Word n-grams for cluster keyboards

Nils Klarlund
klarlund@research.att.com
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932

Michael Riley
riley@research.att.com
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932

Abstract

A *cluster keyboard* partitions the letters of the alphabet onto subset keys. On such keyboards most words are typed with no more key presses than on the standard keyboard, but a key sequence may stand for two or more words. In current practice, this ambiguity problem is addressed by hypothesizing words according to their unigram (occurrence) frequency. When the hypothesized word is not the intended one, an error arises.

In this paper, we study the effect of deploying large, n-gram language models used in speech recognition for improving the error rate. We use the North American Business News (NAB) corpus, which contains hundreds of millions of words.

We report on results for the telephone keypad and for cluster keyboards with 5, 8, 10, and 14 keys based on the QWERTY layout. Despite our assumption that a word hypothesis must be displayed promptly, we show that the error rate can be reduced to up to one-fourth of the rate of the unigram method.

1 Introduction

The text entry methods T9 (by Tegic) and iTap (by Motorola) are found on many mobile phones. They are based on the existing clustering of letters on the eight digit keys 2 to 9 as found on standard telephones in the USA and elsewhere. Other layouts may be based on QWERTY such at the one we propose in Fig. 1, where typing the key marked $\{Q, W\}$ produces either q or w. To type on these

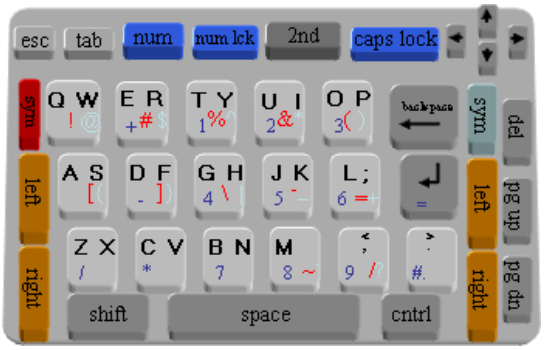

Figure 1: Q14: a familiar layout, but with ambiguous keys. Up to 99.5% of all words entered will be correctly identified through techniques developed in this paper.

devices, the user presses the key corresponding to the letter only once. When the key corresponding to the spacebar is pressed, a dictionary is consulted to find the word corresponding to the sequence of digit keys. If there are such several words, called *homographs* of the cluster layout, then the most likely one is suggested according to its occurrence frequency as calculated over some corpus. If the suggestion is wrong, then the user must explicitly choose a word from a list of alternatives.

If the correct word is not among the alternatives, the user may enter the word using an alternative method. For example, the multi-tap method requires that the cluster key be pressed repeatedly until the correct letter appears. (Heuristics may be employed to hypothesize words that are not in the vocabulary. We do not consider this aspect in the present article.)

1.1 Error Rate and Human Performance

The overhead incurred by the user for validation of a hypothesis is hard to quantify, see (Silfverberg et al., 2000), where it is estimated that five per-

cent of words are hypothesized incorrectly (for T9 used to enter English). According to one model in (Silfverberg et al., 2000), predicted performance drops from 40 words per minute (wpm) to 25 wpm if the user visually inspects every word as soon as it is typed. Each inspection is assumed to add 500 milliseconds to the entry time of the word. Obviously, reducing the error rate will discourage visual inspection and improve efficiency since the process will resemble traditional typing.

1.2 QWERTY-based Cluster Layouts

In addition to T9, at least one cluster keyboard based on QWERTY has been suggested: the *non-keyboard QWERTY* layout (of Visual Languages et al., 1999), where the 26 letters are grouped onto eight keys according to $\{Q, A, Z\}$, $\{W, S, X\}$, $\{E, D, C\}$, $\{R, F, V, T, G, B\}$, $\{Y, H, N, U, J, M\}$, $\{I, K\}$, $\{O, L\}$, and $\{P\}$. Thus, letters serviced by the same finger when touch typing are on the same key in this layout, which we name *Q8*. In (of Visual Languages et al., 1999), it is stated that 3.5% of keyed input calls for correction when a simple syntactic analysis is added to the frequency-based disambiguation, but the texts for which this holds are not detailed.

Since the mental map of the keyboard layout remains the same, QWERTY based layouts may help overcome user resistance to learning new typing techniques. We devise three other layouts, having 5, 10 and 14 keys.

1.3 Our Approach and Results

In this paper, we show through simulations that stochastic language models can be used in a realistic setting to significantly reduce the error rate for typing on variety of cluster keyboards. Our language models consist of probability estimates of occurrences of n-*grams*, sequences of n consecutive words, where n is 1 (*unigrams* or word occurrences), 2 (*bigrams*), or 3 (*trigrams*). Calculated from large corpora, these probabilities are used to hypothesize the words typed. For $n = 1$, the technique amounts to selecting the most frequent word. For $n > 1$, the estimate of a word is dependent on the pattern of n-grams found in the whole sentence.

The problem with this approach is that the analysis can take place only after the whole sentence has been keyed. The telephone keypad study (Rau and Skiena, 1996) follows this traditional whole-sentence technique. The authors set up a human-factors experiment based on this approach. The feedback given to the users as they type is the sequence of keys entered. Only when users terminate keying the sentence are the predicted words displayed. The authors do not draw any conclusions about their usability study, but they do note that typing discomfort increased with sentence length.

Although the authors dismiss this problem, it seems to us that their whole sentence analysis is flawed as a practical approach. We are convinced that the acceptance of cluster keyboards is contingent on the computer almost always correctly guessing the intended word very soon after it has been keyed.

So, although we employ the same statistical framework as in (Rau and Skiena, 1996), we propose in this paper to use it differently, better matching human expectations. As a starting point, we assume that a beginning user will invariably verify the hypothesized word as soon as is has been keyed. To simplify our analysis, we assume that guessing individual characters is not relevant while the word is being typed. But as soon as a word boundary is entered (such as a space) the computer must produce the most likely word in the dictionary that matches the sequence of keys. In this case, we say that the *delay* parameter d is zero since the last hypothesis is committed immediately.

Naturally, we expect that the performance of this more realistic model would be inferior to that of the whole sentence analysis. But with our results, we do answer in the positive the question whether there is a significant improvement over the unigram model.

To further investigate the usefulness of the $d = 0$ model, we stipulate that the cognitive load on the user will be diminished if the following holds: a correction of a wrongly guessed word can be accomplished through a single press on a single correction key. The effect of the correction key is to replace the first hypothesis with the second. Thus, we also study the model of a text entry system, where the user's action upon word entry is reduced to a single choice: do nothing to select the primary word choice or select the secondary choice. This technique will be of particular significance to the beginning user. Therefore, we are interested in calculating for $d = 0$, the *top two* candidates suggested by the language model.

We also study the improved hypotheses that may be generated if the experienced user accepts a delay before a hypothesis by the computer is made final. Thus for $d = 1$, the hypothesis of the second-to-last word may be updated when the last word has been entered. For $d = 2$, also the hypothesis of the third-to-last word may be updated.

Finally, we summarize in plots the effects on accuracy of increasing the vocabulary size and of *shrinking* the models so that they demand less memory and computation. Our simulations indicate a linear relationship between the measured log perplexity (i.e., cross-entropy) and the various error rates we study.

Related Work

Already in 1948, in his famous paper that founded information theory, C.E. Shannon studied n-grams (on letter sequences that could cross word boundaries) to understand the information-theoretic content of English. Later, speech recognition research has focused on language models built in terms of n-grams over words as the tokens.

This is also the view adopted in (Rau and Skiena, 1996), which investigates the use of language models for the telephone keypad. The keypad layout, which is not used any longer to our knowledge, places the letters Q and Z on the "*"-key. Rau and Skiena use bigrams to disambiguate words, once a complete sentence has been keyed.

The n-gram model applied to characters within words, but where word context is not considered, has been studied in (Dunlop and Crossan, 2000; Forcada, 2001). In this work, a user may type with less than one keystroke per character thanks to word completion techniques.

The perhaps most fascinating example of a highly interactive system is that of Dasher (Ward and MacKay, 2002), which uses contextual information to predict individual characters and character intervals. The intervals are continuously displayed to the user in a branching structure.

The use of cluster keyboards are of particular interest to the development of input methods for people whose typing skills are limited or absent. For example, reduced layouts, with changing assignments of letters, are studied in (Kühn and Garbe, 2001; Johansen and Hansen, 2002). Such systems could be helpful to those who use gaze as a means of input or to people whose motor skills allow them to use only a very few number of keys.

The recent study of reduced keyboards in (Tanaka-Ishii et al., 2002) is somewhat similar to ours. As in the work of Dasher, the authors use a PPM (prediction by partial match) language model. This model merges information about n-grams in a user corpus with unigram probabilities from a base dictionary. The text that is entered by the user is appended to the user corpus. As a result, a dynamic language model emerges, but it is able to predict the last word entered only. The PPM study anticipates the use of a completion key to speed up text entry. In contrast, we focus on error rates for words that are entered in their entirety. The authors conclude that a four-button device, where prediction is based on language models, is as efficient for text entry as a telephone keypad.

2 Framework

We describe here cluster keys, statistical methods, and concepts pertaining to the notion of delay.

2.1 Technical Preliminaries

Our alphabet Σ is {a, ..., z, '', '-', **spc**}. We consider text entry for English. Words are separated by space keys or punctuation symbols entered on special keys. We do consider '' (as in "it's") and '-' (as in "e-mailed") as being clustered on a separate *symbol key*. A particular implementation of a cluster keyboard may instead map these characters to particular explicit key combinations. This would improve our results (to an insignificant degree). The '.' key could be mapped to the symbol key as well without affecting our results: no words in our vocabulary end with a '' or '-'. We do consider abbreviations, but we assume that they are marked through the use of a period after each letter keyed. Our results would be slightly better (see Sect. 6) if we do not consider abbreviations. For example, it is virtually impossible to guess most initials of person names. Fortunately, none of the keyboard layouts considered map "i" and "a" to the same key. Thus, the two standard English words that consist of one character require no disambiguation by the user.

2.2 From Training Sets to Prediction

The language models were built using 250 million words from the North American Business News (NAB) corpus, which was collected by DARPA for use in speech recognition research and benchmarking (Paul and Baker, 1992). It consists of Wall Street Journal articles and similar publications. For testing, we used a held-out subset of approximately 1 million words from that corpus (the NAB LM development test set). To investigate task portability, we also used approximately 17,000 words from the Switchboard corpus (Eval '95 test set) (Godfrey et al., 1992). This corpus consists of transcriptions of telephone conversations covering a variety of topics.

Given a vocabulary size v, we identify the v most frequent words from our training corpus and build a lexicon L that maps each word w to a natural number $\iota(w)$, which we call a *word identifier* (*ID*). Given the desired n-gram order N, we build a stochastic backoff language model *LM* (Katz,

1987) from our corpus using a compact finite automata representation (Riccardi et al., 1995) as very commonly used in large vocabulary speech recognition systems. For any sequence $i_1, \ldots, i_n$ of word IDs, this model estimates the probability that the corresponding sequence of words occurs in the text as a sentence. Further, for a given non-negative real number s, we can *shrink* the language model to factor s using the method of Seymore and Rosenfeld (Seymore and Rosenfeld, 1996) to trade-off the size of the language model against its accuracy. Shrinking removes higher-order n-gram transitions that are similar in transition probability to their lower-order counterparts and thus relatively expendable.

For each word w in a given sentence from the test set, let the $I(w)$ be the set of word IDs corresponding to w and its homographs, i.e., to the set of words in the lexicon that would be mapped to the same key sequence as w. If w is in the dictionary, then its ID is guaranteed to be in the set, i.e., $\iota(w) \in I(w)$. If w is not in the lexicon, it is called *out-of-vocabulary*. We consider that an error. The set $I(w)$ may or may not be empty. If $I(w)$ is not empty, then the word guessing algorithm will be able to suggest a word, albeit wrong. The out-of-vocabulary situation is represented by a special *unknown token* (which also has a word ID representation) that replaces the ID of the word.

Our algorithm hypothesizes words using the automata-theoretic representation as follows. Upon reading h keyed words of the sentence, we calculate for the last ω words, the sets $I_{h-\omega+1}, \ldots, I_h$, where I_k is the set of homographs in the lexicon that correspond to the ambiguous representation of the kth word in the sentence. We call ω the *window size*. We do not use the whole history for efficiency reasons. And, we have informally verified that we lose no precision in practice due to this decision. If I_k as just specified is empty, we replace it by the singleton consisting of the unknown token.

We are looking for the sequence of word IDs $i_{h-\omega+1}, \ldots, i_h$ that is most likely to be a prefix of some sentence. To find this sequence, we first make all states in *LM* final so that it will accept any prefix of a sentence and not just complete sentences. We then construct an automaton S that recognizes all sequences $i_{h-\omega+1}, \ldots, i_h$ such that $i_k \in I_k$ for all $h - w < k \leq h$. Next, we compute the finite-state intersection of S and *LM*, which retains precisely those strings in common to S and *LM*. The resulting automaton is a compact representation of each word ID sequence compatible with the I_hs and weighted by the stochastic backoff model's estimate of the probability of that word sequence. Finally, we find the most probable path in this automaton using the well-known single-source shortest-path algorithm in directed, acyclic graphs (labeled here with negative log probabilities) (Cormen et al., 1992). The word ID of the last transition is then the most likely candidate for the last word typed. We use the AT&T Finite-State Machine (FSM) Library for all the finite-state representations and operations described above (Mohri et al., 2000); FSM can be downloaded for non-commercial use at (Mohri et al., 1997).

2.3 Delays, Changes, and Bad Changes

We have just described how our algorithm works for $d = 0$. For $d = 1$, we make a preliminary estimate i for the kth word w_k when h reaches k according to the $d = 0$ strategy. When h becomes $k+1$, we again find the most probable path as above. We read off the word ID i' of the second last transition. If i' is not i, then we say that the kth word hypothesis *changed*. Clearly, we are interested in measuring how often this happens, since it is likely to distract the user. A particularly bothersome aspect of the $d = 1$ method is if the first hypothesis i is correct, but the second i' is wrong. We call this situation a *bad change*. We are interested in measuring the ratio of bad changes to all changes.

The case of $d = 2$ is analogous to $d = 1$.

3 Cluster Keyboards

We have studied the standard telephone keypad layout and four ways of squeezing the QWERTY layout onto fewer keys.

Q14 This keyboard layout is obtained by pairing adjacent keys. So the cluster keys are defined by the partitioning $\{Q, W\}$, $\{E, R\}$, $\{T, Y\}$, $\{U, I\}$, $\{O, P\}$, $\{A, S\}$, $\{D, F\}$, $\{G, H\}$, $\{J, K\}$, $\{L\}$, $\{Z, X\}$, $\{C, V\}$, $\{B, N\}$, $\{M\}$. Thus this keyboard has 14 letter keys. The layout is shown in Fig 1.

Q10 The Q10 layout is the result of merging the three rows of letter keys. Thus, it corresponds to the partitioning $\{Q, A, Z\}$, $\{W, S, X\}$, $\{E, D, C\}$, $\{R, F, V\}$, $\{T, G, B\}$, $\{Y, H, N\}$, $\{U, J, M\}$, $\{I, K\}$, $\{O, L\}$, $\{P\}$. It has 10 letter keys.

Q8 Q8 is obtained from Q10 by merging the four subsets that correspond to the middle of the keyboard. The layout thus is determined by the eight subsets $\{Q, A, Z\}$, $\{W, S, X\}$, $\{E, D, C\}$, $\{R, F, V, T, G, B\}$, $\{Y, H, N, U, J, M\}$, $\{I, K\}$, $\{O, L\}$, $\{P\}$.

Q5 Q5 is obtained from Q10 by overlaying the right half onto the left half. Thus the partitioning is $\{Q, A, Z, Y, H, N\}$, $\{W, S, X, U, J, M\}$, $\{E, D, C, I, K\}$, $\{R, F, V, O, L\}$, $\{T, G, B, P\}$, which yields five keys.

T9 This is the with keyboard described in Sect. 1. For fairness, we note that the 9 in T9 refers to the number of letter keys plus the spacebar key. Thus, T9 is most interestingly compared to Q8.

4 Experimental Setup

As previously mentioned, we have used two test sets: *NAB*, which consists of 54,265 sentences, and *Switchboard*, which consists of 2,475 sentences.

An example from NAB is: "in santa monica california well known hedge fund manager mark strome believes metals such as copper nickel and aluminum are poised for dramatic price rises". Thus, this corpus approximates formal writing.

An example from the Switchboard set is: "i don't want to go through that i mean i'm i'm seeing someone now ". Since this corpus stems from transcribed spoken language, the text structure is different from that of written language, even informal writing. Also, there are very many instances of hesitation and confirmation words that do not occur as frequently in writing, among them are "mhm', "-hum", "uh_huh", "uh-huh', "-hum", "mm", "oh", "uh", "um", "ah". To approximate informal writing, we have removed these words from our training sets.

We have used four different vocabulary sizes. The biggest one consists of 463k words (different inflected forms of the same word count as different words). It consists of this complete set of words found in the NAB corpus. We have also used vocabularies that consists of the most common 10k, 40k, and 160k words from the complete set.

Altogether, we have considered eleven different combinations of vocabulary size, order, and shrink factor. We have run all experiments with window size $\omega = 6$.

A first insight into the challenge of hypothesizing words can be derived from the table below, which summarizes for each of our layouts the percentage of words that have homographs, the maximal number of homographs that a word has, and the average number of homographs.

Layout	#Words w. h.graphs(%)	Max. # h.graphs	Avrg. h.graphs/ words(%)
Q14	14.84	13	0.27
Q10	19.96	20	0.51
Q8	32.61	82	1.84
Q5	54.25	86	6.84
T9	27.52	24	1.00

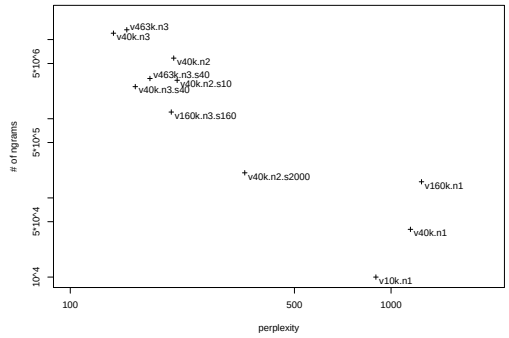

Figure 2: Perplexity v. n-gram size

These numbers are taken with respect to the 463k size vocabulary. The number of homographs is somewhat inflated by various minor anomalies in the corpus.

5 Results

Before summarizing our data, we discuss the out-of-vocabulary issue and the perplexity measure, which help understanding the fit of a language model with a test set.

5.1 Out-of-vocabulary Rates

For our two test sets, we have measured the out-of-vocabulary (OOV) rate to be:

text	10k	40k	160k	463k
NAB	5.58	1.42	.40	.23
Swbd	6.34	1.14	.65	.60

The 160k vocabulary covers both NAB and Switchboard test sets well. The OOV rate is about .5% for both. For the NAB test set, the full vocabulary results in a further drop in the OOV rate to less than one word out of every 400. The error rates in the rest of this section do not include OOV words.

5.2 Perplexity

We measure the quality of our language models in terms of their *perplexity* on the NAB test set. The perplexity of a language model over a test set t that contains N words is $2^{-H(t)}$ where $H(t)$ is the cross entropy $(-\log_2 Pr_{LM}(t))/N$. Thus, the perplexity is the geometric mean number of branches per word, given the information in the language model. In Fig. 2, we have plotted the number of n-grams in each model we have used as a function of its perplexity. A model is characterized by its vocabulary size, its n-gram order, and its shrink factor, if any. For example, the model v463k.n3.s40 has vocabulary size $v = 463k$, order $n = 3$, and shrink factor $s = 40$. From the plot, we see that it offers one of the lowest perplexities with less than 5 million n-grams. If we

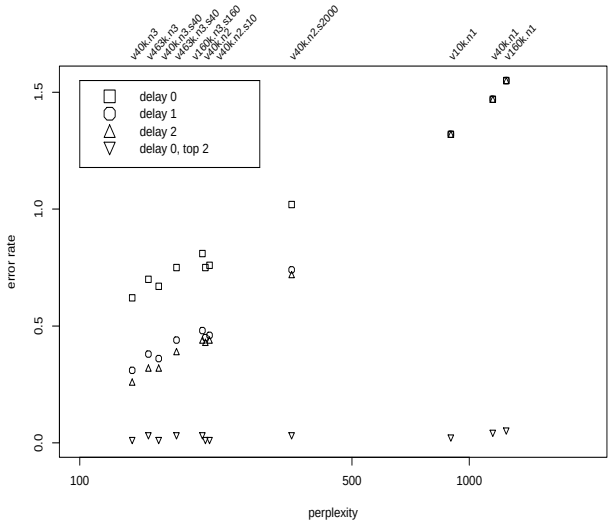

Figure 3: Q14

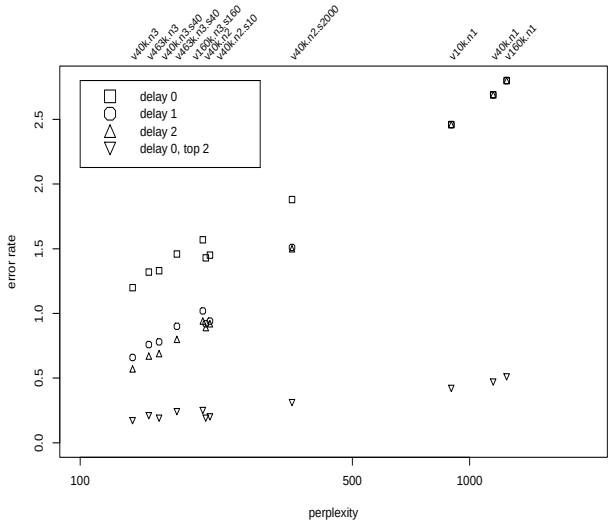

Figure 4: Q10

compare the trigram model v40.n3.s40 and the bigram model v40.n2.s10, we see that they are of almost the same size, but the trigram model is better at predicting words in the NAB test set. Naturally, the unigram models suffer from high perplexity, but they are about two orders of magnitude smaller than the other models we have considered, with the exception of the bigram model for vocabulary size 40k with shrink factor of 2000.

5.3 Error rates for cluster keyboards

We plot the error rates for each layout as a function of the perplexity of language models considered.

Q14 In Fig. 3, we have shown the results for the Q14, the QWERTY-based layout shown in Fig. 1. First, we notice that the top two hypotheses according to all the language models, even the unigram ones, virtually always contain the correct word. The user may type for several thousand words on average before the intended word, if present in dictionary, is not among the two top candidates. Second, we observe that the error rate for the top candidate is about 1.5% for the unigram models, but only .7% for the best trigram models, assuming $d = 0$ (when the user expects the correct word immediately after it has been typed). We also note that accepting a delay of $d = 1$ for the trigram models further cuts the error rates in half to about .4%. Third, we observe that further patience on the part of the user, namely for $d = 2$, carries a relatively modest payoff. For our best model, v40.n3, the error rate drops to only .3%. Fourth, we note the approximate linear relationship between log perplexity and error rates, for all four kinds of errors.

Q10 The 10-key QWERTY layout, Q10, results in error rates reported in Fig. 4. Most rates are about 50-80% higher, except for the rate of words not among the top two candidates, which is around .25% for the better models. This is about an order of magnitude worse than Q14.

Q8 The layout of the non-keyboard QWERTY increases error rates by another 50% or so according to Fig. 5. It is worth noticing that for the best language models, the error rate of the top candidate, assuming delay $d = 0$, is about 2% for our best model, whereas our unigram models yield error rates of about 4%. These numbers can be contrasted to the 3.5% figure that we quoted in Sect. 1.2.

Q5 In Fig. 6, the data for Q5 shows that with a good language model the correct word will be among the top two candidates for approximately 98.5% of all entries. This raises the interesting possibility that typing with just five keys might be almost as effective as on 26 keys. Only for one word out of 70 will a correction needing more than one key press be necessary. Since it reduces the error rate by 3/4, the use of trigram models for this keyboard seems called for.

T9 For T9, it is possible to reach an error rate below 1% if the second-best word is allowed to change. If not, the use of trigrams still cuts the error rate in half over the conventional technique of unigrams.

6 Additional results

It is of interest to see how well the language models do on tasks that are different from business

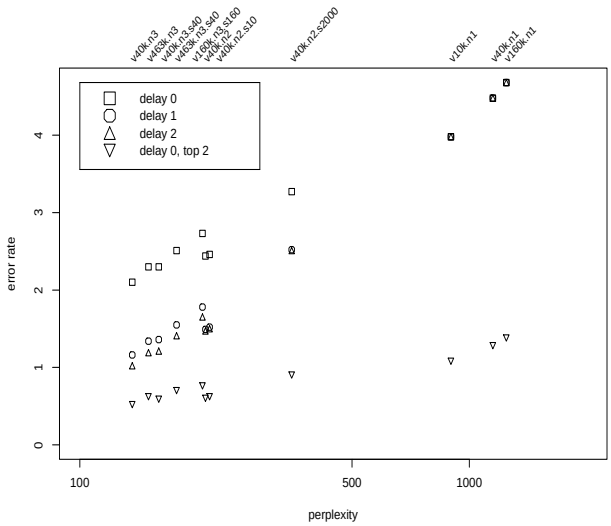

Figure 5: Q8

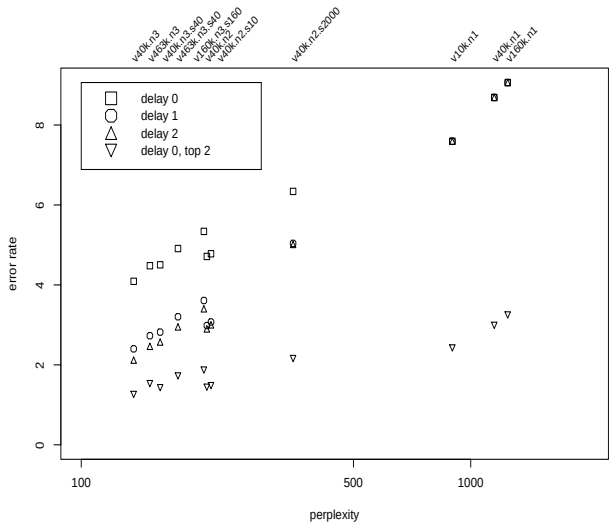

Figure 6: Q5

news. For this purpose, we have used the Switchboard test set modified as described in Sect. 4. Unfortunately, initials and abbreviations are normalized differently in the two test sets. Also, initials and abbreviations are prone to misrecognitions. For example, it is virtually impossible to guess initials correctly if entered via non-singleton cluster keys. Thus, we have corrected the scores so that we disregard all initials and abbreviations. This is a reasonable assumption from a human factors point of view: abbreviations will be so difficult to enter via the non-disambiguated cluster keys that the recommended method always is to use slower explicit entry.

Our language models do not reflect the ways numbers are entered. The problem is that the frequency of "eight" is grossly overestimated on the NAB corpus, since all numbers are spelled out as in "two hundred ninety eight". This has led to spurious misrecognitions of the word "right" which in some cluster layouts is homographic to "eight". We have adjusted the scores for this particular issue.

6.1 Task Portability

With these preliminaries, we study the quality of v463k.n3 on the NAB and Switchboard test sets with $d = 1$:

Error rate	Q14	Q10	Q8	Q5	T9
NAB (%)	.2	.6	1.0	2.3	.7
Swbd (%)	.4	1.5	2.7	4.7	1.0

We see that the error rates increase significantly when going from the NAB test set to the Switchboard set. Still, Q14 works well on the Switchboard test set, since the error rate remains negligible. But the doubling of the error rate to 4.7% for Q5 clearly makes it less usable. With T9, the error rate rises from .7% to 1.0%.

On NAB, we see that the elimination of initials and abbreviations has a small, but measurable effect on the accuracy of the better keyboards. For example, the error rate for Q14 reported in Fig. 3 is .38% and it drops to .24% when not considering initials and abbreviations. An inspection of the actual errors reveals that initials by far constitute the main problem. Our claim in Fig. 1 that Q14 can be used with an accuracy of up to 99.5% is based on this latter figure added to the OOV rate of .23%.

From our informal inspection of the test results, we have seen more cases where artifacts of the normalizations lead to common words being misrecognized. Thus, by further engineering language models, we believe that even more robust ones may be calculated.

6.2 Changes and bad changes for $d = 1, 2$

For v463k.n3, we have calculated the ratio of changes and bad changes, see Sect. 2.3. The table below explains how often changes occur and what percentage of changes are bad.

		Q14	Q10	Q8	Q5	T9
Changes	$d = 1$	0.50	0.99	1.68	3.13	1.25
(%)	$d = 2$	0.10	0.22	0.34	0.66	0.27
Bad changes	$d = 1$	17.95	19.02	16.41	16.62	18.54
(%)	$d = 2$	27.96	23.39	22.48	24.08	21.39

Thus for $d = 1$ and in the case of Q14, the user will be bothered by a bad change rate of .09%, one for each 1100 words entered.

7 Conclusion

We have demonstrated that n-gram models, for $n = 3$, have an important role to play for the entry of English text on cluster keyboards.

Our results in Sect. 6.1 indicate that Q14 is significantly more robust than T9. Also, Q14 allows

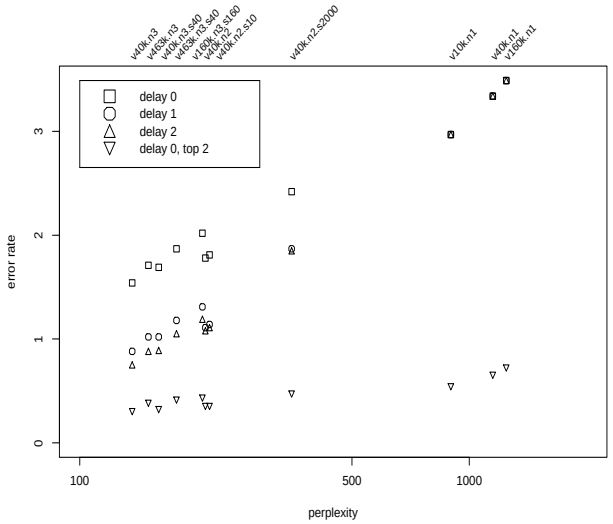

Figure 7: T9

a familiar layout and an easily understood disambiguation method, which uses two modifier keys for choosing the left and right letter on cluster keys, see Fig.1.

Also, as a substitute for spoken communication, the Q5 keyboard may enable disabled persons to communicate effectively using far fewer keys than 26. Compared to some earlier work, we speculate that this result may be turned into a distinct human factors advantage: we assume a fixed layout of keys which requires no interaction except for error situations.

Further studies must be carried out to reveal how text entry in practice is helped by dynamic, user-adaptable models.

Acknowledgements

We thank Anders S. Johansen for inspiring discussions and Don Hindle and Brian Roark for their help in building the language models.

References

Thomas Cormen, Charles Leiserson, and Ronald Rivest. 1992. *Introduction to Algorithms*. The MIT Press, Cambridge, MA.

M. D. Dunlop and A. Crossan. 2000. Predictive text entry methods for mobile phones. *Personal Technologies*, 4(2).

Mikel L. Forcada. 2001. Corpus-based stochastic finite-state predictive text entry for reduced keyboards: application to catalan. In *Procesamiento del Lenguaje Natural, XVII Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural*, volume 27, pages 65–70.

J. Godfrey, E. Holliman, and J. McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 517–520. ICASSP92.

Anders S. Johansen and John Hansen. 2002. Augmentative and alternative communication: The future of text on the move. In *Proceedings of 7th ERCIM Workshop, User Interfaces for all*, page 367 386.

S Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. of Acoustics Speech and Signal Processing*, 35(3):400–401.

Michael Kühn and Jörn Garbe. 2001. Predictive and highly ambiguous typing for a severely speech and motion impaired user. In *Universal Access In Human-Computer Interaction. Proc. of UAHCI 2001*. Lawrence Erlbaum Associates, August.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1997. General-purpose Finite-State Machine Software Tools. *http://www.research.att.com/sw/tools/fsm*, AT&T Labs – Research.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The Design Principles of a Weighted Finite-State Transducer Library. *Theoretical Computer Science*, 231:17–32, January.

Mikael Theory. Journal of Visual Languages, Computing Goldstein, Robert Book, Gunilla Alsiö, and Silvia Tessa. 1999. Non-keyboard QWERTY touch typing: a portable input interface for the mobile user. In *CHI '99*, pages 32–39. ACM Press.

D. Paul and J. Baker. 1992. The Design for the Wall Street Journal-based CSR Corpus. In *Proceedings of International Conference on Speech and Language Processing*, Kobe, Japan. ICSLP92.

Harald Rau and Steven S. Skiena. 1996. Dialing for documents: an experiment in information theory. *Journal of Visual Languages and Computing*, 7(1):79–95.

G. Riccardi, E. Bocchieri, and Roberto Pieraccini. 1995. Non-deterministic stochastic language models for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages I.237–I.240. ICASSP95, May.

Kristie Seymore and Roni Rosenfeld. 1996. Scalable Backoff Language Models. In *Proceedings of International Conference on Speech and Language Processing*, Philadelphia, Pennsylvania.

Miika Silfverberg, I. Scott MacKenzie, and Panu Korhonen. 2000. Predicting text entry speed on mobile phones. In *CHI '2000*, pages 9–16.

Kumiko Tanaka-Ishii, Yusuke Inutsuka, and Masato Takeichi. 2002. Entering text with a fourbutton device. In *COLING 2002. Proceedings of the 19th International Conference on Computational Linguistics*. Morgan Kaufman Publishers.

D. J. Ward and D. J. C. MacKay. 2002. Fast hands-free writing by gaze direction. *Nature*, (6900):838.