Measuring Performance of a Predictive Keyboard Operated by Humming

Ondřej Poláček, Adam J. Sporka, and Zdeněk Míkovec

Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo nam. 13, 12135 Prague 2, Czech Republic
{polacond,sporkaa,xmikovec}@fel.cvut.cz

Abstract. A number of text entry methods use a predictive completion based on letter-level n-gram model. In this paper, we investigate on an optimal length of n-grams stored in such model for a predictive keyboard operated by humming. In order to find the length, we analyze six different corpora, from which a model is built by counting number of primitive operations needed to enter a text. Based on these operations, we provide a formula for estimation of words per minute (WPM) rate. The model and the analysis results are verified in an experiment with three experienced users of the keyboard.

Keywords: Text Entry Methods; N-Gram Model; Measuring Performance; Non-Verbal Vocal Interaction.

1 Introduction

Many virtual keyboards, which has been designed for users with severe motor impairments, employ a text completion technique. They suggest possible continuations of an already written text (further referred to as context). The continuations are extracted from a language model and sorted according to their probability in the context. After entering a completion, the virtual keyboard offers a different set of completions with respect to the updated text. The question arises what is the optimal size of the language model in terms of efficiency of the text entry.

In our previous work, we described a novel text entry method called Humsher [1]. The method is operated by non-verbal vocal interaction (NVVI) [2]. The NVVI can be described as an interaction modality in which sounds other than speech are produced, for example humming [3] or vowels [4]. The Humsher utilizes an adaptive language model for text prediction and is based on selecting letters or longer completions from a list sorted by the probability of their occurrence in the text being entered. We measured the performance of the Humsher in a user study with 17 participants. We explored four interface designs which differed in the organization of the suggestions list (layout, size, etc.) and the navigation in it. We also performed an evaluation of the method with people with motor and speech disability, and found Humsher suitable for them.

2 Related Work

There is a wide range of text entry methods targeting the motor-impaired users, however, several common principles of text entry can be identified in the literature – predictive completion, scanning, and ambiguous keyboards. These principles are often combined to gain a better performance. A text entry method can be accelerated by prediction when a list of possible completions is updated with each entered letter or word. This reduces the number of keystrokes per character. The letters or words already entered are very often used to improve the prediction accuracy. An example of this is a predictive system called Dasher [5] which is based on a continuously updated display and an adaptive letter-level language model. In ambiguous keyboards, the alphabet is divided into several groups of characters. Each group is then assigned to one of the keys. The user enters the desired characters by selecting corresponding keys and is requested to perform a dictionary-assisted disambiguation. The ambiguous keyboards were designed for physically impaired people [6]. Scanning keyboards are often used when the number of distinct signals, which can be issued by the user, is limited to only one or two. Scanning is often combined with ambiguous keyboards [7].

The non-verbal vocal interaction (NVVI) has also been used for keyboard emulation. Sporka et al. [3] describe a humming-based method of keyboard emulation. Each vocal gesture is assigned a specific key on the keyboard. When a sound is produced a corresponding key is emulated. Another keyboard operated by humming is CHANTI [8] which is based on scanning ambiguous keyboard QANTI [7] but the scanning is replaced by direct selection of a key by the humming. The humming input and the PPM language model [9] is combined in Humsher [1]. Both CHANTI and Humsher were evaluated with motor-impaired users.

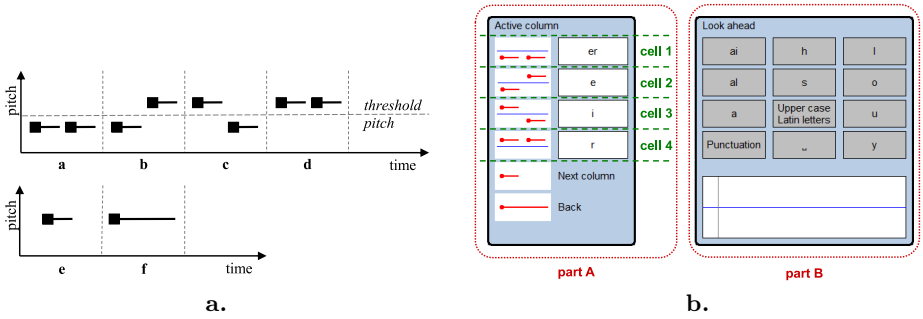

Fig. 1. a. Vocal gestures used for interaction. **b.** Humsher interface. Part A – active column, part B – look ahead matrix.

3 Overview of the Humsher

Our virtual keyboard Humsher utilizes an n-gram language model on letter level based on prediction by partial matching (PPM) algorithm [9]. An n-gram is a sequence of n characters. The n-grams with length equal to one, two, and three character are being called unigrams, bigrams, and trigrams respectively. The language model contains n-grams together with their probabilities. The *order of the model* further refers to the longest n-gram contained in the language model. The model is initialized from a small text corpus, but it adapts as the user types. Possible completions extracted from the language model are offered to the user sorted according to their probability. The probability is predetermined by the context of given length. For kth-order language model the maximal length of the context is equal to k-1. It stands for a reason that the order of the model has an effect on efficiency of both the model and the method itself. For example, the first-order model stores unigrams therefore only static probability of letters can be used for prediction. In case of second-order model, the model stores probabilities of bigrams and one letter context can be used to improve the prediction.

Once the language model is initialized, the keyboard starts offering possible completions sorted according to their probability. The completions can be both letters and strings. After entering a completion, the context is updated and probabilities of following completions are recounted and the layout is displayed accordingly. The length of a completion is not limited, only probability matters. The completions to display are chosen according to the following steps:

1. Add all unigrams to the list L that will be displayed.
2. For each n-gram in the list L compute probability of all (n+1)-grams using the language model and add them to the list L if their probability is higher than a threshold.
3. Repeat step 2 until no n-gram can be added.
4. Sort the list L according to probability of each n-gram.

The original paper on the Humsher [1] discusses and evaluates four different keyboard interfaces. For our experiment, we chose the Direct interface as it was the fastest one relatively well accepted by the users. The keyboard interface utilizes six vocal gestures as depicted in Fig. 1a. The vocal gestures are explicitly identified by its length (short/long) or by its pitch (low/high). In order to distinguish low and high tones a threshold pitch needs to be adjusted for each user - e.g. the difference between male and female voice is as much as one or two octaves. The keyboard (see Fig. 1b) allows users to directly choose from four cells (labeled cell 1 to 4) in the Active column (part A). These cells contain strings that have been determined as the most probable following letters of the context. Cells can be directly selected by vocal gestures depicted in Fig. 1a (gesture a: two consequent low tones - cell 1, b: a low tone followed by a high tone - cell 2, c: a high tone followed by a low tone - cell 3, d: two consequent high tones - cell 4). If there is no cell in the Active column that contains the desired letter, the user has to move the leftmost column in the Look ahead (part B) to the Active column by producing a single short tone (see Fig. 1b, gesture e), and keep repeating it until

the desired n-gram appears in one of the cells in Active column. The written text can be erased by producing a long tone (see Fig. 1b, gesture f). The longer the user keeps producing the tone the faster are the letters erased.

4 Analysis

The main aim of the analysis was to explore how the order of the language model affects efficiency of the Humsher. The other question was how the ideal length of the n-grams differed for various text corpora and languages. We used four publicly available text corpora and three languages:

1. *Dasher.* The corpus is available as a training text for the Dasher[1]. Corpora for various languages is provided – we used English, German and Czech.
2. *AacText.* A crowdsourced corpus of augmentative and alternative communication (AAC) collected by Vertanen and Kristensson [10].
3. *EnronMobile.* A subset of sentences written by Enron employees on BlackBerry mobile devices published by Vertanen and Kristensson [11].
4. *SmsCorpus.* This public research corpus contains SMS messages collected by National University of Singapore[2].

All text corpora were split to training and test data sets. The training data set was used to train the language model and the test data set was used for text entry simulation. Detailed information about corpora is shown in the Table 1.

We used the *gestures per character* (GPC) measure [12] to express a performance of the text entry method. As a gesture is regarded as an atomic operation, and in our case (the humming input) vocal gestures are treated as atomic operations. The GPC rate is defined by the Eq. 1 where CM is a number of column movements that corresponds to finding the desired column by linear scanning, DS is a number of direct selections that corresponds to selection of desired cell by one of four vocal gestures, and $|T|$ is a length of the test data set. The sum of CM and DS corresponds to total number of vocal gestures in an input stream.

$$GPC = \frac{CM + DS}{|T|} \tag{1}$$

The GPC rate is a characteristic measure that is similar to *keystrokes per character* (KSPC) measure and can be used for capturing initial performance of a text entry method [13]. The theoretical text entry speed in terms of *words per minute* (WPM) can be estimated from the CM and DS variables according to the Eq. 2. The constants a and b represent an average time needed for the column movement and the direct selection respectively. These constants are measured in an subsequent experiment with users described in the Section 5.

$$WPM = \frac{|T|}{5} \times \frac{60}{aCM + bDS} \tag{2}$$

[1] `http://www.inference.phy.cam.ac.uk/dasher/Download.html`

[2] `http://wing.comp.nus.edu.sg:8080/SMSCorpus/history.jsp`; version 2011.12.30

We analyzed theoretical GPC for each corpus, while changing the order of the language model (i.e. size of n-grams stored in the model) from 1 to 16. As the user actions were simulated by computer, no human errors were taken into account. The resulting dependence of GPC on the order of the language model is depicted in Fig. 2 The effect is significant for 1st-order to 5th-order model. The difference for higher order models is negligible. Minimal GPC values and corresponding order of the language model are shown in the Table 2. Nevertheless, using 6th-order model is sufficient for each corpus as the difference of 6th-order GPC value and the minimal GPC value is always less than 2%.

Table 1. Minimal theoretical GPC and corresponding order of the language model

Corpus	Training part size	Words	Unique words	Sentences	Test part size
Dasher (English)	289 KB	51 064	8 676	2 568	30 KB
Dasher (German)	884 KB	122 130	21 951	7 051	53 KB
Dasher (Czech)	419 KB	59 179	21 131	6 148	33 KB
AacText	127 KB	25 125	2 206	3 646	14 KB
EnronMobile	97 KB	18 472	3 041	2 050	10 KB
SmsCorpus	2 748 KB	536 029	27 905	81 836	80 KB

Table 2. Minimal theoretical GPC and corresponding order of the language model

Corpus	Dasher (English)	Dasher (German)	Dasher (Czech)	AacText	EnronMobile	SmsCorpus
Min. GPC	1.23	1.17	1.55	1.11	1.27	1.42
Order	7	11	5	8	6	11

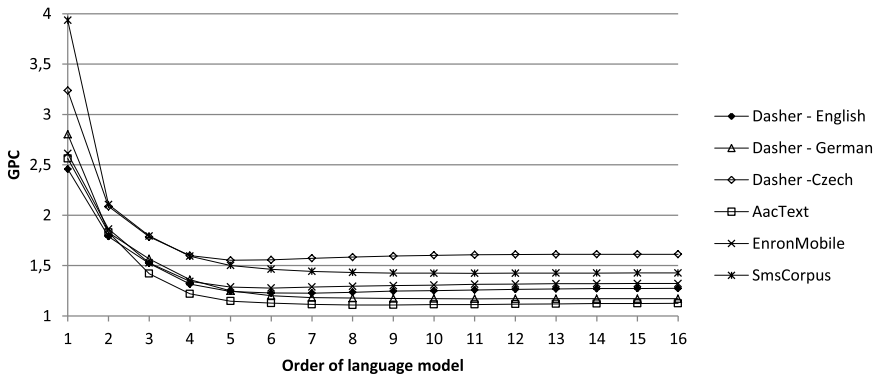

Fig. 2. Dependence of Humsher text input efficiency in terms of GPC on the order of language model

5 Experiment

The aim of the experiment was to validate results from the aforementioned analysis and to find out the a and b values for the WPM estimation (see Eq. 2). In the experiment 3 able-bodied participants (all men, aged 29–36) took part. They had previous experience with the Humsher as they already participated in an experiment described in [1].

The task in the experiment was to copy two sentences. The independent variables were a corpus and an order of the language model. We used all six corpora and following orders of the language model: 1, 3, 6, and 12. The sentences were unique for each corpus and they were chosen from the test part of each corpora. The experiment was conducted in two trials, in each trial the participant had to copy the two sentences under all conditions (4 orders $\times$ 6 corpora = 24 conditions). Each participant had to copy 96 sentences in the experiment (2 trials $\times$ 2 sentences $\times$ 24 conditions). The experiment took approximately 4 hours per participant. The sequence of corpora the order of the model was randomized for each participant to compensate for learning effects.

Results. Two-factor ANOVA showed a significant interaction between the order and the corpus for GPC ($F(15,264) = 21.3$, $p<.001$) and WPM ($F(15,264) = 6.45$, $p<.001$) values. Therefore, we evaluated the effect of order on both values separately for each corpus. The Table 3 shows significantly different pairs in the GPC and WPM rates denoted by *less than* ($<$) and *greater than* ($>$) sign. The differences were considered significant on $p<.05$ level.

Two-factor ANOVA for the a and b values showed no interaction between order and corpus, nor significant main effect for the corpus factor. However, significant main effect for the context was found for both values (a: ($F(3,264) = 40.5$, $p<.001$); b: ($F(3,264) = 15.2$, $p<.001$)). Subsequent ANOVA and post-hoc pairwise comparisons on the order of the model revealed that both values a ($F(3,254) = 33.1$, $p<.001$) and b ($F(3,254) = 14.1$, $p<.001$)) are significantly lower for the 1st-order model. The explanation is simple. Using the 1st-order model ($a = 1.30$s; $b = 1.25$s), the layout of characters is static regardless the already written context. The letter are offered according to their frequency in a corpus. In case of higher-order models ($a = 1.58$s; $b = 1.56$s), on the other hand, a context is used for prediction and it causes different layout of letters (or n-grams) as the user types. Therefore, higher effort is needed to visually locate desired letters, which is reflected in longer time needed to produce a vocal gesture. Note that the difference between the a and b values is minimal. Therefore, we can merge them into one variable, which simplifies the Eq. 2.

In order to evaluate the model of WPM estimation (see Eq. 2), a correlation calculation was performed between estimated and measured WPM for all test conditions. Correlation coefficients of the WPM rate for the three participants were .91, .88, and .93 respectively. As correlations above .90 are considered very high in experiments with users [14], this result indicates validity of the model for WPM estimation (see Eq. 2). However, the correlation can be even improved

Table 3. Significant differences of the four orders of language model (1,3,6, and 11) in GPC and WPM rates for each corpus.

Corpus	GPC	WPM
Dasher (English)	1 > 3,6,12	1 < 3,6,12
Dasher (German)	1 > 3,6,12	1 < 3 < 6,12
Dasher (Czech)	1 > 3,6,12	1 < 3,6,12
AacText	1 > 3 > 6,12	1 < 3 < 6,12
EnronMobile	1 > 3,6,12	1 < 3 < 6,12
SmsCorpus	1 > 3 > 6,12	1 < 3 < 6,12

by incorporating corrections. The number of correction per character rate was .03, .12, and .10 for each participant respectively. The GPC correlation was even higher (.98, .97, and 0.98) confirming the correctness of the corpora analysis.

6 Conclusion

In this paper we analyzed dependence of gesture per character (GPC) rate on order of language model when using Humsher – a predictive keyboard operated by humming. For the analysis purposes, we used six publicly available text corpora with three languages variants. The analysis results were verified in an experiment with three experienced users. We defined and verified a relation for estimating WPM rate from number of primitive operations. In the experiment, we found that producing vocal gestures was faster when no prediction was used and the layout of characters was statically arranged (1st-order language models). However, the prediction used with higher-order models helps to achieve significantly better GPC and WPM rates. We also found that the 6th-order model is adequate for the optimal performance of the keyboard. It is significantly faster than 1st- and 3rd-order model, but is not significantly faster than the 12th-order model. In some cases the 12th-order model even decreases the performance of typing.

The number of corrections influences the WPM rate. However, this fact is not incorporated in the model for WPM estimation. Therefore, in the future work we would like to focus on improvement of the WPM estimation formula by incorporating the error rate or the number of corrections.

Acknowledgments. This research has been supported by the Veritas project (IST-247765).

References

1. Polacek, O., Mikovec, Z., Sporka, A.J., Slavik, P.: Humsher: A Predictive Keyboard Operated by Humming. In: 13th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS'11, pp. 75–82. ACM, New York (2011)

2. Igarashi, T., Hughes, J.F.: Voice as sound: using non-verbal voice input for interactive control. In: 14th Annual ACM Symposium on User Interface Software and Technology, UIST'01, pp. 155–156. ACM, New York (2001)
3. Sporka, A. J., Kurniawan, S.H., Slavik, P.: . Non-speech operated emulation of keyboard. In: Clarkson, J., Langdon, P., and Robinson, P. (eds.) Designing Accessible Technology, pp 145–154. Springer, London (2006)
4. Harada, S., Landay, J.A., Malkin, J., Li, X. and Bilmes, J.A.: The Vocal Joystick: Evaluation of voice-based cursor control techniques. In: 8th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS'06, pp. 197–204. ACM, New York (2006)
5. Ward, D.J., Blackwell, A.F., MacKay, D.J.C.: Dasher – a data entry interface using continuous gestures and language models. In: 13th ACM Symp. on User Interface Software and Technology, UIST'00, pp. 129-137. ACM, New York (2000)
6. Kushler, C.: AAC: Using a Reduced Keyboard. Technical report (1998)
7. Felzer, T., MacKenzie, I., Beckerle, P., Rinderknecht, S.: Qanti: A software tool for quick ambiguous non-standard text input. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) Computers Helping People with Special Needs. LNCS, vol. 6180, pp. 128–135. Springer, Heidelberg (2010)
8. Sporka, A.J., Felzer, T., Kurniawan, S.H., Polacek, O., Haiduk, P., MacKenzie, I.S.: CHANTI: Predictive Text Entry Using Non-verbal Vocal Input. In: 2011 Annual Conference on Human Factors in Computing Systems, CHI'11, pp. 2463–2472. ACM, New York (2011)
9. Teahan, W.: Probability estimation for PPM. In: New Zealand Research Students' Conference (1995)
10. Vertanen, K., Kristensson, P.O.: The Imagination of Crowds: Conversational AAC Language Modeling using Crowdsourcing and Large Data Sources. In: Conference on Empirical Methods in Natural Language Processing, EMNLP'11, pp. 700–711. Association for Computational Linguistics (2011)
11. Vertanen, K., Kristensson, P.O.: A versatile dataset for text entry evaluations based on genuine mobile emails. In: 13th Int. Conf. on Human Computer Interaction with Mobile Devices and Services, MobileHCI'11, pp. 295–198. ACM, New York (2011)
12. Wobbrock, J.O.: Measures of text entry performance. In: MacKenzie, I.S., and Tanaka-Ishii, K. (eds.) Text Entry Systems: Mobility, Accessibility, Universality, pp. 47–74. Morgan Kaufmann (2007)
13. MacKenzie, I.S.: KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques. In: 4th International Symposium on Human-Computer Interaction with Mobile Devices, pp. 195–210, Springer, Heidelberg (2002)
14. MacKenzie, I. S.: Movement time prediction in human-computer interfaces. In: Graphics Interface '92, pp. 140-150, Toronto. Canadian Information Processing Society (1992)