

# Informatics in Radiology (*infoRAD*)

## Radiology Report Entry with Automatic Phrase Completion Driven by Language Modeling<sup>1</sup>

John Eng, MD • Jason M. Eisner, PhD

### SUPPLEMENTAL MATERIAL

Additional formulas to supplement this article are available online at [radiographics.rsna.org/cgi/content/full/24/5/1493/DC1](http://radiographics.rsna.org/cgi/content/full/24/5/1493/DC1).

Keyboard entry or correction of radiology reports by radiologists and transcriptionists remains necessary in many settings despite advances in computerized speech recognition. A report entry system that implements an automated phrase completion feature based on language modeling was developed and tested. The special text editor uses context to predict the full word or phrase being typed, updating the displayed prediction after each keystroke. At any point, pressing the tab key inserts the predicted phrase without having to type the remaining characters of the phrase. Successive words of the phrase are predicted by a trigram language model. Phrase lengths are chosen to minimize the expected number of keystrokes as predicted by the language model. Operation is highly and automatically customized to each user. The language model was trained on 36,843 general radiography reports, which were consecutively generated and contained 1.48 million words. Performance was tested on 200 randomly selected reports outside of the training set. The phrase completion technique reduced the average number of keystrokes per report from 194 to 58; the average reduction factor was 3.3 (geometric mean) (95% confidence interval, 3.2–3.5). The algorithm significantly reduced the number of keystrokes required to generate a radiography report ( $P < .00005$ ).

©RSNA, 2004

**Abbreviations:** EOP = end of phrase, OOV = out of vocabulary

**Index terms:** Computers • Information management • Information management, systems • Radiology reporting systems

**RadioGraphics 2004;** 24:1493–1501 • **Published online** 10.1148/rg.245035197 • **Content Code:** HP

<sup>1</sup>From the Russell H. Morgan Department of Radiology and Radiological Science, Johns Hopkins University School of Medicine, Baltimore, Md (J.E.); and the Department of Computer Science, Johns Hopkins University Whiting School of Engineering, Baltimore (J.M.E.). Presented as an *infoRAD* exhibit at the 2002 RSNA scientific assembly. Received September 29, 2003; revision requested December 22 and received February 5, 2004; accepted March 17. Both authors have no financial relationships to disclose. **Address correspondence to** J.E., Department of Radiology, Johns Hopkins Hospital, Central Radiology Viewing Area, Rm 117, 600 N Wolfe St, Baltimore, MD 21287 (e-mail: [jeng@jhmi.edu](mailto:jeng@jhmi.edu)).

©RSNA, 2004

## Introduction

Keyboard entry of radiology reports is a necessity in most radiology practices, ranging from primary report entry by transcriptionists to report correction by radiologists. Advances in the development of speech recognition systems have made these systems practical in some settings, but some challenges remain. For example, in noisy environments such as a busy emergency department, speech recognition systems may be prone to errors caused by background noise. Yet bustling service areas such as the emergency department are often the ones that could benefit most from the rapid turnaround offered by automated reporting systems.

Speech recognition systems require the correction of recognition errors, which can distract the radiologist from looking at the images being interpreted. As a result, the time spent by the radiologist per case is greater for speech recognition systems than for traditional dictation (1,2). Recognizing accented speech or speech in noisy environments is particularly difficult. Another limitation of speech recognition systems is that they are not available for many languages other than English.

Given that keyboard text entry is a necessity in many environments, methods exist to improve the efficiency of keyboard entry. Automatic word completion and spelling checkers are commonly available in word processing programs. In some reporting systems, there is support for keyboard-activated macros and predefined ("canned") report templates. While helpful, these aids have some drawbacks, a major one being the substan-

tial effort that may be required to create, maintain, and memorize a concise yet generalizable system of macros and/or predefined reports.

A large part of the success of speech recognition systems derives from their reliance on language modeling, a computational method that involves statistical models for describing word sequences (3). Can language modeling also improve the speed of text entry when one is typing from a keyboard? The objective of the present work is to improve the efficiency of keyboard entry of radiology reports by implementing a context-sensitive phrase completion algorithm based on language modeling. While the implementation was done primarily with the radiologist user in mind, the program's main function is applicable to other types of users performing keyboard entry, such as transcriptionists.

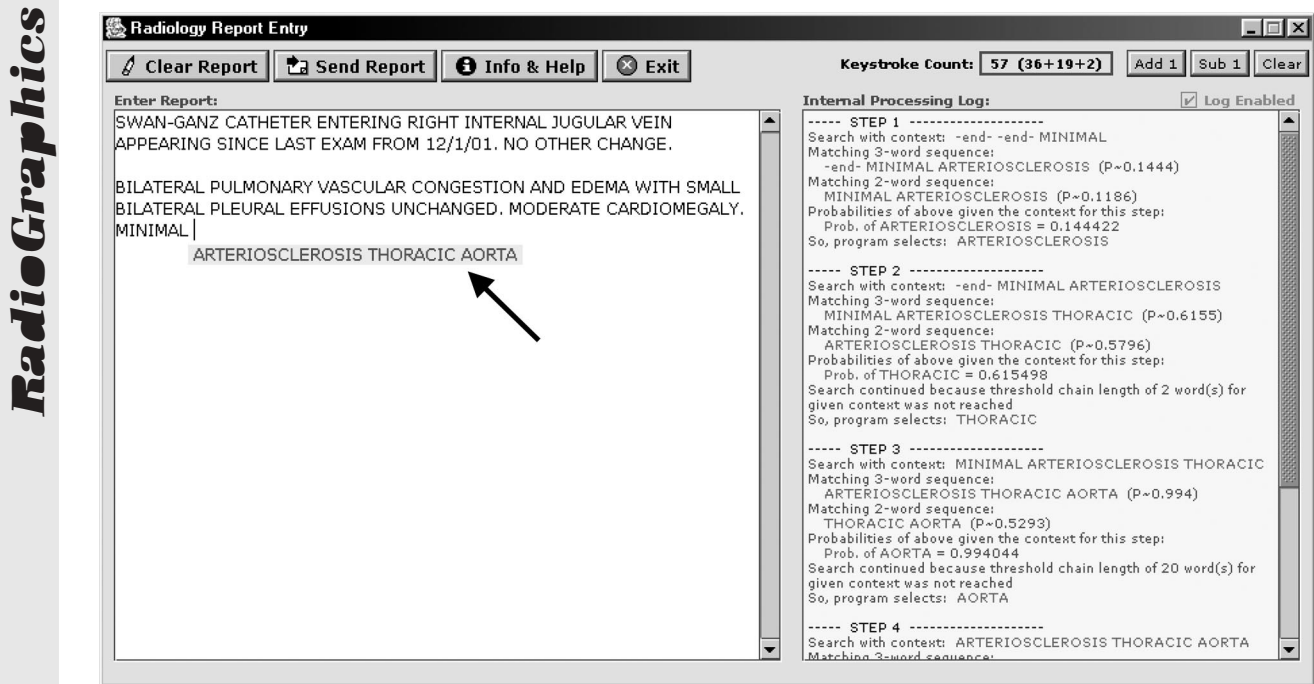
## Materials and Methods

A context-sensitive automatic phrase completion algorithm was implemented as a prototype computer program called ForeWord (Fig 1). ForeWord is a standalone application written by the first author in the Java programming language (version 1.4.1; Sun Microsystems, Palo Alto, Calif). The program has been tested on both the Mac OS 10.3 (Apple Computer, Cupertino, Calif) and Windows 2000 (Microsoft, Redmond, Wash) operating systems. As the user types the report, ForeWord continuously predicts the word(s) the user most likely intends to type next and places the word(s) inside a suggestion box underneath the current line. At any time, the user can insert the entire suggested phrase by typing the tab key without having to type the remaining characters of the phrase. If only the beginning of the suggested phrase is correct, the user can insert it one word at a time by typing the backtick key (above the tab key on most keyboards). If even the first suggested word is incorrect, the user can continue typing; with each additional letter, ForeWord will revise its suggested phrase to extend what has been typed so far. Therefore, ForeWord's computational task is to continuously choose the optimum words and optimum number of words to place in the suggested phrase.

ForeWord's phrase completion algorithm employs an  $N$ -gram language model (3). An  $N$ -gram is a group of  $N$  words. Therefore, a trigram is a group of three words, a bigram is a pair of words, and a unigram is a single word. ForeWord employs a trigram model ( $N = 3$ ), which is the most common model type and the one used in most speech recognition systems. The  $N$ -gram language model allows the calculation of word probabilities given the  $N - 1$  preceding words. The

### TAKE-HOME POINTS

- Automatic phrase completion driven by language modeling can significantly reduce the number of keystrokes necessary to type a report.
- Language modeling enables the automatic creation of phrase completion systems that are optimized for each individual user but require no memorization to use.
- Unlike the word completion function in word processors, phrase completion extends the prediction beyond one word up to an optimal phrase length.
- Constructing language models requires an adequate amount of sample text from each user.



**Figure 1.** Main display window of the ForeWord application for generation of radiology reports. As the report is typed in the text area on the left, the predicted word or phrase appears in the suggestion box (arrow) underneath the cursor. The suggestion box is continuously updated with each keystroke. The user presses the tab key to insert all the text from the suggestion box or the backtick key to insert one word of it at a time. In this prototype, a summary of the program's internal language processing is shown in the text area on the right.

calculation of conditional word probabilities (ie, the probability of a word,  $w_3$ , given the two preceding words,  $w_1$  and  $w_2$ ) is a key component in ForeWord's phrase completion algorithm. In ForeWord, these probabilities are estimated using the Katz backoff procedure with Witten-Bell discounting (3). The formulas for this procedure are available online at [radiographics.rsna.org/cgi/content/full/24/5/1493/DC1](http://radiographics.rsna.org/cgi/content/full/24/5/1493/DC1).

Like other  $N$ -gram language models, ForeWord's trigram language model is constructed from a sample of text, called the *corpus*, taken from the domain of the intended application. In this work, the corpus consisted of all general radiography reports generated by the first author in his clinical practice at a large academic medical institution from July 1997 to December 1999, a 30-month period. The reports were obtained in electronic form from the radiology information system. All header information was removed from the text, including patient identifiers, examination identifiers, and paragraph headings. All sentence punctuation was removed, including punctuation in the middle of sentences (eg, comma). Terminating punctuation marks (period, semicolon, colon, question mark, and exclamation mark) were replaced with a special "end of phrase" (EOP) token. The ends of paragraphs and reports were considered equivalent to the ends of sentences. Grouping of the text into paragraphs and

reports was otherwise ignored. For simplicity, all alphabetic characters were converted to uppercase. Words occurring fewer than 10 times in the corpus were replaced with a special "out of vocabulary" (OOV) token; this step eliminated most of the misspelled words from the corpus. Both the EOP and OOV tokens were subsequently treated as ordinary words. The corpus was separated into its component trigrams, bigrams, and unigrams using scripts written in the Perl programming language (version 5.6.1).

ForeWord's phrase completion algorithm is activated by every alphanumeric keystroke. Phrase completion is also activated again after inserting suggested text by typing the tab or backtick key. The phrase completion algorithm is composed of two stages: word completion and word chaining. Word completion guesses the rest of the word that the user is currently typing. Word chaining then guesses zero or more subsequent words to extend the prediction into a phrase. A major component of word chaining is the determination of the best chain length.

Both stages use the language model to guess the most probable word given the context of the two preceding words. Word completion chooses the most probable word beginning with the letters

## Keyboard Input



## Internal Calculations

Context	Most common next word, $w_3$ , that begins with "A"	Naive estimate: $c(w_1 w_2 w_3) / c(w_1 w_2)$ $= P_{\text{naive}}(w_3   w_1 w_2)$	ForeWord's estimate: $P_{\text{backoff}}(w_3   w_1 w_2)$
LEFT LATERAL	ABDOMEN	$17/296 = 0.057$	0.052
LATERAL	ANGULATION	$0/296 = 0.000$	0.003
∅	AND	$2/296 = 0.007$	0.006

## Program Response

LEFT      LATERAL      A  
 ABDOMEN  
 Suggested next word

**Figure 2.** Example of how the most likely word is determined during the word completion stage. The user has already entered the words *LEFT LATERAL* and intends to type the subsequent word *ABDOMEN*. The user starts by typing the letter *A*. ForeWord suggests the word starting with *A* that it estimates to be most probable in the context *LEFT LATERAL*. This will be one of the words *ABDOMEN*, *ANGULATION*, and *AND*, which were most commonly found in the corpus after *LEFT LATERAL*, after *LATERAL*, and in isolation, respectively. ForeWord grants some probability to words that were never seen in the context *LEFT LATERAL* at the expense of words that were seen in that context.

that the user has typed so far (Fig 2). In word chaining, the language model is used to form a hypothetical word chain in which each successive word is guessed to be the most likely word given the two preceding words, one or both of which are guesses themselves. Whenever a new character is typed, the entire word chain is predicted again from the beginning, going through both stages, and the display is updated to suggest the new chain. Because punctuation marks were removed from the corpus, they do not appear in the suggested phrase and must be typed when needed.

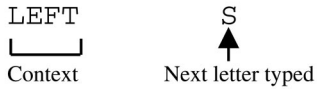
How long should the predicted word chain be? In principle, ForeWord could extend it indefinitely. However, chain prediction is essentially an all-or-none gamble. If the chain is fully correct, then the user can insert it into the report all at once by pressing the tab key. If the chain is incorrect, then the user must fall back to a single-word mode in which words are inserted one at a time with the backtick key. Thus, longer chains save more keystrokes if correct but are less likely to be correct.

It turns out that given the context formed by the last two words  $w_1 w_2$  in the text extended with the current chain, a *threshold chain length*  $L(w_1 w_2)$  can be determined. If the current chain contains

at least  $L(w_1 w_2)$  words, then it is not worth the gamble to extend it any further (Fig 3). If it contains fewer than  $L(w_1 w_2)$  words, then it should be extended by the most likely next word  $w_3$  (provided that this word is not the EOP or OOV token). In the latter case, the chaining decision procedure is repeated, to decide whether to add a further word  $w_4$ ; this time it compares the length of the newly extended chain against the new threshold  $L(w_2 w_3)$ . Chaining is repeated until the threshold chain length is reached or until the most likely next word is the EOP or OOV token.

How is the threshold chain length  $L(w_1 w_2)$  determined? The current  $n$ -word chain ending in  $w_1 w_2$  is assumed to be entirely correct, since otherwise the user's behavior is not affected by whether ForeWord extends it further. The *infinite chain* is the suggested phrase that would be predicted if we continued extending the current chain forever using the language model. Next, we define the *maximal correct chain*, which is the longest prefix of the infinite chain that is actually correct according to the user's intent. Ideally, ForeWord would suggest the maximal correct chain to the user, but lacking telepathy, it may often suggest something shorter or longer. Finally, we define the function  $K(n, w_1 w_2)$ , which is the *expected* number of tab and backtick keystrokes needed to type the rest of the *unknown* maximal correct chain (after which the user must resort to typing

**Keyboard Input**

LEFT                      S  
  
 Context                      Next letter typed

**Internal Calculations**

Current chain	Current chain length	Threshold chain length, $L(w_1w_2)$	Extend further?
SUBCLAVIAN	1	$L(\text{LEFT SUBCLAVIAN}) = 4$	Yes
SUBCLAVIAN VENOUS	2	$L(\text{SUBCLAVIAN VENOUS}) = 7$	Yes
SUBCLAVIAN VENOUS CATHETER	3	$L(\text{VENOUS CATHETER}) = 2$	No

**Program Response**

LEFT                      S  
 SUBCLAVIAN VENOUS CATHETER  
 Suggested phrase

**Figure 3.** Example of the word chaining stage. In this example, the user has already entered the word *LEFT* and wishes to continue with a phrase starting with *S*. The most likely chain of words is found by the word chaining algorithm, and the series is truncated when the threshold chain length is reached or exceeded. Note that the threshold chain length changes with each chained word. Among other considerations, it tends to be larger when ForeWord is confident about predicting the following word. In this example, ForeWord is very confident that *SUBCLAVIAN VENOUS* should be extended with *CATHETER*.

more letters). More precisely, it is the expected cost of typing the maximal correct chain. Currently, we consider each tab keystroke to have a cost of  $C_{tab} = 1$ , and each backtick keystroke to have a greater cost of  $C_{backtick} = 1.4$  since it is annoying to accept only part of a phrase and since the backtick key is an unfamiliar key. ForeWord applies an optimal policy, extending the current chain if and only if this will lead to a lower expected cost  $K(n, w_1w_2)$ .

The expected cost  $K(n, w_1w_2)$  is mathematically defined as follows:

$$\sum_S \Pr \left( \begin{array}{l} S \text{ is maximal correct chain} \end{array} \middle| \begin{array}{l} \text{current chain has length } n, \\ \text{ends in the two words } w_1w_2, \\ \text{and is correct by assumption} \end{array} \right) \cdot (\text{cost of typing } S),$$

where  $S$  is a sequence of words ranging over all finite prefixes of the infinite chain, having lengths  $n, n+1, n+2, \dots$ . The probability of  $S$  is determined by the language model, but the number of keystrokes needed to type all of  $S$  is determined by ForeWord's own word chaining policy, which may not suggest all of  $S$  as a single phrase or may incorrectly suggest a chain longer than  $S$ . This means that  $K(n, w_1w_2)$  depends on ForeWord's chaining policy. As ForeWord uses an *optimal*

chaining policy that is defined to depend in turn on  $K(n, w_1w_2)$ , both  $K$  and the policy must be found by iteratively solving a circular system of equations.

By assumption, the current chain is correct so far. Let  $E_{extend}$  be the expected number of keystrokes to type the maximal correct chain if the current chain were extended with (at least) the most likely candidate word  $w_3$ :

$$E_{extend}(n, w_1w_2) = P(w_3 | w_1w_2)K(n+1, w_2w_3) + [1 - P(w_3 | w_1w_2)]nC_{backtick},$$

where  $P(w_3 | w_1w_2)$  is the probability of  $w_3$  given the word sequence  $w_1w_2$ . Note that  $E_{extend}$  is the sum of two terms: the expected keystrokes if  $w_3$  is actually the intended word and the expected keystrokes if  $w_3$  is not the intended word. In the former case,  $K(n+1, w_2w_3)$  takes into account any further extensions of the chain. In the latter case, ForeWord has predicted too much: The user must type the backtick  $n$  times to accept the words in the suggested phrase individually up to, but not including, the last one ( $w_3$ ).

Similarly, we define  $E_{noextends}$  the expected number of keystrokes needed to type the maximal

correct chain if the candidate word  $w_3$  were *not* added to the current chain:

$$E_{noextend}(w_1w_2) = C_{tab} + P(w_3 | w_1w_2)K(1, w_2w_3).$$

Here, the suggested phrase is just the current chain, which is correct by assumption, so the user will type the tab key to accept it at a cost of  $C_{tab}$ . ForeWord will then immediately make a new prediction beginning with the most likely next word,  $w_3$ . The second term accounts for the case in which  $w_3$  is in fact correct. In this case, ForeWord's original prediction was too short; the tab key did not accept the entire maximal correct chain, and the user must now type  $K(1, w_2w_3)$  additional keystrokes to accept the rest of it (starting with  $w_3$ ).

$K$  is updated iteratively by recalculating the optimal policy given the  $E$  values:

$$K'(n, w_1w_2) = \min[E_{extend}(n, w_1w_2), E_{noextend}(w_1w_2)],$$

where  $\min(x, y)$  denotes the smaller of  $x$  and  $y$ .  $K$  is then updated to equal  $K'$ , and new values of  $E_{extend}$  and  $E_{noextend}$  are calculated based on this new  $K$ . Iteration is continued until

$$|K'(n, w_1w_2) - K(n, w_1w_2)| < \epsilon$$

for all  $n$  and for all  $w_1w_2$  in the corpus. For purposes of this work, the tolerance  $\epsilon$  was set to 0.001 and the initial values of all  $K(n, w_1w_2)$  were set to 0. After  $K$ ,  $E_{extend}$ , and  $E_{noextend}$  converge,  $L(w_1w_2)$  is simply defined as the largest  $n$  for which  $E_{extend}(n, w_1w_2) < E_{noextend}(w_1w_2)$ .

Only  $L(w_1w_2)$  is required for ForeWord to process user keystrokes; the iterative calculations need to be performed only once for a given corpus. Although  $n$  is unbounded, the values on each iteration can be calculated in finite time: For each  $w_1w_2$ , first compute and store  $E_{noextend}(w_1w_2)$ , then compute and store  $E_{extend}(n, w_1w_2)$  and  $K'(n, w_1w_2)$  for  $n = \{1, 2, 3, \dots\}$  successively until  $E_{extend}(n, w_1w_2)$  exceeds  $E_{noextend}(w_1w_2)$ . Nothing need be explicitly computed or stored for higher values of  $n$ .

To test the effectiveness of the automatic phrase completion algorithm, 200 reports outside of the corpus were randomly chosen from general radiography reports generated by the first author from January 2000 to June 2000, the 6-month period immediately following the period from which the corpus was derived. The numbers of keystrokes required to generate each of the 200

reports by using ForeWord were recorded by a typist simulation function within ForeWord. The typist simulation function is completely independent of ForeWord's phrase completion algorithm. These keystroke numbers were compared to the number of characters in each report, which is equal to the number of keystrokes that would be required to generate each report without the aid of phrase completion.

The keystroke counts in the absence and presence of phrase completion were compared by calculating the logarithm of their ratio and performing a  $t$  test of this log-ratio. The null hypothesis of the  $t$  test was that the log-ratio of the keystroke counts is 0 (corresponding to a ratio of 1). A ratio was chosen instead of a difference so that the comparison would be independent of report length. The logarithm function was applied to transform the ratios into a data set that was normally distributed. With 200 test reports and assuming an estimated standard deviation of the log-ratio to be 0.2 (from preliminary data not shown), the statistical power to detect a log-ratio of at least 0.046 (corresponding to a 5% reduction of keystroke count) was 0.9 at the .05 level of significance.

In addition, a paired  $t$  test was employed to compare this log-ratio to that obtained with single word completion only (word chaining disabled). Normality of the distribution of the log-ratio values was examined with the Shapiro-Wilk  $W$  test. All statistical analyses were performed with the Stata program (version 8; StataCorp, College Station, Tex). Means  $\pm$  standard deviation are reported along with a 95% confidence interval where appropriate.

## Results

The corpus contained 36,843 general radiography reports with a total of 10,389,657 characters forming a total of 1,476,671 words. This corpus contained 1965 unique unigrams, 39,184 unique bigrams, and 139,864 unique trigrams. The reports were composed of the following mixture of radiographic examination types: 73% chest, 20% musculoskeletal, 5% gastrointestinal or genitourinary, and 2% miscellaneous.

For the 200 randomly chosen test reports, which did not appear in the corpus, the mean number of keystrokes without phrase completion (ie, characters of text) was  $194 \pm 103$  (95% confidence interval, 180–208). With automatic phrase completion, the mean number of keystrokes was reduced to  $58 \pm 33$  (95% confidence interval, 54–63). The mean log-ratio was  $1.20 \pm 0.27$  (95% confidence interval, 1.17–1.24), which was significantly different from 0 ( $P < .00005$ ).

SWAN-GANZ CATHETER ENTERING RIGHT INTERNAL  
JUGULAR VEIN APPEARING SINCE LAST EXAM FROM  
12/1/01. NO OTHER CHANGE. BILATERAL  
PULMONARY VASCULAR CONGESTION AND EDEMA  
WITH SMALL BILATERAL PLEURAL EFFUSIONS  
UNCHANGED. MODERATE CARDIOMEGALY. MINIMAL  
ARTERIOSCLEROSIS THORACIC AORTA. NO OTHER  
ABNORMALITY.

**Figure 4.** Example of how automated phrase completion works on a sample report that is not in the corpus. The shaded characters are those typed by the user; the underlined characters correspond to pressing of the tab key to insert the suggested word(s); and the double-underlined characters correspond to pressing of the backtick key to insert a portion of the suggested phrase. All other characters were inserted by automated phrase completion. ForeWord was particularly effective on this report, reducing the number of keystrokes by a factor of 4.7 (64 keystrokes to type 280 characters). Punctuation and out-of-vocabulary (OOV) text such as “12/1/01” must be manually typed in full because they are not part of the corpus. In this example, the user had to type *SW* to trigger *SWAN-GANZ CATHETER* because just typing *S* produced *SMALL* as the suggested phrase. Two backticks were typed to accept *INTERNAL JUGULAR* because the full suggested phrase was *INTERNAL JUGULAR VENOUS CATHETER*. When the user typed *M* for *MINIMAL*, the optimum chain length was one word. After accepting *MINIMAL*, ForeWord correctly predicted the most likely continuation, *ARTERIOSCLEROSIS THORACIC AORTA*, without further typing.

After converting to a ratio, this result corresponds to a geometric mean keystroke reduction factor (ratio) of 3.3 (95% confidence interval, 3.2–3.5). The log-ratio values in these calculations were found to be normally distributed according to the Shapiro-Wilk *W* test.

An average of  $16 \pm 9$  (95% confidence interval, 15–17) keystrokes per report were the tab key (28% [16 of 58]). An average of  $1.8 \pm 1.6$  (95% confidence interval, 1.6–2.0) keystrokes per report were the backtick key (3% [1.8 of 58]). The performance of ForeWord on a sample report outside of the corpus is depicted in Figure 4.

With word chaining disabled (and only single word completion enabled), the mean number of keystrokes increased to  $66 \pm 36$  (95% confidence interval, 61–71), an increase of 8 keystrokes per report on average. The mean number of tab keystrokes increased to  $26 \pm 14$  (95% confidence interval, 24–28), accounting for the 8-keystroke difference plus the absence of backtick keystrokes (necessary only in word chaining). Disabling word chaining resulted in a significantly lower keystroke reduction factor of 2.9 (geometric mean; 95% confidence interval, 2.8–3.0) ( $P < .00005$ ).

## Discussion

The ForeWord program demonstrates that a phrase completion algorithm driven by a language model can reduce the number of keystrokes by a factor of 3.3 in a radiology report domain. To the extent that keystroke count is correlated with transcription speed, the keystroke reduction implies that report generation with ForeWord's phrase completion algorithm would be faster than unassisted transcription. In addition to potentially increasing the typing speed, the reduced number of keystrokes also diminishes the likelihood of typographical errors. Furthermore, typing is easier because many of the keystrokes (28% in this study) are the same key—the tab key.

Alternative methods for speeding up the keyboard entry of text include the automatic word completion function found in many word processors and the support for report templates found in some transcription systems. Unlike these alternative methods, the language model approach does not require personal effort to create and memorize a system of predefined text. Instead, the process of creating the language model from a corpus of text is purely computational. All that is required is a sample of text from each individual user. From each user's corpus, the language model approach automatically generates a phrase completion system that is optimized for that user.

Aside from its use in speech recognition and other applications, language modeling has been previously employed for text completion in a few diverse applications where ordinary typing is difficult. Prominent examples include text entry for mobile computing devices, such as wireless telephones and personal digital assistants (4), and text entry by individuals with physical or learning disabilities (5,6). These applications use language modeling to predict individual letters (4,5) or single words (6).

While based on similar language modeling methods, ForeWord can make variable-length predictions: Its word chaining algorithm is able to extend the prediction beyond one word. We have found only two previous text completion projects with this capability, perhaps because it is mainly useful in low-entropy domains such as radiology reports. ForeWord's novel contribution here is its optimal determination of chain length. The Predict and Reactive Keyboard systems for typing computer commands (7) employ essentially the same chaining algorithm as we do, but always extend the prediction all the way to the right edge of the screen; keys like our backtick, or mouse

clicks, are used to indicate how much of the phrase to accept. The TransType system for entering translations of a known text (8) uses a more sophisticated chaining algorithm, which recognizes (unlike ForeWord) that the most probable three-word chain may not be an extension of the most probable two-word chain. However, TransType always chooses the chain length with the greatest expected immediate cost savings. By contrast, ForeWord also takes into account the expected effect on cost savings at subsequent steps.

The average number of keystrokes required to generate a character (keystrokes per character [KSPC]) has been suggested as a measure of text entry performance for mobile computing devices (9). Such devices (eg, wireless telephones) typically have a limited keypad, which may require multiple keystrokes to generate a single character ( $KSPC > 1$ ). By using word and phrase completion algorithms, it is possible to achieve a KSPC of less than 1. For example, ForeWord's keystroke reduction factor of 3.3 corresponds to a KSPC of 0.30. Word prediction algorithms generally have a KSPC of approximately 0.5 (9). ForeWord's greater performance may be due to the capability of predicting multiple-word phrases, as evidenced by its poorer performance ( $KSPC = 0.34$ ) when word chaining is disabled. However, even without word chaining, ForeWord's performance is better than a KSPC of 0.5; this is perhaps attributable to the focused domain of the application (radiology reporting), which makes the next word more predictable.

Since the *N*-gram language model is purely statistical and does not rely on any type of semantic analysis, it should work equally well with all languages that are based on an alphabet. Therefore, ForeWord would require few, if any, modifications to work with other languages. Of course, corpora in those languages would still be required to construct the language model.

The user-specificity of language modeling is obviously an advantage for the user. However, this specificity also means that a text generation

system driven by language modeling will perform poorly for a new user who has no corpus known to the system. The system cannot be used "out of the box" without an appropriate corpus of text. The availability of text is arguably the most significant limitation of language modeling applications in radiology. Even though radiology reports are commonly transcribed and stored in electronic form in radiology information systems, these systems are unlikely to have reasonably easy functions for bulk export of reports in electronic form. As radiology information systems continue to develop, we hope that the text of the reports does not continue to be stored inside inaccessible proprietary database structures.

Although language modeling can make keyboard text entry faster, it is not known how the resulting speed compares with alternatives such as speech recognition. Further work is required to examine the performance of ForeWord in a live clinical setting with respect to speech recognition and conventional dictation. Such work would require the direct measurement of the time required to generate reports with each method. Also needed are clinical studies with a variety of different users to determine the relationship between reporting style and effectiveness of the phrase completion algorithm. As apparent in Figure 4, the first author's reporting style is telegraphic, relying on short phrases rather than complete sentences. Language models derived from users with a more narrative reporting style may be less successful than those derived from users with a more concise telegraphic style.

While it is natural to consider ForeWord and speech recognition as two text generation methods to be compared, a potentially effective application of ForeWord may be one in tandem with a speech recognition system. Reports generated by speech recognition lack misspellings and other visually obvious cues that indicate errors. This increases the difficulty of proofreading. ForeWord's language model could be applied to the output of a speech recognition system to calculate the conditional probability of each word and highlight all words below a certain probability threshold. The highlighting of low-probability words could aid proofreading. In effect, the

speech recognition system's general-purpose language model would be augmented with ForeWord's user-specific language model.

The number of keystrokes is an incomplete measure of system efficiency. Beside simple keystroke count, other factors can affect typing speed, such as the physical distance between successive keys, the frequency of unfamiliar keys, and the cognitive load associated with choosing particular keys. The unfamiliarity and cognitive load factors are addressed somewhat in ForeWord's cost-minimizing word chaining algorithm: The backtick key is considered to have higher cost than the tab key ( $C_{backtick} > C_{tab}$ ), causing ForeWord to suggest somewhat shorter phrases so as to reduce the need for backticks. Further clinical testing is required to examine the relationship between keystroke count, typing speed, and user satisfaction. For example, users may prefer a less "annoying" text editor even if it slightly increases the keystroke count or slightly decreases the directly measured typing speed. Such testing could provide a better choice of  $C_{backtick}$ . Key-specific or contextual costs for alphanumeric keys could also be introduced, allowing revised word completion and word chaining algorithms that could save the user trouble by suggesting a long or hard-to-type word, even when another word is slightly more probable. In principle, ForeWord could estimate these key-specific costs as TransType does (8), by measuring how long it takes for the user to type each key.

The current prototype implements a static language model derived from a static corpus. A final implementation could benefit from a dynamic language model whose corpus is continuously updated with the text of each new report generated by the system. This technique would allow the system to adapt gradually to a new user, with performance rising rapidly at first and then leveling off as the language model's probability estimates converge to the truth. An individual user's language model could also be improved by adding scaled-down  $N$ -gram counts obtained from other users' reports. Such a *mixture model* will perform better in contexts where the individual user's data are sparse. Further improvement of ForeWord would include the prediction of capitalization and punctuation so that reports do not appear in all uppercase letters.

## Summary

A system of automatic phrase completion driven by language modeling can substantially reduce the number of keystrokes needed to generate radiology reports. The key requirement is the availability of appropriate text corpora to create the underlying language models. Language modeling possesses a key advantage over other text generation aids because the creation of individualized language models from a corpus can be completely automated. A novel feature of the system described is the prediction of multiword phrases of optimal length.

## References

1. Hayt DB, Alexander S. The pros and cons of implementing PACS and speech recognition systems. *J Digit Imaging* 2001; 14:149–157.
2. Gale B, Saffriel Y, Lukban A, Kalowitz J, Fleischer J, Gordon D. Radiology report production times: voice recognition vs. transcription. *Radiol Manage* 2001; 23(2):18–22.
3. Jurafsky D, Martin JH. *Speech and language processing*. Upper Saddle River, NJ: Prentice Hall, 2000; 191–234.
4. MacKenzie IS, Soukoreff RW. Text entry for mobile computing: models and methods, theory and practice. *Hum Comput Interaction* 2002; 17:147–198.
5. Ward DJ, MacKay DJ. Fast hands-free writing by gaze direction. *Nature* 2002; 418:838.
6. Nantais T, Shein F, Johansson M. Efficacy of the word prediction algorithm in WordQ. In: *Proceedings of the 24th Annual Conference of the Rehabilitation Engineering and Assistive Technology Society of North America*. Arlington, Va: Rehabilitation Engineering and Assistive Technology Society of North America, 2001; 77–79.
7. Darragh JJ, Witten IH. *The reactive keyboard*. Cambridge, England: Cambridge University Press, 1992.
8. Foster G, Langlais P, Lapalme G. User-friendly text prediction for translators. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. East Stroudsburg, Pa: Association for Computational Linguistics, 2002; 148–155.
9. MacKenzie IS. KSPC (keystrokes per character) as a characteristic of text entry techniques. In: *Proceedings of the Fourth International Symposium on Human-Computer Interaction with Mobile Devices*. Heidelberg, Germany: Springer-Verlag, 2002; 195–210.