has the same probability with all the others, so equation (5) becomes:

$$r = \frac{P(D \mid B_1)}{P(D \mid B_2)} \tag{6}$$

The probability the model gives to the data can be extracted using the following formula of Glymour and Cooper, [4]:

$$P(D \mid B) = \prod_{i=1}^{n} \prod_{j=1}^{qi} \frac{\Gamma(\frac{\Xi}{q_i})}{\Gamma(\frac{\Xi}{q_i} + N_{ij})} \prod_{k=1}^{ri} \frac{\Gamma(\frac{\Xi}{r_i q_i} + N_{ijk})}{\Gamma(\frac{\Xi}{r_i q_i})} \tag{7}$$

where:

- Γ is the gamma function.
- n equals to the number of variables.
- ri denotes the number of values in i:th variable.
- qi denotes the number of possible different data value combinations the parent variables can take.
- Nij depicts the number of rows in data that have j:th data value combinations for parents of i:th variable.
- Nijk corresponds to the number of rows that have k:th value for the i:th variable and which also have j:th data value combinations for parents of i:th variable.
- Ξ is the equivalent sample size, a parameter that determines how readily we change our beliefs about the quantitative nature of dependencies when we see the data. In our study, we follow a simple choice inspired by Jeffreys' [5] prior. Ξ equals to the average number of values variables have, divided by 2.

We have applied the above equation to tabular data, meaning that the training file contained columns that correspond to the distinct variables of the network and the rows that correspond to each data entry.

Given the great number of possible networks produced by the learning process, a search algorithm has to be applied. We follow greedy search, which is based on the assumption that all possible network combinations produce a candidate best one, with one modification: instead of comparing all candidate networks, we consider investigating the set that resembles the current best model most, meaning that we consider examining other networks from the group of those that have almost same set of dependency statements. In general, a BBN is capable of computing the probability distribution for any partial subset of variables, given the values or distributions of any subset of the remaining variables. Note that the values have to be discretised, and different discretisation size affects the network. As we shall discuss in the result section, BBN are a significant tool for knowledge representation, visualising the relationships between features and subsets of them. This fact has a significant result on identifying which features are actually affect the class variable, thus reducing training data size without any significant impact in the performance.

4 Predicting Text Sequence

Having discussed the advantages and disadvantages of STEM and T9, our initial goal was simply to incorporate the positive aspects of these into one single approach. Furthermore, resource reduction was a high motivation for our research. As previously mentioned, the most significant problem is ambiguity of letters belonging to the same group. The goal is simply to type the desired character using as less keystrokes as possible. In STEM, the average number of keystrokes for a SMS message reaches 2.072 as measured in a sample of 386870 letters concerning words from a of the DELOS[3] Greek corpus. The ideal number would have to approximate 1. Our approach, which will be referred to as BAPTI (Bayesian Predictive Text Input) from now on, uses Bayesian knowledge to infer about the probability of a letter given the key that was pressed and its immediate predecessors (e.g. sequence of letters entered). We have been experimenting with the Greek language, because it is more ambiguous than English, due to the large number of vowels. We are of the belief that the new proposed methodology, combines speed enhancement with robustness when dealing with words not listed in a dictionary. Moreover, we have managed to incorporate minimal resources, a significant advantage compared with the large dictionary entries of T9 (about five thousand words considered the most popular across an analysis of English texts).

The Bayesian prior probabilities for every letter have been calculated by training Bayesian Belief Networks from large corpora. In our case, we used the DELOS Greek corpus, which is consisted of approximately 70Mb of raw text. BAPTI uses this prior probability to infer about the most probable letter in the group of letters that lie in the key that the user pressed. The level of network complexity is increasing in proportion to the length of the word that the user wishes to enter. However, due to the memory limitations of a mobile phone, we do not consider prefixes consisting of more than three letters. In case the system incorrectly predicts a letter, a special purpose function key (#) can alter the output to the second most probable letter and so on.

[3] DELOS Project Nr: EPET II, 98 LE-12

Figure 1 illustrates an example of a BBN taking the three predecessors of a letter as well as the key that was pressed into account. Nodes three letters before, two letters before and one letter before represent the corresponding prefixes. Node KEY symbolizes the key that was pressed, and takes values from two to nine (nominal). Finally, STATE has three distinct values, namely one, two and three that represent the position of a Greek letter in a key group. The network encodes a conditional probability table that can predict which STATE value is most probable, provided the values of all or a subset of the other nodes. As an example, consider that a user wants to write the Greek word "ΗΛΙΚΙΑ". Suppose also that the system has correctly guessed the subset "ΗΛΙΚ". In order to enter letter "Ι", the user presses key 4 where letters Η, Θ and Ι lie. The network can calculate probability for each of them given the prefix "ΛΙΚ" and key 4. The most probable letter would be returned. In case that it is not the correct one, the system would output the second most probable or the third. Throughout the experimental phase, using prefixes of three, prediction accuracy never dropped below 95.5%.

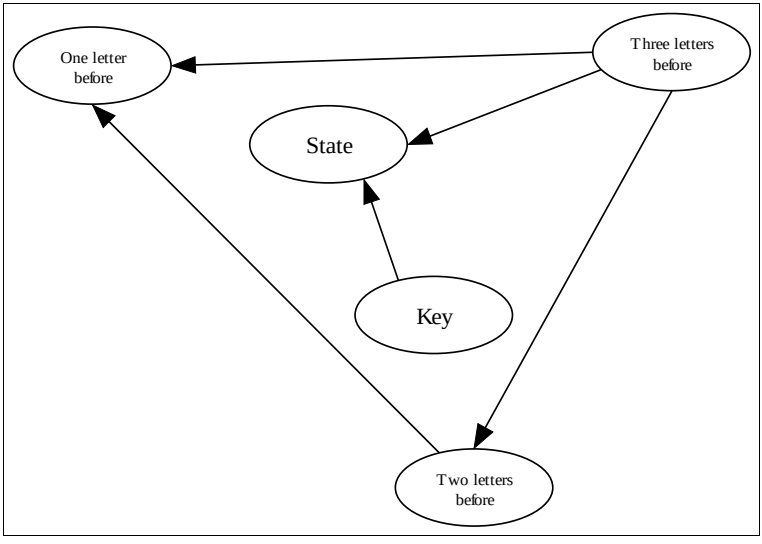

Fig. 1. Diagrammatic representation of Bayesian network obtained

As expected, the more complex a Bayesian network is, the less probable it is for the system to predict the incorrect letter. This of course directly imposes an impact to memory requirements. However, even in the worst case, the number of states that the system should hold in memory is approximately 330.000, a number that seems rational and operative to store.

5 Keystroke Level Model to Evaluate Proposed Method

Keystroke Level Model (KLM) is an analytic predictive method inspired by the Human Motor Processor Model [2]. This model focuses on unit tasks within a user

machine interaction environment which consists of a small number of sequenced operations. The model assumes two phases in task execution. During the first phase decisions are made on how to accomplish the task using the primitives of the system. During the second phase the execution of the task takes place without high level mental activity. The model assumes expertise from user. This method has been empirically validated against a range of systems and a wide selection of tasks, and the predictions made were found to be remarkably accurate (prediction errors less than 20%, as stated by Olson and Olson, [10]).

Assuming negligible times for system (mobile device) response and mental operators (the user is assumed to have decided what to write and knows exactly the positioning of letters on the keypad) , we can develop a model to predict times for an expert user to enter a word. According to this model the time to complete entry of a word using STEM is:

$$T_{STEM} = \text{time to enter X letters + time to move to another key} = X[nT_P + T_{PER} + (1-P_{CK})T_{WAIT}] + (X-1)P_{CK}T_{CK} \quad (8)$$

And time to complete entry of a word using the proposed method is:

$$T_{BAPTI} = \text{time to enter X letters (no } T_{WAIT} \text{ required)+ time to move to another key + time to press \#} = X[T_P + T_{PER}] + (X-1)P_{CK}T_{CK} + X\,(P_{ERROR1} + P_{ERROR2})\,(T_{CK} + T_P) \quad (9)$$

where:

- *X* denotes the number of letters for a specific word.
- *n* denotes the average number of keystrokes to select a specific letter using STEM (calculated 2.0229 from a sample of 386870 letters).
- T_P denotes average key press time. (165 milliseconds (Silfverberg et. al 2000))
- T_{PER} denotes time required from user to perceive correct entry.(500 milliseconds).
- P_{CK} probability of requiring a letter contained in a different key than the previously pressed. (calculated 0.89 from a sample of 386870 letters).
- T_{WAIT} time waiting for cursor to proceed, when successive letter contained in the same key.(depends on phone, for Nokia models is 1500 milliseconds [13]).
- T_{CK} required for a user to move to another key. (approximately calculated by using Fitt's Law: 215 milliseconds [13]).
- P_{ERROR1},P_{ERROR2} are probability for a proposed letter not be the required one, and probability for the second proposed letter not be the required one, respectively. (calculated as 0.045 and 0.002 respectively).

Applying equations (8) and (9), we obtain T_{STEM} = 5695,8 msec and T_{BAPTI}= 3590,5 msec for an average Greek word length (X=6). Increase in task efficiency is 34,72% in terms of time required and average number of keystrokes required is 12,13 and 6,39 respectively, a difference of 47,35%. Modeling of T9 method does not give accurate results because of the inconsistent behaviour of the algorithm. More

specifically, the keystrokes per letter required is reduced to one, except from the cases where the first proposed word is not the one that user wants to enter forcing him to choose across a list of proposed words. Secondly, if the word required is not in T9's dictionary, the user has to alter the text entry method to STEM thus further reducing efficiency of the task. Unfortunately no published study exists concerning the proportion of desired words present in the dictionary –especially for Greek language, and on how often a word other than the desired one appears. Therefore, no accurate dialogue modelling can take place. Direct comparison of BAPTI against T9 should take place using actual prototypes to have an indication of performance. However, lack of such a hardware prototype limits our research in this point.

6 Prototype Implementation and Performance Evaluation

In the present section, we discuss usability issues in the context of STEM and BAPTI. Having already theoretically modeled each technique's dialogue performance concerning the time to complete word entry, we intended to verify BAPTI's performance in the real world. For that reason, we have implemented a mobile phone keypad emulator where users were supposed to edit messages using BAPTI. Figure 2 depicts a snapshot of the described tool. The left part of the tool consists of a single line mobile screen simulator, where the user verifies the system's output and the standard keypad that mobile phones use. The arrangement of the Greek letters in every key was identical to that of Nokia 6110 and 5110 models. For our experiments, we considered only capital letters, since they are most commonly used by the Greek users. Moreover, in the lower part, the system outputs the probability for each state of the last pressed key. Emulator traces the number of keystrokes using BAPTI and compares to those that would be needed by STEM for the same message. The right part of the simulator contains the graphical representation of the number of keystrokes needed by during the editing procedure. This graph is dynamically updated across the editing progress, thus providing a better sense of each method's behavior.

The dashed line represents the number of keystrokes using STEM while the continuous line represents the number of keystrokes using our approach. As we could observe from an example text messaging task, BAPTI is better than STEM throughout the whole editing process with an average keystroke number that approximates 1.06. On the other hand, STEM converges to a value of about 1.94 which agrees to our initial expectations (Figure 2). Performance measurements in terms of time required to complete text entry task could not be compared directly to the KLM model at the moment, because of the non negligible response time required by the system to find the appropriate probabilities due to early prototyping issues.

To evaluate real world performance of the proposed method, we have conducted preliminary experiments using ten SMS prototype phrases of varying length containing high informal word rate. Table 1 tabulates analytic results concerning the number of keystrokes needed from BAPTI and STEM and error rates of single errors and double errors (e.g. second and third keystroke required to access desired letter respectively).

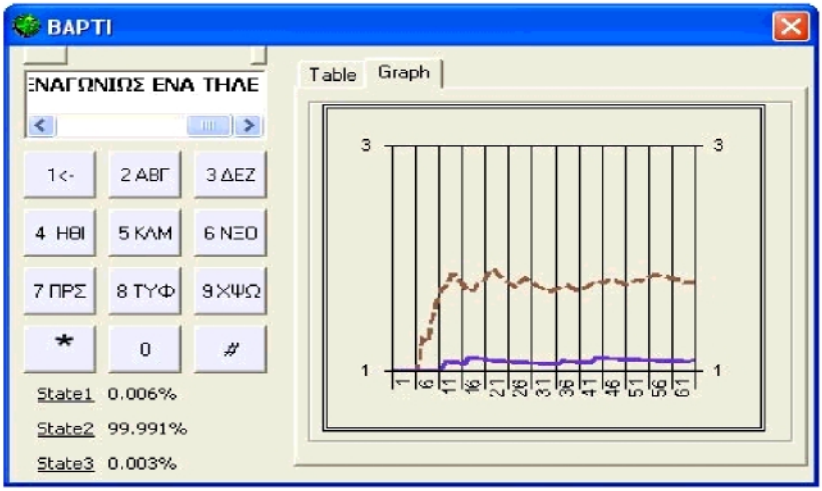

Fig. 2. BAPTI SMS emulator.

Table 1. Comparative results regarding the needed number of keystrokes for STEM and BAPTI methods, obtained from real world preliminary experiment.

Phrase	#of words	Characters	BAPTI	STEM	% Improvem.	S.Errors	D. Errors
1	28	143	158	256	38.3%	9.1%	0.7%
2	29	160	169	268	36.9%	5.6%	0.0%
3	29	158	179	279	35.8%	8.2%	2.5%
4	14	87	102	142	28.2%	8.0%	4.6%
5	25	150	165	265	37.7%	8.7%	0.7%
6	20	122	134	218	38.5%	6.6%	1.6%
7	28	154	169	276	38.8%	7.1%	1.3%
8	16	86	92	150	38.7%	4.7%	1.2%
9	19	117	129	213	39.4%	8.5%	0.9%
10	10	58	63	107	41.1%	8.6%	0.0%
Total	**218**	**1235**	**1360**	**2174**	**37.4%**	**7.5%**	**1.3%**

Having analyzed the results we could clearly distinguish an improvement of 37.4% concerning the effort required to edit a message in terms of keystroke number. The percentage of correctly predicting a letter by BAPTI is 91.2%. Note that the average keystroke numbers excluding spaces within words for BAPTI and STEM are 1.118 and 1.907 respectively, depicting an improvement of 41.3%. The difference between the methods is considered statistically significant ($p<0.0001$ and the 95% confidence interval of the difference is [-0.86,-0.71]). A notable remark is that the extracted results have a close convergence to our initial predictions derived by KLM modeling.

7 Conclusions

We have presented a novel technique, named BAPTI for improving text entry usability in mobile keypads. BAPTI is based on Bayesian knowledge, obtained by training with raw text corpora, about the probability of a letter that was pressed by the user to be the desired one, among the other candidate letters that belong to the same key, given its predecessors. We have argued that BAPTI performs better than the standard mobile input method in context of keystroke count. A significant advantage of our approach is that it is not restricted to orthographic linguistic knowledge, as with dictionary-based methods, which would decrease its performance in case of unknown words. We have also emphasized on the multilingual character of BAPTI, which allows for easy adaptation to any other language. Concerning the evaluation, we have modeled both BAPTI and STEM using Keystroke Level Modeling and formed a prototype emulator for actual experimentation. Theoretical analysis depicted satisfactory results, with BAPTI to behave better than STEM by a factor of 34.72% concerning time efficiency and approximately 47,35% concerning the number of key presses. Preliminary experiments were carried out using the implemented emulator and have verified the accuracy of KLM predictions. A request for patent concerning the BAPTI technique is in process.

As for future work, our intention is to improve disambiguation accuracy by incorporating more domain specific corpora with the existing, as well as developing an algorithm that would allow quick typing but without reducing the sense of word progress towards the user's entry goal. Prototype should be improved also in terms of system response time thus enabling extensive user testing and comparison of proposed text entry methods in various aspects.

References

1. Buchanan G., Jones M., Thimbleby H., Farrant S., Pazzani M. Improving mobile internet usability. Proceedings of the Web 2001 Conference, Hong Kong, ACM Press (2001)
2. Card, S. K., Moran, T. P., & Newell, A. The keystroke-level model for user performance time with interactive systems. Communications of the ACM, 23 (7), (1980) 396-410.
3. Cooper J., Herskovits E. A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 9 (1992) 309-347.
4. Glymour C., Cooper G. (eds.). Computation, Causation & Discovery. AAAI Press/The MIT Press, Menlo Park (1999)
5. Jeffreys H. Theory of Probability. Clarendon Press, Oxford. (1939)
6. Jensen F.. An Introduction to Bayesian Networks. New York: Springer-Verlag (1996)
7. Ketola P., Roykkee M. Three Facets of Usability in Mobile Handsets, In CHI 2001, Workshop, Mobile Communications: Understanding Users, Adoption & Design Sunday and Monday, April 1 -2, 2001 Seattle, Washington. ACM (2001)
8. Longmate E., Baber C., Trabak A.. A study of text messaging within a digital community, In Human Computer Interaction 2001.Panhellenic conference with international participation, December 7-9 2001, Patras, Greece (2001) 257-262.
9. Mitchell T. Machine Learning. Mc Graw-Hill (1997)

10. Olson J., Olson G. The Growth of Cognitive Modeling in Human-Computer Interaction Since GOMS, Human Computer Interaction ,Vol.5, Lawrence Erlbaum Associates, (1990) 221-265.
11. Pearl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann (1988)
12. Rodden, T., Cheverst K, Davies N. and Dix A. Exploiting Context in HCI design for Mobile Systems. In Workshop on Human Computer Interaction with Mobile Devices. Glasgow (1998)
13. Silfverberg, M., MacKenzie, I. S., & Korhonen, P. Predicting text entry speeds on mobile phones. Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2000, New York: ACM (2000) 9-16.