IMPROVING THE USER INTERFACE OF DICTATION SOFTWARE

Ben Kraal, Michael Wagner & Penny Collings

Human-Computer Communication Laboratory,
School of Computing, University of Canberra

ABSTRACT: Some research suggests that speech dictation software is hard for novices to use and recommends that it be adapted to their preferred interaction style. Older research into how novice users approach new software can provide insights into the preconceptions that novices have about all software and the ways in which dictation software can be redesigned to be aware of these preconceptions. The tendency of novice users to persist with spoken error correction illustrates that novices have an incorrect mental model of dictation that is closely related to their existing mental model of typing. This paper suggests that dictation software should be redesigned to make clear the differences in use between typing and dictation.

INTRODUCTION

A series of studies into the use of commercial speech recognition dictation systems (Halverson, Horn et al. 1999; Karat, Halverson et al. 1999; Karat, Horn et al. 2000) found that inexperienced users of such software tend to correct errors using spoken methods. The same study showed that subjects who use dictation software for an extended period of time (about 20 hours) tend to augment spoken correction with multimodal correction and keyboard correction. The same study also showed that expert users of dictation software almost totally abandon voice correction in favour of multimodal and keyboard correction strategies. Halverson, Horn et al made three recommendations (1999). First, that dictation software be adapted to the novice users' correction strategies by increasing recognition accuracy for re-dictation. Second that dictation software be adapted to novice users' preference for using spoken corrections. Third, recognise that novice users tend to stay with one particular error correction strategy at the expense of other more efficient or reliable strategies.

The second and third recommendations are the basis for this paper and its contention that novice or inexperienced dictation users' problems with error correction do not only reveal that correction by voice is difficult and needs improvement but also that current dictation software almost forces novice users to adopt poor interaction techniques.

A study by Carroll and Rosson (1987) into word processor users showed that the users were hampered in their learning of the software by their previous experience with other software, or as was the case in 1987, typewriters. This can also be applied to dictation software as many users approach dictation as if it were the same as typing; however there are some important differences in the way that word processing software is used and the way that dictation should be performed.

NOVICES' STRATEGIES FOR ERROR CORRECTION

The tendency for novices to persist with one strategy (for error correction) is not unique to dictation systems. Carroll and Rosson (1987) described this phenomenon with regards to novice users and word processing software. The 'paradox of the active user' describes how users tend to rely on one method for achieving any goal instead of investigating alternatives that may prove more efficient. The paradox arises because users are motivated to produce content using the software but are unwilling to learn about the software to increase their productivity. This is known as the 'productivity paradox' or the end-product focus. The end-product focus means that a user's primary goal is creating a document, at the expense of spending time learning about the system even when such learning may increase overall productivity. What users already know about other software may inhibit their learning about the new software – the 'assimilation paradox' or assimilation bias. The assimilation bias leads users to find misleading similarities between familiar and new software, leading to mistaken comparisons and even preventing users from seeing new possibilities for the new software.

The paradox can be seen by comparing the novice and extended-use participants from the study in Karat, Halverson et al (1999). The novice users persisted in using voice correction even though they said they felt there was a better way of correcting errors. They said they figured that it involved integrating speech with the keyboard but they did not know how to go about it (Halverson, Horn et al.

1999). In the same study the extended-use subjects who changed their error correction strategy to integrating speech and keyboard increased their rate of corrected words per minute compared with the novices. This focus on production might have been exacerbated by the experimental conditions that were present for these users. That is, the experimenters might have unduly influenced the subjects to use spoken correction because, as stated in Karat, Halverson et al (1999), the users were taught how to correct by voice before beginning the experimental tasks.

Karat, Halverson et al also noted that subjects used, or adapted to dictation, the same correction strategies they used for keyboard. The keyboard correction strategies were: (1) backspace over error and re-type, (2) select error and re-type and (3) dialog strategies (find, search and replace, etc). The speech strategies used by novices were very similar: (1) backspacing (using commands similar to SCRATCH or UNDO) and re-dictating, (2) selecting (using SELECT or similar) and re-dictating and (3) using correction dialogues.

Instead of recommending, as Halverson, Horn et al (1999) did that dictation software adapt to novice users' interaction styles, this paper recommends that dictation software tackle poor interaction and encourage users to learn more efficient multimodal or keyboard error correction strategies.

THE PRODUCTION PARADOX

Almost all users of computers experience the production paradox – it is the desire to start to use a piece of software before, or even instead of, learning about it. The production paradox is a problem for dictation software because novice users want to dictate the same way that they type. It is not usual for users to proofread and edit after they have produced a large block of content. As Karat, Halverson et al (1999) note, it is typical for users of word processing software to edit 'inline', that is, as they type. However, this interaction paradigm for keyboard-based word processors would seem to be inappropriate for dictation software because expert users abandon inline voice correction for a mixed mode correction strategy (Karat, Horn et al. 2000).

It might be possible that novice users who do not discover the mixed mode correction strategy abandon the dictation software altogether in favour of the more familiar keyboard. Further, it might be that any user who persists to discover the mixed mode correction strategy has some imperative to do so, such as an injury that prevents the use of a keyboard (or the need to use dictation software for a minimum of 20 hours).

So, it would seem that a stumbling block for novice users of dictation software is the interaction techniques that they have learnt from other software, at least with respect to error correction.

THE ASSIMILATION PARADOX

Carroll and Rosson (1987) say that instead of assuming novice users are blank slates they should be considered as experts in non-computer domains. When using dictation software it is probably more correct to describe novice users as experts in computer word-processing and novices in dictation. This distinction between the purpose and the interaction paradigm is where many novice users would seem to be caught and it is in this area that the most work needs to be done to adapt the dictation interface to (a) allow users to be productive and (b) understand the difference between dictation software and word-processing.

In typical keyboard-based word-processing it is quite easy for a user to 'feel' when an error is made. As typing allows users to see the creation of text letter-by-letter, errors can be recognised before entire words are formed. Dictation allows the user to 'feel' whole-word errors, for example stutters, hesitations and mispronunciations but does not always give the user the ability to 'feel' partial word errors. Instead, the software uses statistical modelling to construct whole words and phrases. This can, and does, lead to recognition errors on behalf of the software that the user has no control over. In a keyboarding error correction paradigm the user will typically correct errors as they occur, inline, as text is created. Halverson, Horn et al's (1999) study showed that spoken inline error corrections frequently led to cascades of errors as the attempts to correct by voice were themselves misrecognised, forcing users to correct their attempted corrections! These cascades of errors occur because dictation software is much better at recognising phrases and sentences than single words due to the nature of how words are recognised, typically with n-gram modelling. The frequency of

error cascades while correcting by voice probably lead more experienced dictation users to switch to a mixed mode correction strategy instead of persisting with a spoken inline method.

Compounding the difficulty in recognising errors while, or even after, dictating is that the errors are unlike those that occur when typing. Typical typing errors are spelling mistakes while typical speech recognition errors are inappropriate or misrecognised words. The added difficulty is that while the user considers the recognition errors to contain inappropriate words, the language model of the software (obviously) considers them to be entirely appropriate. Additionally, while a spell checker can 'watch' text as it is typed and show spelling errors, the language model of the dictation software is already 'watching' what is spoken and is unable to recognise the errors as they occur.

DESIGNING FOR THE PRODUCTION PARADOX

Carroll and Rosson (1987) state that they do not see the production paradox as a problem to be solved. Instead, the production paradox is a feature of how people approach tasks and is a factor to take into account when designing software.

By treating the user's desire to produce content as a starting point it is possible to design for that point of view while teaching the user about the system. Carroll and Rosson (1987) use the example of guided expert cards – small instruction cards that are task focussed. In contrast, a user manual is overly function oriented where the expert cards are task oriented, addressing specific tasks that a user may wish to complete.

In a dictation software system such task-oriented instructions might include 'How to begin dictation' which instead of only advising what icons to click would instead offer process-oriented advice such as: Be clear about the contents you wish to dictate, prepare a dictation plan, and choose the simple word, use simple sentences and have one idea to a paragraph (Roman 1971). It would also be possible for the task-oriented instructions to describe the benefits of dictating blocks of text and then proofreading instead of the more typing-oriented inline 'as-you-go' error correction method.

MITIGATING AND DESIGNING FOR THE ASSIMILATION PARADOX.

Like the production paradox, the assimilation paradox is a feature of how people learn and is not a fault to be removed.

One approach to assimilation is to design for it, or attack it directly. For example, In the case of dictation software this would mean illustrating the differences between dictation and typing instead of focussing on the similarities. One of the largest differences between creating content in a word processor and by dictation is the recommendation by dictation experts that users not look at the screen while dictating (Janal 1999; Newman 2001). This is intended to reduce the distraction that can be caused when waiting for the software to catch up with what was said, but it brings a side effect that users are prevented from correcting inline (because they can't see the errors) and must perform a proofreading pass. It is probably desirable to prevent users from correcting inline by voice as it may reduce error cascades. In the short term, before inline correction is greatly improved, it is probably more beneficial to user productivity to avoid the inline correction functionality, focus on content creation and locate errors with a post creation proofreading pass. In the longer term as inline correction becomes more reliable it will not be necessary to attack assimilation but more necessary to mitigate its undesirable effects.

There are many other ways to mitigate the effects of assimilation. Meeting novice or naïve user expectations is discounted by Carroll and Rosson (1987) because novices expectations are often incorrect, incomplete or based on a faulty understanding. This can be seen in dictation when users correct by voice by dictating single words. Halverson, Horn et al state that the success rate for single word corrections was 47% (Halverson, Horn et al. 1999). The novice users' voice correction rates could have possibly been improved simply by providing them with the information that multi-word dictation is significantly more likely to be recognised correctly than a single word.

Finally, it is possible to design for assimilation. This means that new ways to explain the information to novices must be found For example, recognition errors could be explained by saying that the computer is like a non-native speaker of English – occasionally the computer will mis-recognise words

that are new to it or not be familiar with the use of a particular word in a particular context. Designing for assimilation is very difficult as the designer must determine when it is appropriate to present assimilative information to the user (Carroll and Rosson 1987). In both the short and long term it is likely that users will benefit from being explicitly instructed that dictation is not word processing but a separate skill that can be learned. That dictation is a separate skill will need to be imparted implicitly and explicitly in dictation software. In the short term it is more likely that users will have to be instructed explicitly in the use of dictation software for their best advantage where in the long term dictation software may be able to accommodate user expectations while educating them about better dictation practice and so leading to greater dictation efficiency.

SUGGESTIONS FOR A NEW DICTATION INTERFACE

The considerations above lead to a number of suggestions for short and medium term improvements of the dictation human-computer interface.

SHORT TERM

These short term suggestions are not intended to require new recognition software development and are user interface suggestions only:

Hide spoken navigation commands. By 'hiding' support for spoken navigation of a dictated text, users will be encouraged to use the mouse or keyboard to navigate the document. This will prevent some of the cascades of error correction found by Karat, Halverson et al (1999) as well as encouraging the use of other error correction methods such as multimodal or keyboard correction. It is also possible that this might increase dictation performance because the recogniser is guaranteed that all words spoken are intended for dictation and command words do not have to be filtered out. This functionality is already available in software such as Dragon NaturallySpeaking v6.0 and other dictation systems.

Encourage multi-word re-dictation in error correction. This could be implemented by popping up a dialog box with the selected word and the surrounding words and asking the user to re-dictate the phrase. Upon re-dictating the phrase the dialog box would disappear without requiring a confirming click from the user.

Provide task-oriented instructions for dictation within dictation software. Instructions on how to improve dictation that take into account how the software actually works and are readily accessible will be more likely to encourage users to adapt their work practises from typing-centric to dictation-centric methods.

MEDIUM AND LONG TERM

Longer term measures for improving dictation interfaces could include modifying how the language model works and is adapted to the user. Specific user interface issues might include provision of support for the idea of not looking at the screen while dictating. Users may find this disorienting as they will want to know that the system is actually working, however this feedback could be provided by a prominent word counter or other symbolic animation to indicate that words are being recognised even though they are not being displayed on the screen.

Another long term goal is increasing the ability of the software to distinguish between dictation and commands.

A medium term interface goal could be a significantly improved inline correction method. However, this is not to suggest that inline correction should take the form it does today. Instead of a focus on correcting recognition errors the focus should instead be on misspoken words so that users are able to correct inline when they stumble over a word or quickly change their mind, for example:

User: We will provide fourteen, sorry that's forty places for attendees at the conference.

Display: We will provide forty places for attendees at the conference.

Finally, it is necessary in speaker dependant applications to adapt the acoustic and language models of the software to the user of the system. Typically, this is achieved through the correction dialogs however using these correction dialogs has been found to be error prone, or not possible at all (Halverson, Horn et al. 1999; Karat, Halverson et al. 1999). By allowing access to such functionality from the keyboard the user could control the without dealing with recognition errors. For example, imagine a short report on Cuba has been dictated and the user has completed dictation and is now beginning the proofreading stage. Every time the user spoke 'Havana' the system has transcribed 'banana'. Instead of re-speaking the correction the user could make the correction to the acoustic model with the keyboard and mouse. Then as the system has recorded all of the speech that compromises the dictation the newly changed acoustic model can be used to re-process the recorded speech and make all the subsequent changes automatically. These adaptations to the acoustic model could be described as 'post-speech' adaptations.

Following on from the idea of adapting the acoustic and language models during proofreading it is possible to consider the use of speech recognition in situations where the speaker is not the editor of the document. In this case the editor should be able to use the recorded voice of the speaker and the power of the voice recognition software to make corrections to the transcribed text. In this situation, a person, or group of people, is recorded and the speech is processed either in real time or later to produce a transcript. An editor then corrects any recognition errors in the transcript using the post-speech adaptations described above.

Applying the suggestions made above to the situation of a distinct speaker and editor leads to the following scenario:

> George owns and runs a small accounting practice. He employs two other accountants and one secretary, Emily. George is a classic hunt-and-peck typist when it comes to words but his fingers fly over the numeric keypad when he's using a spreadsheet. George dictates letters into dictation software for sending out to clients. Before he starts dictating he sketches an outline for each letter he will dictate. When he is ready, George starts his dictation system on his PC, picks up his hand-held mic and starts speaking. The screen doesn't show the words as he speaks them, instead a prominent counter shows the number of words spoken. George doesn't usually notice the counter as he doesn't look at the screen as he dictates but it can be useful to reassure himself that the software is working. When he's finished dictating George saves the file to the practice's small network and sends an internal email to Emily letter her know that there are some letters to edit and send out.
>
> Emily receives the email and opens the file George has saved. As the practice has just started using this method the acoustic and language models are not yet really well adapted to George's voice. In the case of these letters the software has misrecognised the acronym 'ATO', sometimes as 'eighty oh' and sometimes as 'hate Theo'. Emily can tell this because she is able to play the sound associated with the dictation as well as seeing the transcribed text. Emily selects the first wrong instance, changes it to 'ATO' and then instructs the software to re-transcribe the recording of George. Even though it took George 5 minutes to dictate the letter the re-transcription only takes 50 seconds and Emily is able to continue working on the errors that remain in the text. When Emily is satisfied that the letter is correct she has George confirm it is what he intended and the letter is then sent out.

CONCLUSION

Users bring incorrect or incomplete mental models to dictation software. Their mental models come from what they know about typing and word processing. Treating dictation the same as typing means that novices find dictation software hard to use. Novices make no effort the change their mental models because they want to be productive, that is create documents, not learn about how to correctly use the software. The interface to dictation software should subtly be re-designed to help the users build the correct mental models.

Supporting novice users' incorrect mental models by providing support for them makes dictation software harder to use. Because novices want to correct their dictation as they dictate the recogniser must separate dictation words from command instructions but in current technology the recogniser finds it hard to separate commands from dictation. When better separation of command and dictation

is possible it will be easier for novices to use dictation software because their mental model of typing will be much closer to the reality of dictation software.

The short term suggestions for a new dictation interface – hiding spoken navigation commands, encouraging multi-word dictation and providing task-oriented instruction – could be implemented in the next release of most dictation software. The medium and long term suggestions – allowing adaptations to the acousitic and language models to be made by mouse and keyboard, providing more natural inline correction for stumbles and mispronunciations and allowing editing of transcripts by editors - are not expected to be seen for some time.

Speech has the potential to be an exciting interaction paradigm for many computer applications however it must be recognised that speech is fundamentally different to using a mouse and keyboard and it is a disservice to users, particularly novices, to suggest that speech can be used as a replacement for them. Instead, applications that use speech recognition must carefully educate the user as to how speech is different from mouse and keyboard and how to best use speech to augment or replace traditional interaction devices.

REFERENCES

Carroll, J. M. and M. B. Rosson (1987). Paradox of the Active User. Interfacing thought : cognitive aspects of human-computer interaction. J. M. Carroll. Cambridge, MA, MIT Press: 80-111.

Halverson, C. A., D. B. Horn, et al. (1999). The Beauty of Errors: Patterns of Error Correction in Desktop Speech Systems. INTERACT '99, IOS Press.

Janal, D. S. (1999). Business Speak: Using speech technology to streamline your business, John Wiley & Sons, Inc.

Karat, C.-M., C. Halverson, et al. (1999). Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems. Proceeding of the CHI 99 conference on Human factors in computing systems : the CHI is the limit, Pittsburgh, Pennsylvania, United States, ACM Press.

Karat, J., D. B. Horn, et al. (2000). Overcoming Unusability: Developing efficient strategies in speech recognition systems. CHI 2000, The Hague, The Netherlands.

Newman, D. (2001). The Dragon Naturally Speaking Guide: speech recognition made fast and simple. Berkeley, Waveside Publishing.

Roman, E. (1971). The Art of Dictation. London, Gower Press.